# Wide Angle Laser Audio Eavesdropping Device

A Major Qualifying Project Report:
Submitted to the Faculty
of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the
Degree of Bachelor of Science
By


_____
Cody Brenneman

_____
Kevin Manelski


Date: 23 April 2007

Submitted to:

_____
Professor John McNeill, Co-Advisor

# Abstract

Eavesdropping on conversations using conventional microphones is often difficult due to the attenuation of acoustic waves over long distances. By using surfaces close to the speaker as the microphone and detecting changes in those surfaces from a distance, speech can be reproduced more accurately then using long range microphones. If a reflective surface such as a window were utilized in this manner then a laser and photodetector can be used to measure changes due to acoustic pressure. Additionally, laser light maintains its integrity over longer distances and is more efficiently detected then pressure from acoustic waves. We are proposing a device which can detect a conversation optically in a greater variety of environments then a microphone.

# Table of Contents

# Table of Figures

# Executive Summary

Human speech is an important means of transmitting information. The frequency range of intelligible speech is extremely narrow and low [3] and there is no hiding of information through encoding. As a result, speech detection is an important means of retrieving sensitive information. Detection of speech involves detection of acoustic wave's incident on a membrane, typically within a microphone. Microphones are not always ideal for detecting speech in situations where sensitive information is being exchanged due to their size and low sensitivity at a distance from the source. However, by separating the membrane from the device that interprets movement in the membrane, any surface with acoustic wave's incident upon it can become a microphone.

The goal of the Laser Audio Surveillance Device is to use a reflective surface (window) adjacent to a room as a microphone's membrane in order to detect speech within that room. Acoustic pressure incident on a window causes it to flex and by aiming a laser incident to that window, the flexing of the window can be interpreted as changes in light intensity at a photodetector. Laser light retains its integrity over much longer distances then acoustic waves so there is no loss of quality if the detector is placed a significant distance from the window. Laser light is also a much cleaner source then acoustic waves and can be amplified with little loss of quality.

However, detection and processing of laser light requires a much more complex interpretation system then a typical microphone. The laser light must be isolated from ambient light and because the window is anchored to a wall, further signal processing must take place to isolate acoustic waves from vibrations due to surroundings.

These problems were scrutinized during MQP projects for both the 2004/2005 and 2005/2006 academic years. Although both groups made significant progress in designing a serviceable Laser Audio Surveillance Device, there were still a number of obstacles that prevented then from producing an inexpensive, relatively simple and versatile product that can be used in a variety of environments.

For the current design, the device must be capable of detecting specular reflections of the laser light off the window rather than having the reflected beam hit the surface of the photodetector directly. This will greatly increase the range of use for the device rather then having the detector located directly across from the targeted window.

All the subsystems for the device have been mentioned. First, there is the laser driver which produces the laser light that is incident to the window. Next is the optical setup for focusing and collecting of the reflected laser light. The laser light is incident on a photodetector which makes up the optical front end or Transimpedience Amplifier, where the light is converted to an electrical signal. Then the signal is processed using Matched Filtering to increase the SNR enough to reproduce the audio signal accurately. Finally the signal is amplified enough to drive a speaker for acoustic reproduction.

However, the final system was not able to reproduce an audio signal because of one significant limitation. The bandwidth of the Matched Filter was not large enough to capture samples quickly due to limitations of the A/D converter for PIC microprocessor. As a result, testing of the design only took place for individual subsystem functionality and not for quality of the acoustic output. Because the limitation is innate to the device and not the system, the Laser Audio Surveillance Device can be made functional with a more suitable PIC and there are no design issues within the overall system.

# Acknowledgements

There are a number of people we would like to thank whose efforts made this project possible. First we would like to show our appreciation for the New England Center for Analog and Mixed Signal Integrated Circuit Design (NECAMSID). Specifically, the companies Allegro Microsystems in Worcester, MA, Analog Devices in Wilmington, MA, BAE Systems in Nashua and Merrimack, NH and Texas Instruments in Manchester, NH provided us with the financial means to investigate and implement a number of different project options that would have be unavailable to us otherwise.

We would also like to thank the previous project groups, specifically Phone Le and Benjamin Cooper of the 2004/2005 MQP group and Vincent Amendolare and Wade Sarraf of the 2005/2006 MQP group who provided us with a starting point for our design as well as insight into a number of difficulties we might face during implementation of the Laser Audio Surveillance Device.

Finally, we would like to thank our project advisors for their support and insight for the duration of our project. Professor John McNeill provided immense analog circuit design support as well as providing sponsoring for the project. Professor Brian King provided immense support for Optical systems and MATLAB as well as insight in into the 2 previous projects which kept our project moving forward. Their help was essential to the design and construction of the Laser Audio Surveillance Device.

# 1: Introduction

The surface of earth is crowded with waves of different forms, from acoustic pressure to electromagnetic radiation. Waves may also contain sensitive information, either through speech, radio or other means of communication. Current means of detecting speech in particular, at a distance from the speaker, involve parabolic microphones [2]. However, these devices have a limited range and poor low frequency response. The human vocal chords resonate at very low frequencies, less than 210 Hz [1] for adults, limiting the frequencies at which audible speech is produced. The limit for speech intelligibility is only 2048 Hz [3] making parabolic microphones less than ideal for eavesdropping on conversations.

Rather than detecting acoustic waves directly like the parabolic microphone, detection through an indirect method of measuring acoustic waves may produce a more accurate representation of human speech. One detector which does not deteriorate over distances as readily as pressure from acoustic waves is light. Light may function as a detector for acoustic waves by detecting inflections of a reflective surface near the source of speech, such as a mirror, or more commonly a window.

The window functions as the membrane of a typical microphone. Microphones use a membrane to translate the acoustic waves as a product of speech into an electrical signal, either through changes in capacitance of the membrane (condenser) , physical movement of the membrane (dynamic microphones) or by using a piezoelectric device [4].

The advantages of using light as an indirect detector for eavesdropping as opposed to direct detection of acoustic waves is that light can travel longer distances over

open space without loss of signal and suffers no attenuation for frequencies as low as speech.

We are proposing a device which uses a laser to detect acoustic waves within a room. The laser is incident on a reflective surface adjacent to the room, such as a window. Acoustic waves produced due to speech will exert physical pressure on the window causing it to flex. The flexing of the window will cause changes in intensity of the reflected beam which will be detected by a photosensor. The current produced by the photosensor will be treated to increase the signal to noise ratio and finally translated into an audio signal which will output to a speaker.

# 2: Background

## *2-1: Previous Project Work*

During the 2004/2005 academic year, Phone Le and Benjamin Cooper completed an MQP to construct a device capable of listening to a conversation within a room remotely. They achieved their goal by treating a window adjacent to that room like a microphone and using a laser beam to detect deflections in the window. The advantage of using a laser rather than a microphone placed in the room or a pressure sensitive device placed directly on the window is that it does not require entry into the room and is virtually undetectable to individuals present inside the room. The focus of their project was to make the device more inexpensive then other remote eavesdropping devices currently available. They were less concerned then the later projects with range of working environments and placement of the device.

Although their device had a more limited range of operation then later designs including ours, many of their individual subsystems could be adapted to our design with very few modifications. The first of those systems was the optical system. It was composed of the laser diode, the photodiode, lenses and the window surface. In the 2004/2005 project, the light reflected off the window is reflected back directly onto the photodiode. As a result, the photo detector has a greater proportion of the laser light intact then in our design, which only detects the specular reflection of the laser beam off the window. Although this makes detection easier due to the abundance of photons, the device must be nearly perpendicular to the targeted window. A diagram of their photo detection scheme is given in figure 2-1.

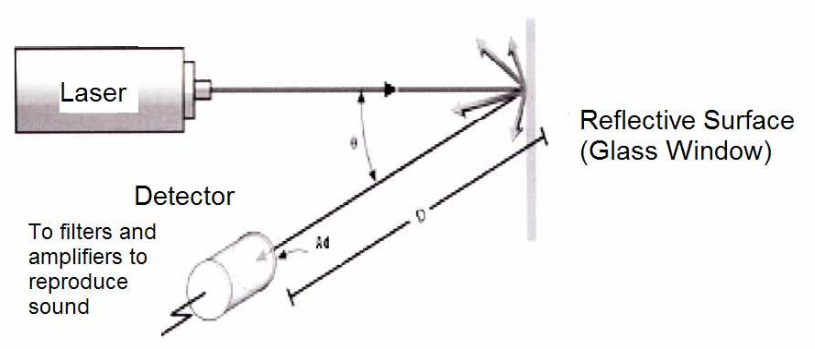


Figure 2-1 – Optical Detection Scheme

The second system which may be adapted by our design is the sound reproduction system. Their system consisted of a prebuilt audio amplifier, the LM 386, and a supporting network of resistors and capacitors. This stage does not need to be modified to accommodate our design as long as the input voltage range remains the same.

A system level block diagram of their entire project is given in figure 2-2.

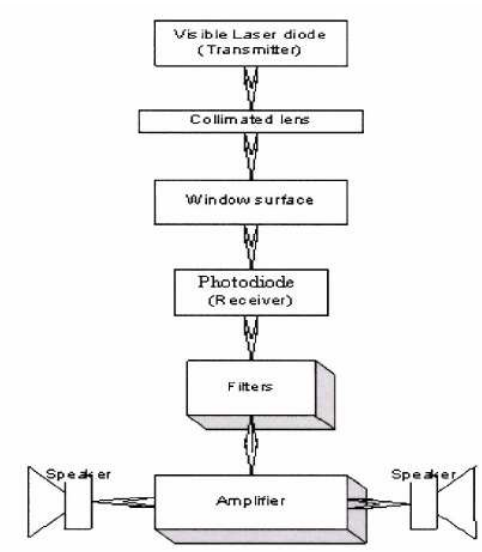


Figure 2-2 – System Level Block Diagram of 2004/2005 Project

The remaining subsystem, the filter, consists of amplifiers and a band pass filter. Our design may contain these components but signal conditioning within that block is different due to the much lower light levels detected at the photodiode. Additionally, our laser diode will be modulated differently and requires a different design then their laser driver. The next project to improve upon this design was completed in the 2005/2006 academic year.

During the 2005/2006 academic year, Vincent Amendolare and Wade Sarraf completed an MQP with the same functional intent as ours. Vincent and Wade aimed to construct a device capable of eavesdropping on a conversation within a room with a window. They had an identical systems level block diagram to ours and their outline gave us a solid foundation for our project. That project group was limited however, in terms of time and budget. Their project was not completed due to a number of difficulties they encountered which are described in detail in their report. Additionally, they made a number of recommendations for future MQP groups working on similar projects.

Their laser driver worked as expected but was not refined given the limitations discussed above. They recommended implementing an IC laser driver with modulation capabilities to control the laser which was incident on the window. Additionally, they recommended that the optical noise floor of the beam, discussed in a later section, be reduced through some sort of feedback system linked to the optical output of the laser. These changes result in a more flexible and stable light source, which would be easier to isolate at the detection stage.

The most significant limitations to Vincent and Wade's system were in the detection stage. The modulation frequency, which allows for the system to differentiate the reflected laser light from ambient light, was limited by the photo detector. Increasing the modulation frequency allows the system to gather samples more quickly which increases the frequency of sound that can be detected. Another recommendation was to use low noise components, especially the Op Amp, which was a significant source of interference.

The clamper stage, which removed the DC offset from the detection stage output, was also a source of interference. Because the noise was already present at ground, the clamper just reduced the positive gain of the desired signal while leaving the noise constant. As a result, the clamper output was completely saturated with noise. The previous project group recommended completely eliminating the clamper stage and redesigning the succeeding systems.

The analog multiplier functioned as expected but had several drawbacks our design should avoid. Because they modulated their signal with a square wave to identify it, their multiplier was a single MOSFET that would ground the signal when it was at 0V DC and multiply the signal by 1 when it was greater than 0V. The signal multiplier, which is the first stage in signal processing, needs to be redesigned to accommodate higher frequency signals with negative as well as positive voltages. By using a signal multiplier IC instead of a MOSFET, a greater variety of modulating waveforms may be implemented.

The integrator is the final stage that the previous group made recommendations to improve. Their integrator did not reset itself completely with each sampling cycle which

gave each sample an ever increasing offset which made the signal impossible to distinguish after several samples. Our integrator needs to be completely reset with each sampling cycle and contribute as little interference as possible to our system.

The remaining subsystems existed within a PIC microprocessor and involved a pulse width modulation scheme to translate the integrator output to an acoustic output. Our PIC microprocessor will have multiple functions within the system and will follow a different programmed design then theirs. A system level block diagram is presented in figure 2-3.

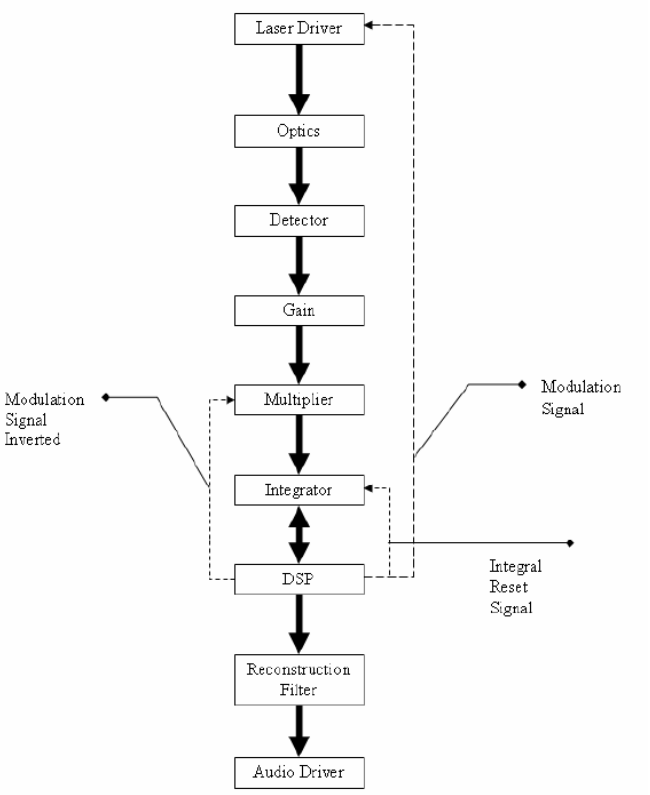


Figure 2-3 – System Level Block Diagram for the 2005/2006 Project

The previous year's project gave us a number of insights into the difficulties we will face as we design our surveillance device. Each module needs a significant amount

of work before the entire system is functional and their advice gave us a starting point for researching solutions to these potential problems.

## *2-2: Reducing Noise in Electrical Circuits*

Noise is present in every electrical circuit and although there is no way to completely eliminate its presence, there are various methods for reducing it to a negligible level. This is a critical step in creating a serviceable Wide Angle Laser Audio Surveillance Device due to the low power signals being detected. Without noise reduction, the small signal from the photo detector will have a power level less than noise and will not be easy to distinguish and amplify. Noise sources that may affect the performance of the device are described as well as techniques for reducing interference.

Proper shielding and grounding are effective methods to reduce noise within a circuit. They can both be incorporated into one noise reduction network that should be used with any low power and or high frequency signal. When connecting between separate, isolated circuits, wires can be shielded from magnetic interference through twisted conductor-ground pairs. This form of shielding is most effective at frequencies below 100 kHz where magnetic interference would have the greatest impact. A coaxial cable should be used at frequencies above 1 MHz because it has uniform characteristic impedance and provides protection against capacitive coupling if one end is grounded around the complete circumference of the cable. A conductive metal box to surround the circuit provides extremely good protection against ambient electromagnetic interference and if coaxial cables are the only electrical breaks in the shield than the effect of ambient electromagnetic waves should be minimized [5].

Grounding of the components to the shield should be done at one junction for lower frequencies and the components should be grounded as close to their terminals as possible for higher frequencies. Figure 2-4 demonstrates one technique for incorporating both low and high frequency grounding into one system. The next step after establishing a circuit layout is to populate it with passive and active components which must be chosen to minimize signal distortion [5].

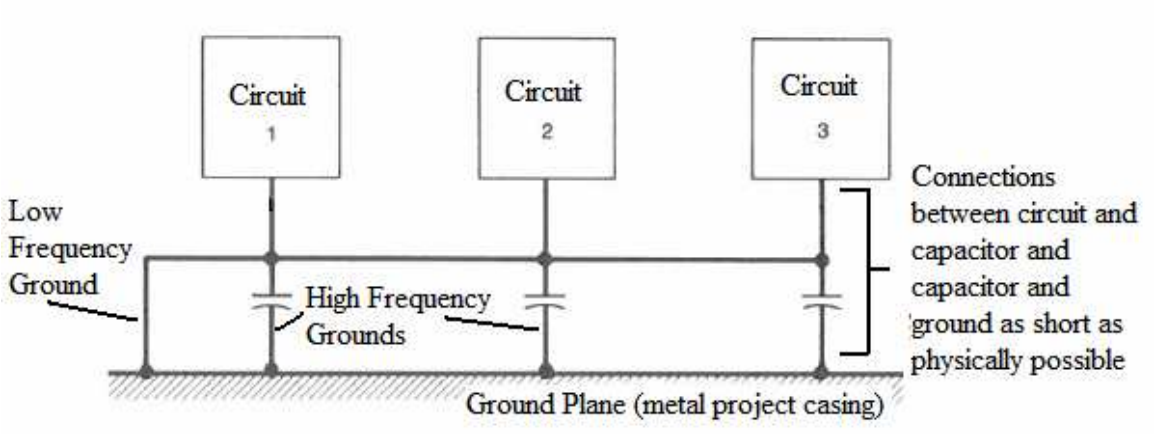


Figure 2-4 - Hybrid Grounding Scheme

Due to the variety of components available between resistors and capacitors, selecting components for a given task can be difficult. Capacitor selection should be made based on operating frequencies, resistance and inductance. Electrolytic capacitors are the worst for noise reduction because they have a high resistance, high inductance and poor switching properties. The advantage of electrolytic capacitors is the high capacitance to volume ratio. Paper and Mylar capacitors have a higher operating frequency range than electrolytic capacitors and are often used in noise reduction. Mica,

glass and low loss ceramic capacitors have an even higher frequency operating range than paper and Mylar, up into the hundreds of megahertz and polystyrene have the best noise and frequency range properties of all the above mentioned components. In situations where a high frequency response and high capacitance are needed, capacitors of different dielectrics may be combined in parallel to take advantage of the positive traits from each such as wiring an electrolytic and a polystyrene capacitor to obtain a large capacitance without taking up excessive space with many polystyrene capacitors.

Resistor selection is very simple due to the limited number of resistor types available. Inductance can be neglected from all resistors except for wire wound. Capacitance is only a problem in resistors larger than 1 MΩ. Finally, a high power rating usually equates to a lower noise level so high power resistors are well suited to this application despite the low power signals. All resistors generate noise because all dissipate heat. Composition resistors generate the most noise, film resistors generate less noise and wire wound generate the least noise.

Active components such as op amps and MOSFETs are not as easily broken down due to their complexity. For the Op Amp, noise levels are typically given in specification sheets. One Op Amp that may be used in the TIA is Analog Device's AD8033. Table 2-1 and figure 2-5 are noise values included on the specification sheet.

| | **Conditions: T = 295 K, $V_S$ = ±5 V, $R_L$ = 1kΩ, Gain = 2** | **Typical Values** |
|---|---|---|
| **Noise/Harmonic Performance** | | |
| **Distortion** | **$F_C$ = 1 MHz, $V_O$ = 2V pk-pk** | |
| **Second Harmonic** | **$R_L$ = 500 Ω** | **-82 dB** |

| | | |
|---|---|---|
| | **$R_L$ = 1 kΩ** | **-85 dB** |
| **Third Harmonic** | **$R_L$ = 500 Ω** | **-70 dB** |
| | **$R_L$ = 1 kΩ** | **-81 dB** |
| **Crosstalk, Output to Output** | **F = 1 MHz, Gain = 2** | **-86 dB** |
| **Input Voltage Noise** | **F = 100 kHz** | **11 nV/Hz$^{1/2}$** |
| **Input Current Noise** | **F = 100 kHz** | **0.7 fA/Hz$^{1/2}$** |

Table 2-1: Noise Specifications for AD8033

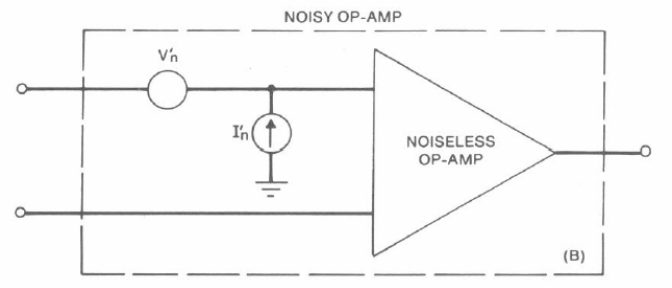


Figure 2-5 - Op Amp Noise Model

The values for noise voltage and current in table 2-1 are represented in figure 2-5. The noise voltages and currents for each input are combined into 2 terms. Calculating the actual $V_N$ and $I_N$ of an input can be accomplished by dividing the given values by $2^{1/2}$.

Figure 2-6 is not used in calculating noise but gives a general picture of noise versus frequency which is dependent on the application. The frequencies with the lowest noise are the modulating frequencies that will most likely be used at the laser output stage making this a suitable amplifier within our system.

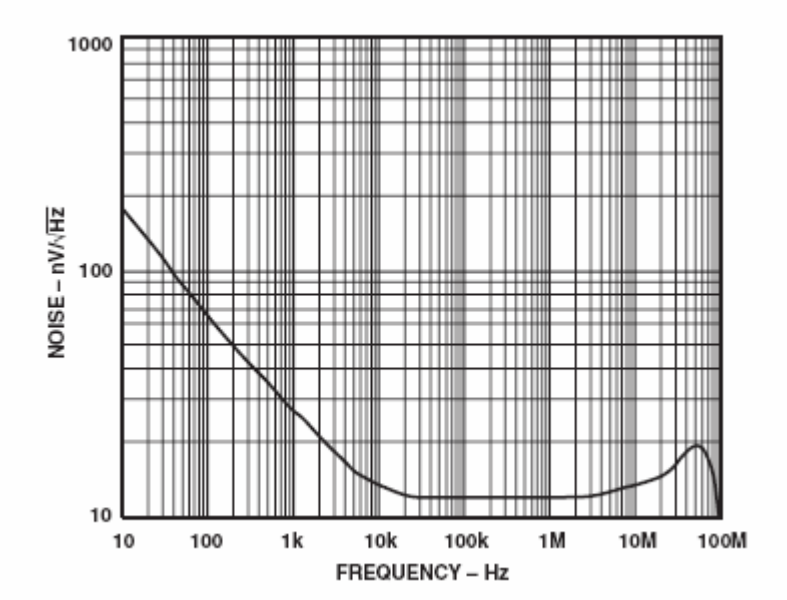


Figure 2-6 - AD8033 Noise Response vs. Frequency

### 2-2-1: Reducing Optical Noise

Regardless of the source and detector used there are a number of methods for reducing noise optically. For reducing modulated background noise, an optical filter can be used. The filter should be mounted with its coating facing the detector or else the filter will radiate IR into the detector.

Another method to reduce modulated background noise is to reduce the field of view of the detector. The aim is to reject undesired photons and as a result the interior of the field limiting device should be coated with light absorbing material.

Choosing the smallest detector possible reduces the capacitance of the photodiode. However, for pulling the weakest signals out of ambient light, a larger detect will be more beneficial because the background light will increase the current output more slowly than the coherent laser light for a given photodetector.

## 2-3: System Block Diagram

One of the major contributions of last year's projects was a system block diagram. The diagram shows each subsystem and the role it plays in the overall functionality. Our block diagram is shown in figure 2-7.

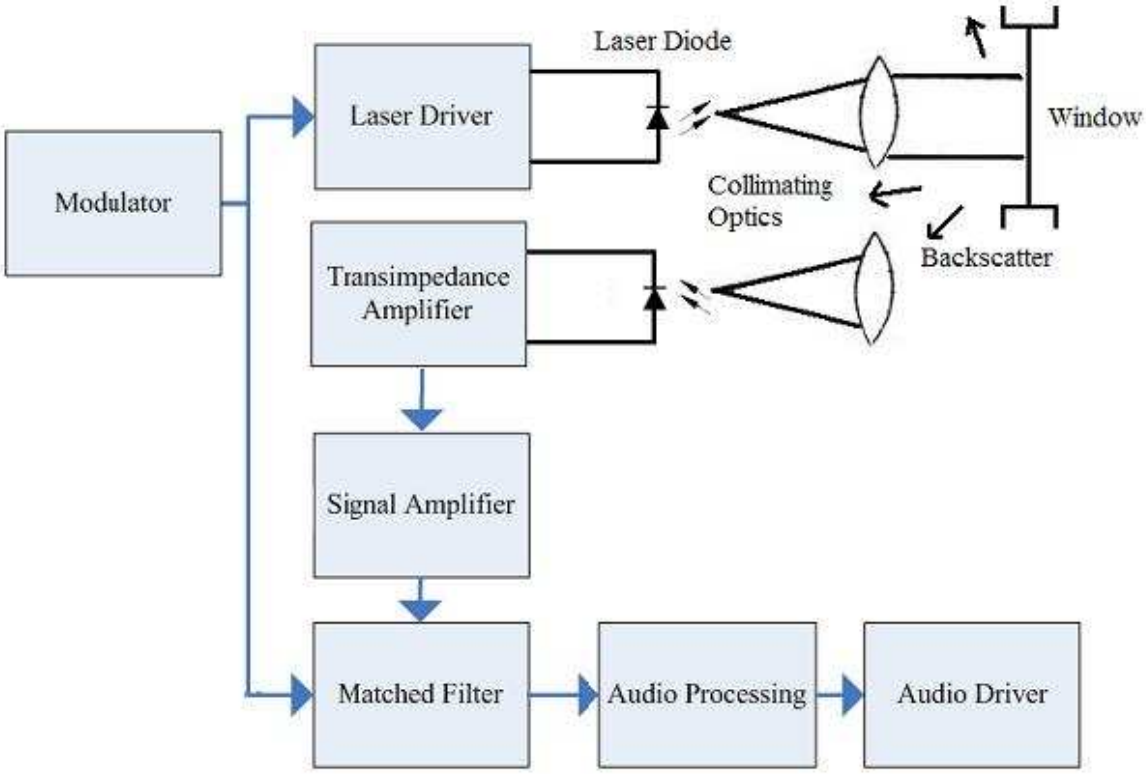


Figure 2-7 - System block diagram

Each system block presents design challenges that must be addressed. The following sections describe the functionality of each block, as well as the challenges and goals associated with them.

### 2-3-1: Transimpedance and Signal Amplifiers

The laser reflected from the window is detected using a photodiode. The photodiode produces a small current proportional to the amount of light incident on its surface. The photodiode presents a critical tradeoff. The more sensitive the photodiode is the more light we can detect. However, the larger the detector surface the larger the diode's capacitance, which reduces the overall bandwidth. One of the major goals of the transimpedance amplifier block is to select a photodiode that meets the sensitivity and capacitance constraints of the system.

In order to process the signal we receive, we must convert the photodiode current into an electrical potential. The current produced by the photodiode is very small, so the circuit must have a large gain. At the same time the noise should not have any gain during amplification so it does not overwhelm the information contained in the signal.

We have chosen to implement an op-amp transimpedance amplifier because of its bandwidth capabilities. A large bandwidth is necessary for the matched filter stage to be effective. The biggest challenge of the transimpedance amplifier block is to increase the bandwidth of the circuit without significantly increasing the noise. We have researched some techniques such as the cascoded TIA configuration which should minimize the bandwidth limitations set by the photo diode capacitance.

The signal amplifier will consist of a non-inverting op-amp configuration. The challenge of this subsystem is to keep the noise gain to a minimum. The noise analysis of our system is essential to minimizing the noise gain without decreasing the bandwidth.

### 2-3-2: Matched Filter

Matched filtering is a technique used to maximize the SNR of a system. The matched filter block multiplies the incoming signal with the known modulation signal. The output is then integrated for a pre-set amount of time before the cycle is reset. The output of the integrator at the end of a cycle is proportional to the amplitude we are looking for. This technique keeps the noise floor relatively constant while increasing the desired signal.

The biggest challenge of the matched filter block is the multiplication stage. Specifically, the demodulating signal must be in phase with the TIA output. If the incoming signal is ninety degrees out of phase with our multiplication signal, the output will be zero. To begin to tackle this problem we researched demodulating with quadrature phased signals. Another challenge for the matched filter is designing the integrator because it needs to integrate quickly and must work with a range of potentials.

### 2-3-3: Laser Driver

The laser driver is a current source driven by the PIC microprocessor. The driver has the capability to modulate at the speed we require, while keeping a bias on the laser so it does not turn off completely. The biggest challenge of this block was to determine the best and most cost effective solution for our application.

### 2-3-4: Modulator

The modulator is connected to the laser driver, demodulator and integrator blocks and is dependent on the design of the TIA. For the matched filter to work effectively, we needed a high modulation frequency. This modulation signal is used to drive the laser

diode through the driver circuit. However, the frequency is limited by the bandwidth provided by the transimpedance amplifier. The modulator thus presents a challenge of selecting the optimum frequency for our application, weighing the tradeoffs between matched filter resolution and transimpedance amplifier bandwidth.

### 2-3-5: Audio Processing and Driver

The audio processing portion of the system will be implemented last. We have not had much time to research the methods at this point but we know the processing portion seems to favor the digital domain. With the matched filter portion of the system working properly the processing should be relatively simple. The driver portion will simply consist of a power amplifier to drive a speaker which we have not implemented.

# 3: Design

## *3-1: Front End*

The light gathering component of our front end receiver is its most sensitive part of due to the fact that lost photons can not be recovered. The goal is to gather as many targeted photons as possible before they are lost, the cleaning up of the signal can be taken care of electrically.

Photodiodes are the preferred device for detection of optical signals due to their relatively low cost, high speed and high quantum efficiency (~90% of the light incident to its surface is converted to electricity). Additionally, their conversion from light to current is linear due to the functional nature of semiconductors, making performance very predictable. Nonlinearity is the result of extremely high intensities of light and as a result, a larger diffuse spot will product a more linear current response than a high intensity, tightly focused spot. This establishes that detecting the specular reflection of laser light will result in a linear current at the photodiode.

### 3-1-1: Transimpedance Amplifier

The photodiode that will receive the reflected light signal produces a current proportional to the amount of light it detects. This current is on the order of micro-amps, and thus must be amplified and converted to a usable voltage signal. This is accomplished using a current-to-voltage converter, also known as a Transimpedance Amplifier (TIA). The ideal TIA consists simply of an op-amp and a feedback resistor.

If all components followed their ideal characteristics there would be few challenges for circuit designers to overcome. Unfortunately there are always non-idealities that must be dealt with and their effects analyzed. In the case of operational amplifier circuits, the parasitic capacitances of external components can cause the amplifier to become unstable. The ideal TIA circuit is shown in figure 3-1.

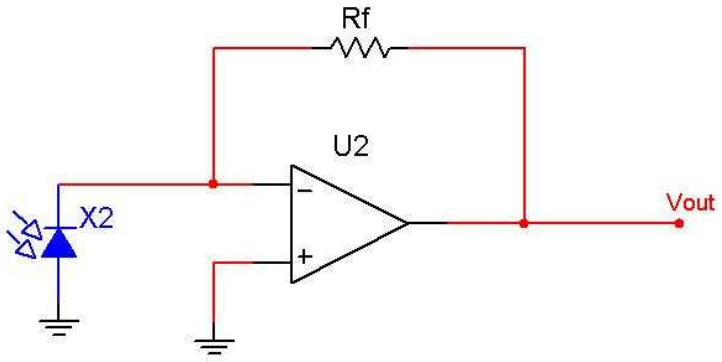


Figure 3-1 - Ideal Transimpedance Amplifier

In the ideal case, the current produced by the photodiode will all flow through $R_f$ producing a voltage at $V_{out}$ proportional to the current produced. The current is thus converted to a voltage with a gain of $R_f$, which was the goal of this amplifier stage.

### 3-1-2: Stability

Unfortunately, the amplifier in figure 3-1 is unstable when built using real components. The photodiode has a parasitic junction capacitance associated with it. This capacitance, combined with the feedback resistor $R_f$ produces an unexpected pole in the frequency response of the amplifier. Figure 3-2 shows the new amplifier with the diode capacitance included.

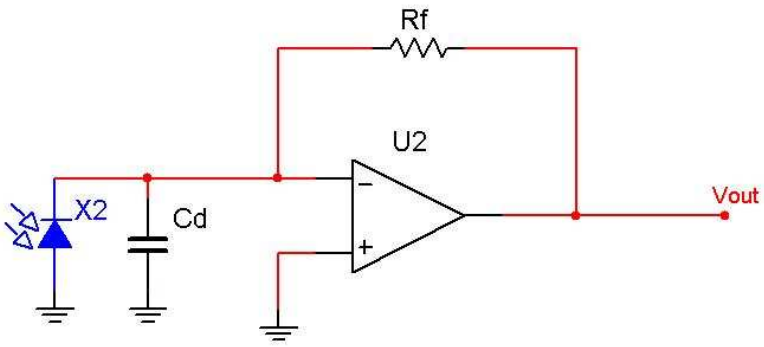


Figure 3-2 - Amplifier with parasitic diode capacitance

To analyze the stability of this circuit, first we must determine the feedback network and examine its frequency response. To find the feedback network, all sources first must be suppressed using superposition. The photodiode can be modeled as a current source and it is redrawn as an open circuit. To see the feedback network more clearly, the photodiode is removed to create an open circuit, and the circuit can be redrawn as shown in figure 3-3.

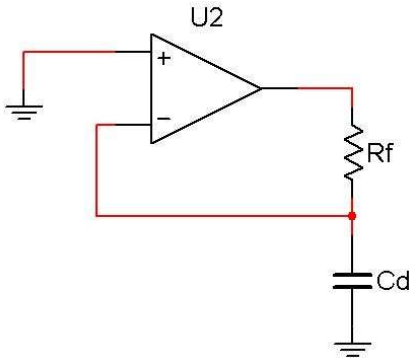


Figure 3-3 - Feedback network redrawn

The feedback network, referred to from now on as beta (β), is now seen to be a voltage divider formed by the diode capacitor impedance and the feedback resistance. To understand the effect of this feedback situation we must find the frequency response of

the network and compare it to the inherent frequency response of the op-amp. To find frequency response of β, we must use the voltage divider formula, shown here:

$$\beta = \frac{Z_{Cd}}{Z_{Cd} + Z_{Rf}}$$

Where $Z_{Cd}$ is the impedance of the diode capacitance, which is defined as

$$Z_{Cd} = \frac{1}{sC_d}$$

and the impedance of the feedback resistor $Z_{Rf}$ is simply

$$Z_{Rf} = R_f$$

Substituting the impedances in to the voltage divider formula shown above:

$$\beta = \frac{\frac{1}{sC_d}}{\frac{1}{sC_d} + 1} = \frac{1}{1 + sR_f C_d}$$

To extract information about the stability of the system from the previous equation, we must find the frequency value that makes the denominator approach 0. This value is called a pole of the system. The pole of this system exists at a frequency of

$$f_p = \frac{-1}{2\pi R_f C_d}$$

As an example, we chose to use the values the 2005/2006 project group used. The capacitance of the diode was approximately 15 pF and they used a 200kohm feedback resistor. These values place the pole of the beta network at f = 53 kHz. A pole causes the magnitude response of the network to decrease at a rate of –20dB/decade and also introduces a phase lag of 90 degrees. These values are important in determining the stability of the system as a whole. To determine the system stability we must look at the

overall transfer function of the system, shown in the following equation, where A is the gain of the op-amp as a function of frequency.

$$H = \frac{A}{1 + A\beta}$$

To find the poles of the overall transfer function can be quite challenging, so instead we look at the boundary conditions. These conditions are where the magnitude of A multiplied by β equals unity and the total phase lag around the loop equals at least 180 degrees. If these two conditions are met, then the system will be unstable and the phase lag of the loop will need to be compensated. Figure 3-4a shows the open-loop gain response of the AD8033 op-amp, using an integrator transfer function for simplicity.

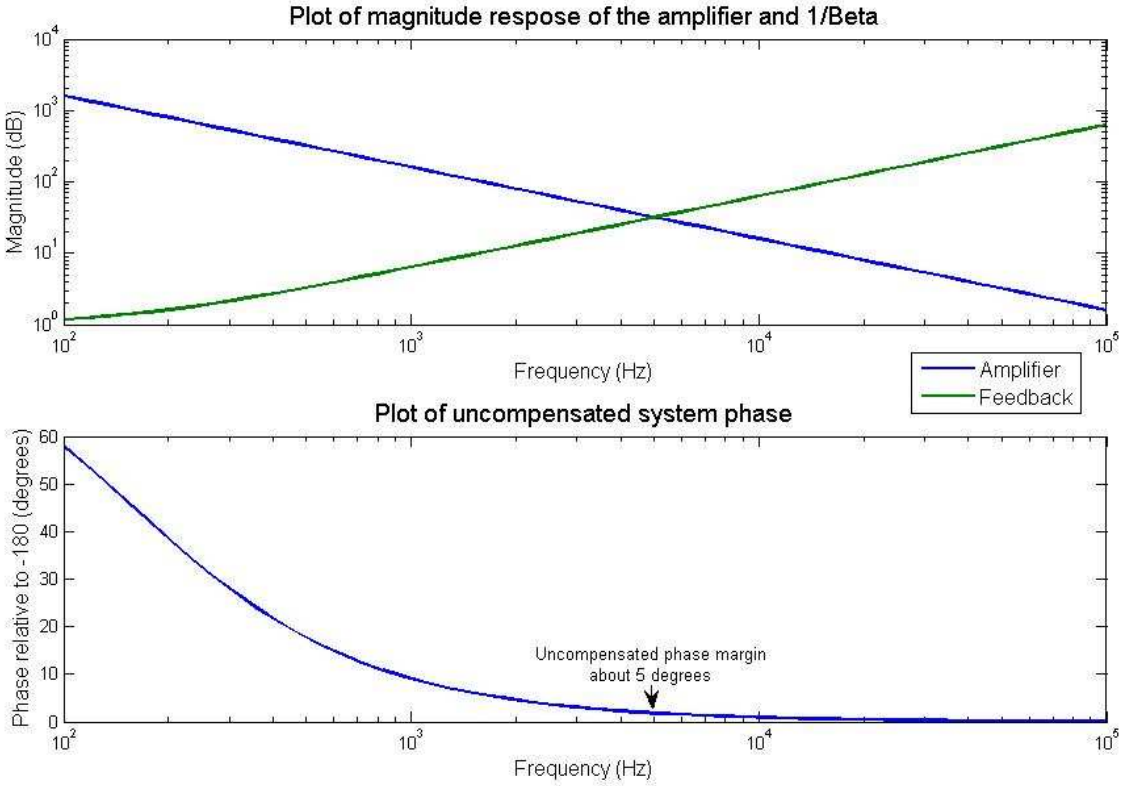


Figure 3-4 - Frequency response of A and Beta

Also shown in figure 3-4a is the magnitude response of β. For convenience, we graphed 1/ β so we could simply look at where it intersects A to determine where the magnitude of A·β is equal to unity. Figure 3-4b shows the phase of the output with respect to -180 degrees. We show the phase in this manner because we are primarily interested in the phase margin value, which is the difference between unity gain phase and -180 degrees. The total phase lag of A and β combined is very close to -180 degrees when A·β is unity, with a phase margin of only 5 degrees. This means the system is close to being unstable and the output will oscillate.

Another bad side effect of the unwanted pole is that it limits the bandwidth of the system. After the pole frequency, the feedback network begins to attenuate the signal. The bandwidth is thus essentially limited to the frequency where the op-amp gain can no longer support the feedback attenuation. This frequency is at the magic point where A·β is unity, in this case about 50kHz.

### 3-1-3: Compensation

To counteract the instability of the system, a zero must be added to the response. A zero causes an increase in the magnitude gain of 20dB/decade and introduces a phase lead of 90 degrees. Introducing a phase lead will keep the A·β unity point away from -180 degrees. The best way to introduce a zero is by putting a small capacitor across the feedback resistor, as shown in figure 3-5.

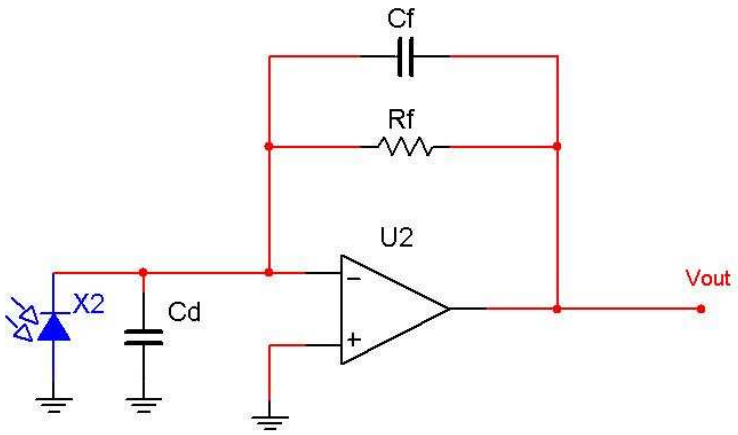


Figure 3-5 - TIA with feedback capacitor

Using the same procedure as we did previously, we find the feedback network transfer function to be:

$$\beta = \frac{1 + sR_f C_f}{1 + sR_f (C_f + C_d)}$$

Solving for the roots of the numerator and denominator we find the zeros and poles of the feedback network respectively.

$$f_z = \frac{-1}{2\pi R_f C_f} \qquad f_p = \frac{-1}{2\pi R_f (C_f + C_d)}$$

Following the example of the previous project as a starting point, we chose $C_f$ to be 3.3pF. Using this value in the pole and zero formulas we get $f_p$ = 43kHz and $f_z$ = 241kHz. The plot of the magnitude and phase response is shown next in figure 3-6.

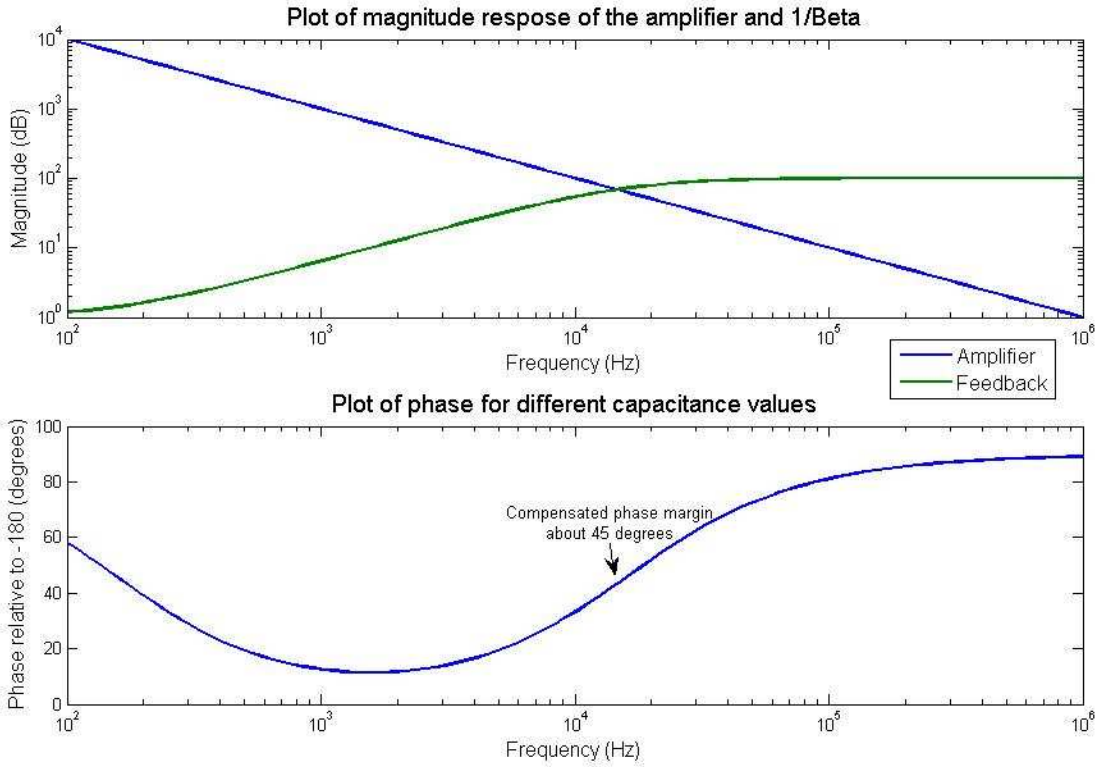


Figure 3-6 - Frequency Response of A and Beta with Feedback Zero

The plot now shows the total phase lag of the system when A·β is unity to be about 135 degrees. This corresponds to a phase margin of about 45 degrees, which is enough to ensure stability of the system. We can also see the drawbacks of adding the zero into the feedback network. The extra capacitance reduces the overall bandwidth of the system from 50kHz to about 15kHz.

The balance of values for the feedback resistor and capacitor along with the diode capacitance provides an interesting design challenge for this amplifier. We must select the values such that we get a high enough gain with the resistor, yet a wide enough bandwidth to accommodate our signal. The bandwidth, as we have seen, is set by a combination of the feedback and diode capacitances.

### 3-1-4: Bandwidth and Noise

Bandwidth and noise are closely coupled throughout our design. To get the highest SNR output signal, we will implement a matched filter. Although the audio signal we wish to recover only has a bandwidth of about 4kHz, for the matched filter to increase the SNR it requires that our incoming signal be modulated at a frequency much higher than 4kHz.

Our original intuition told us that the higher the modulation frequency, the greater the SNR increase would be. Our goal was thus to have a modulation frequency of about 1MHz. But, as shown in the discussion on the matched filter, our MATLAB simulations revealed to us that it is the integration time that governs the SNR increase, not the modulation frequency. This discovery greatly relieved our bandwidth constraints and we were able to reduce our modulation frequency to 100kHz.

The basic TIA topology, though, still did not provide enough bandwidth for our 100kHz signal. We thus began our research into improvements to the basic circuit. We compared three different topologies, the cascode, the bootstrap and a combinational circuit. The simplest of the three is the cascode, shown below in figure 3-7.

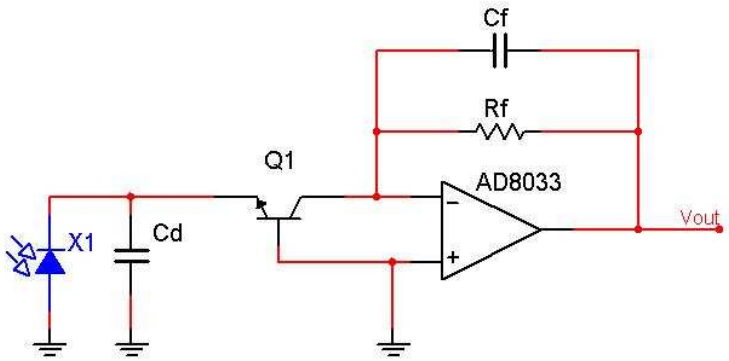


Figure 3-7 - Cascode TIA

The cascode is a very effective method of increasing the bandwidth of the TIA. The transistor transfers the photodiode current at its emitter to the collector with very little attenuation. The transistor also shields the op-amp from the large diode capacitance. The bandwidth is thus now governed only by the feedback, op-amp and transistor collector-base capacitances. The effective removal of the large diode capacitance greatly increases the bandwidth.

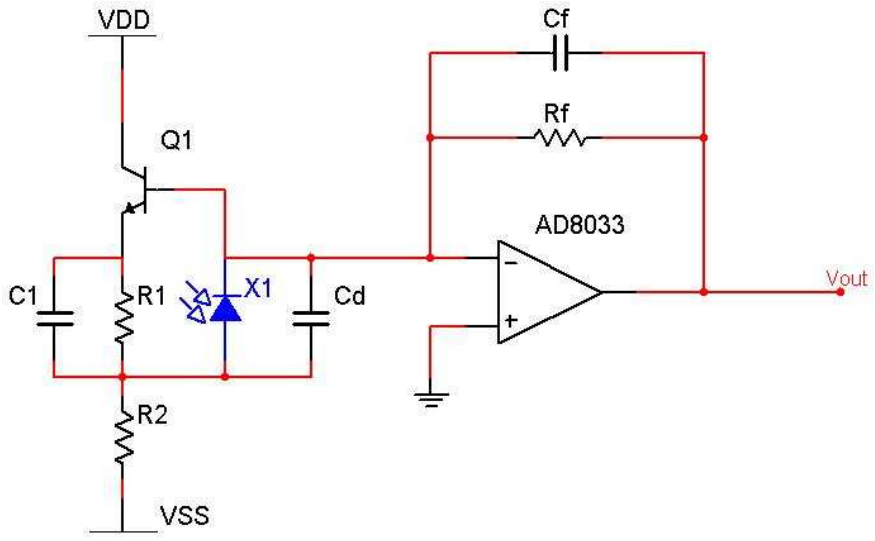


Figure 3-8 - Bootstrap TIA

Figure 3-8 shows the schematic for a bootstrapped TIA. The bootstrap is an active method of reducing the photodiode capacitance. Essentially, any voltage that appears at one end of the capacitance is "mirrored" to the other side of the capacitor. Thus the signal swing across the capacitor is very close to zero, and the effect of the capacitance is eliminated. Combining the two techniques is the most complicated, but also the most effective at increasing the bandwidth. The combined schematic is shown below in figure 3-9.

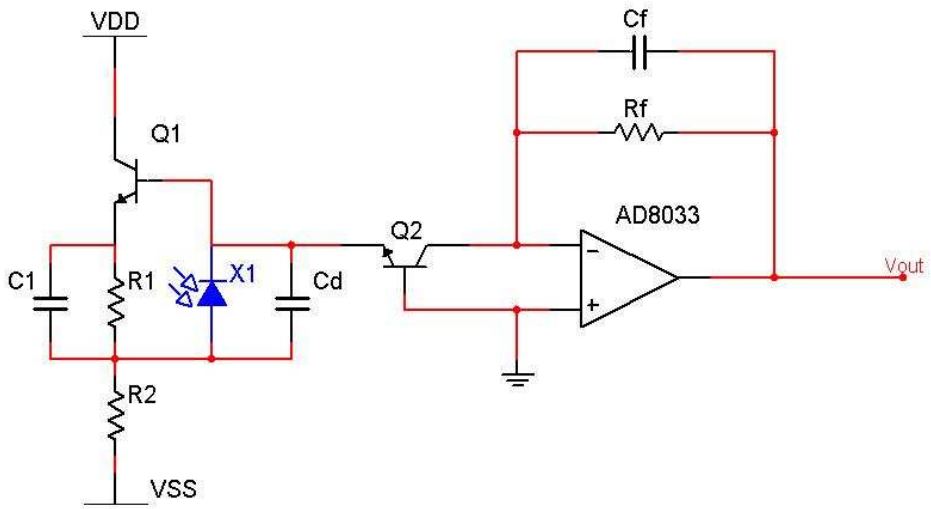


Figure 3-9 - Bootstrap and Cascoded TIA

The combination of active capacitance canceling and shielding produces the largest bandwidth of the three. But, this bandwidth comes at the cost of circuit complexity and poor noise performance.

To make a fair comparison of the three choices, we simulated them all in LTspice. We then exported the data and made comparison plots in MATLAB. We then compared the bandwidth and noise density plots generated by our simulations of the three topologies.

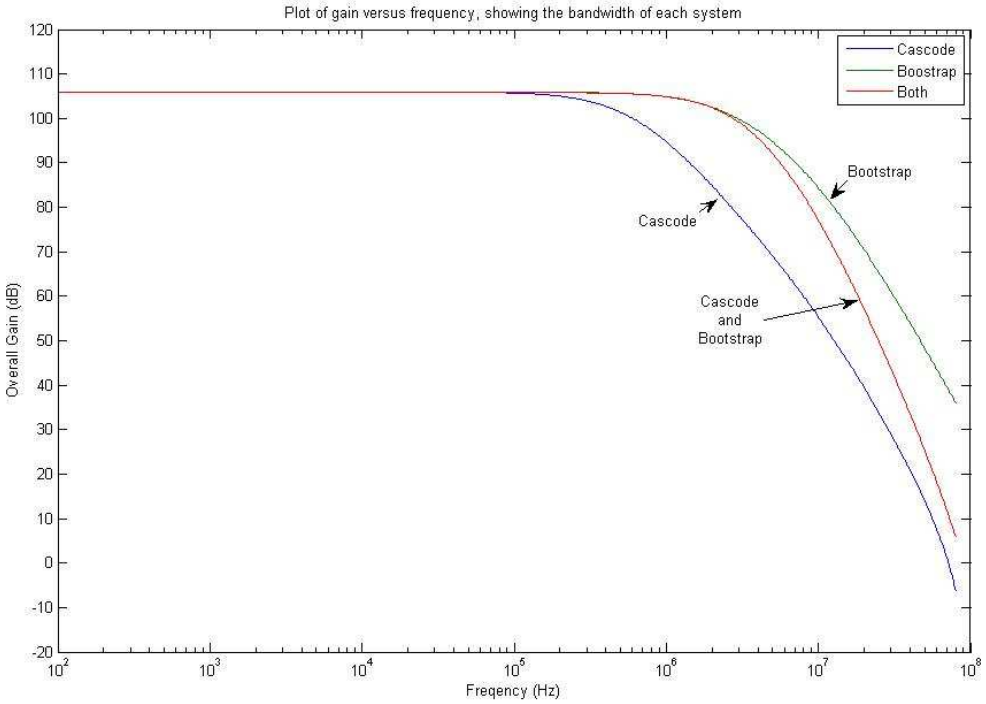


Figure 3-10 - Bandwidth comparison for three TIA topologies

As seen in figure 3-10, all three topologies meet the 100kHz bandwidth requirement. Our initial feeling was to implement the cascode and bootstrap, because it has the highest bandwidth. But, after our discovery that high modulation frequency is not the priority, we began to rethink our decision. To help solidify our decision we also examined the noise density plots for the three choices, shown in figure 3-11 below.

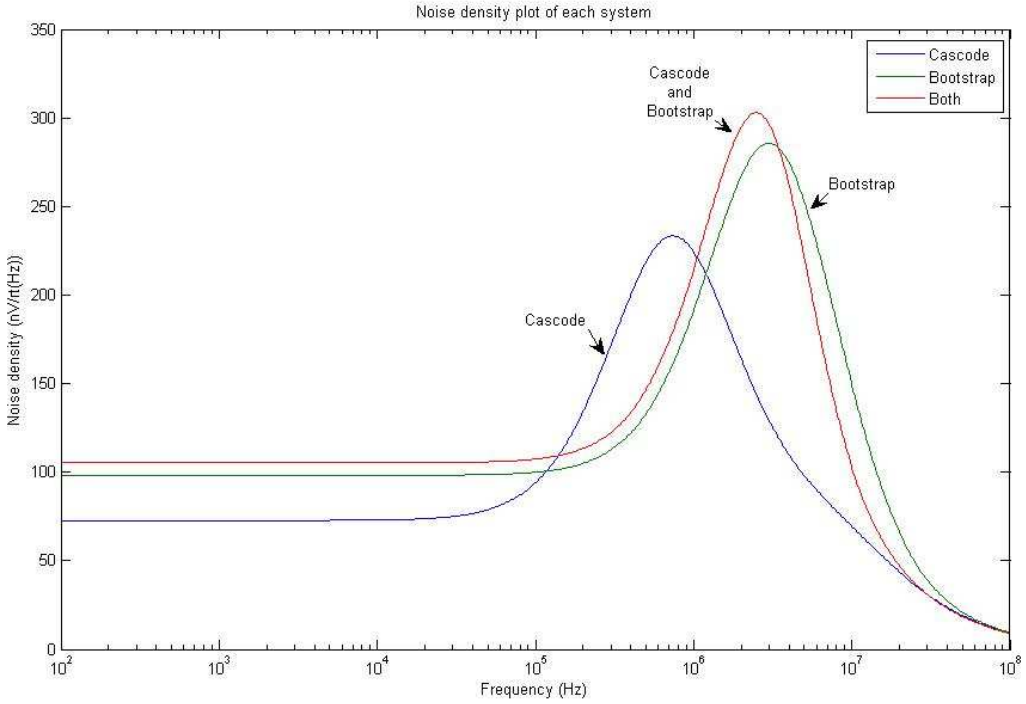


Figure 3-11 - Noise Density Comparison for three TIA Topologies

As our intuition dictated, the simplest of the three topologies yielded the lowest overall noise density. To figure out the total noise in the system, we would have to integrate the noise density curve over our bandwidth range. The matched filter, though, acts as an extremely narrow bandpass filter, essentially eliminating any frequency that does not match its modulation frequency. Because of this, the overall density curve is not as important as the specific noise density value at 100kHz. The cascode also has the lowest 100kHz noise density value of the three choices.

Thus, the cascode configuration seems to be the best compromise between noise and bandwidth considerations. Also, as an added "bonus", the cascode configuration is the simplest of the three to implement, involving the addition of only a single transistor

and some biasing resistors. The bandwidth is large enough to accommodate our 100kHz switching frequency, and the noise density around 100kHz is only about 90nV/√Hz.

## *3-2: Matched Filter*

Matched filtering is a technique developed for use in RADAR to eliminate interference in the reflected signal. The same technique can be applied to any noisy signal as long as we know the period and phase of the original signal. The noise present in the reflected signal is typically white gaussian noise but the matched filter can also reduce noise that varies as a function of frequency.

Mathematically, the filter is a convolution of the original modulating signal with the reflected signal plus noise(TIA Output). The filter as a result maximizes the peaks of the reflected signal at the points where the original waveform peaked and takes the sum of the power of the larger peak signal over a fixed sampling period. The sum of the power increases over time for the original signal while interference, because it has a flat power spectral density, fluctuates but does not contribute significantly to the sum of power. The longer the integration period, the more distinguished the signal becomes within the noise. However, a longer period also reduces the sampling rate for the output of the integrator, which must be larger than the highest frequency of sound we wish to discern, 4 kHz.

The impulse response for the matched filter is given below.

$$h_0(t) = f(t_1-t)$$

The impulse response of the filter is the original signal $f(t)$, 180 degrees out of phase and given a time shift $t_1$ that accounts for the period of the original signal. An example of a matched signal is shown in figure 3-12.

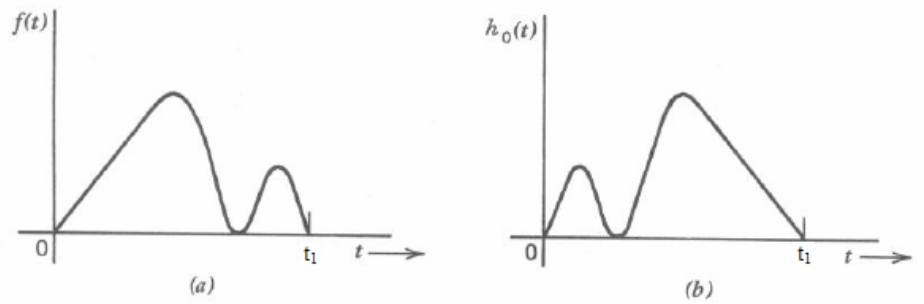


Figure 3-12 – Original Signal(a) and the Matched Signal(b) to Convolve With

A block diagram of a matched filter is given in figure 3-13.

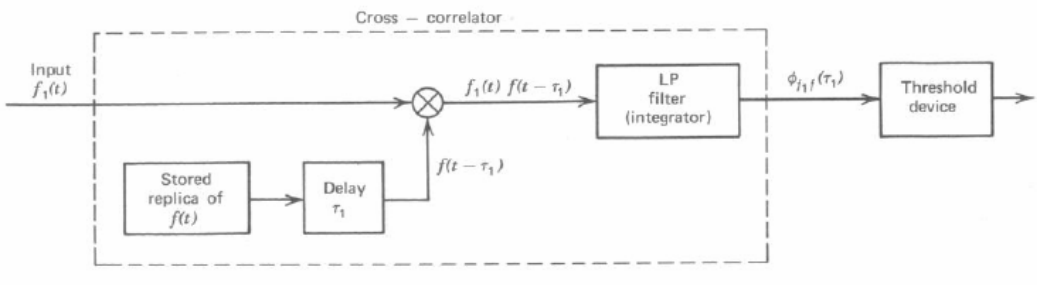


Figure 3-13 – Matched Filter Block Diagram

$F_1$(t) is the reflected signal plus interference which is multiplied by the modulating signal *f*(t) with a phase delay $\tau_1$ and is then integrated and samples are taken when they reach a desired threshold which is several times larger than the noise threshold. Although the integrator in the block diagram is a low pass filter, an Op Amp integrator will increase flexibility by making the integration period variable and eliminating the threshold detection block.

In order for the matched filter to work optimally, the matched signal must be in phase with the reflected signal to give the largest power increase prior to integration. This is often the most difficult part in designing a matched filter and numerous solutions exist. Using a Phase Locked Loop (PLL) for the modulating signal which determines the frequency of both the original signal which will be reflected and the matched signal will

guarantee both are in phase when they are correlated. However, the implementation of a PLL is beyond the scope of this project and given time constraints, an alternative method should be used. Splitting the noisy signal and matching it with two modulating signals 90 degrees out of phase or in quadrature(Q) phase will produce two different outputs, one of which will have a greater power for the peaks we wish to integrate. Combining both signals by summing the squares of both in phase and quadrature phase signals will produce a workable output. This is the most desirable method because it is not too difficult to implement and allows the system to run continuously which may be difficult with a PLL.

The correlating of signals can be accomplished using an analog multiplier which have relatively low output noise and can mix signals up to several gigahertz in frequency. The integration can be accomplished using a low noise Op Amp as shown in figure 3-14. The integrator can sum $V_{IN}$ until the output is grounded which occurs at the end of each sampling period. Threshold detection can be eliminated once the sampling frequency is established because the output of the integrator at the end of a sample will be large enough to discern from noise.

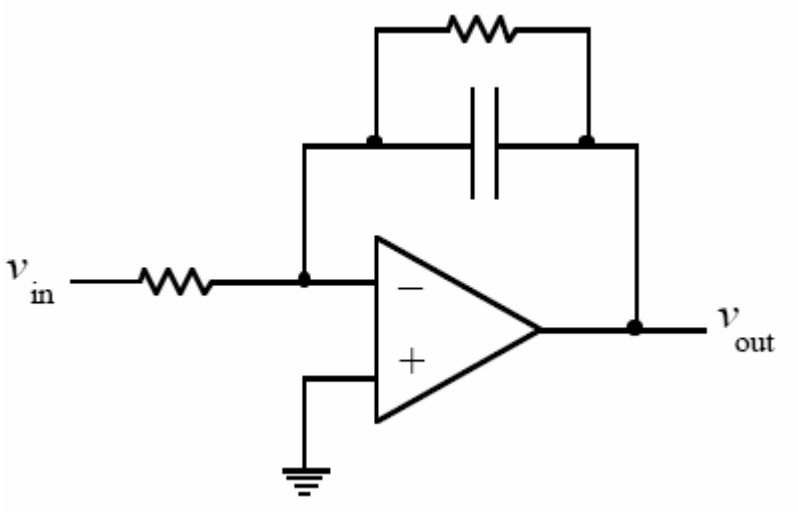


Figure 3-14 – Op Amp Integrator

### 3-2-1: Modulation Frequency Simulations

The matched filter is essential to our system because of its ability to maximize the SNR of the signal. In order to optimize the matched filter, we ran simulations that varied the modulation frequency and phase offset of the modulating and demodulating signals. To determine the best modulating frequency, we simulated the response of the matched filter in MATLAB and adjusted the frequency until we observed the largest increase in SNR from input to output. Figure 3-15 shows the output of the matched filter with an input signal with no added noise and an input signal with an SNR of –10dB.

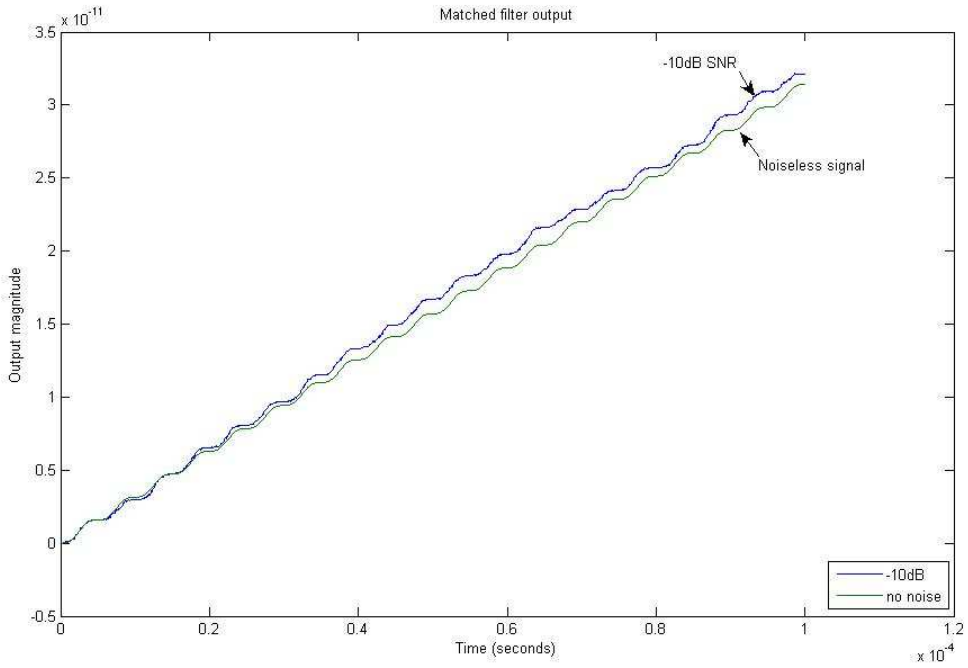


Figure 3-15 - Matched filter output

To determine the new SNR of the output signal, we plotted the distribution of output values with respect to the noiseless values. The new SNR value is then the variance of the outputs divided by the average value. To determine the optimum frequency we simulated with three frequencies: 100kHz, 500kHz, and 1MHz. We used an input signal with an SNR of –10dB and used our distribution plots to determine the output SNR with different modulation frequencies. The following figures show the distribution of the filter output for different frequencies.

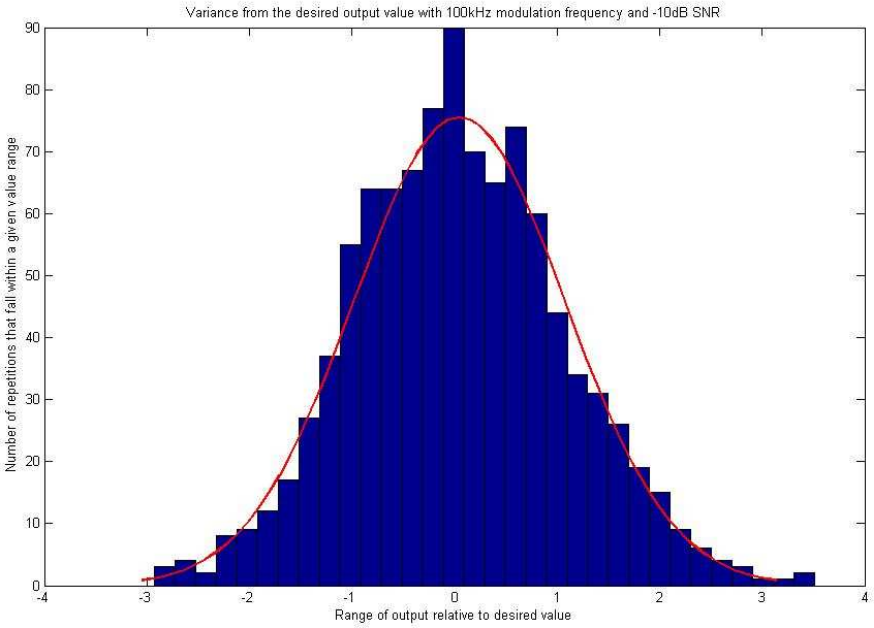


Figure 3-16 - Distribution for 100kHz

Noiseless output value = 31.41

Standard deviation ($\sigma$) = 1.0047

Input SNR = **-10dB**

Output SNR = Noiseless value / $\sigma$ = 31.41 / 1.0047 = 31.26 = **29.9dB**

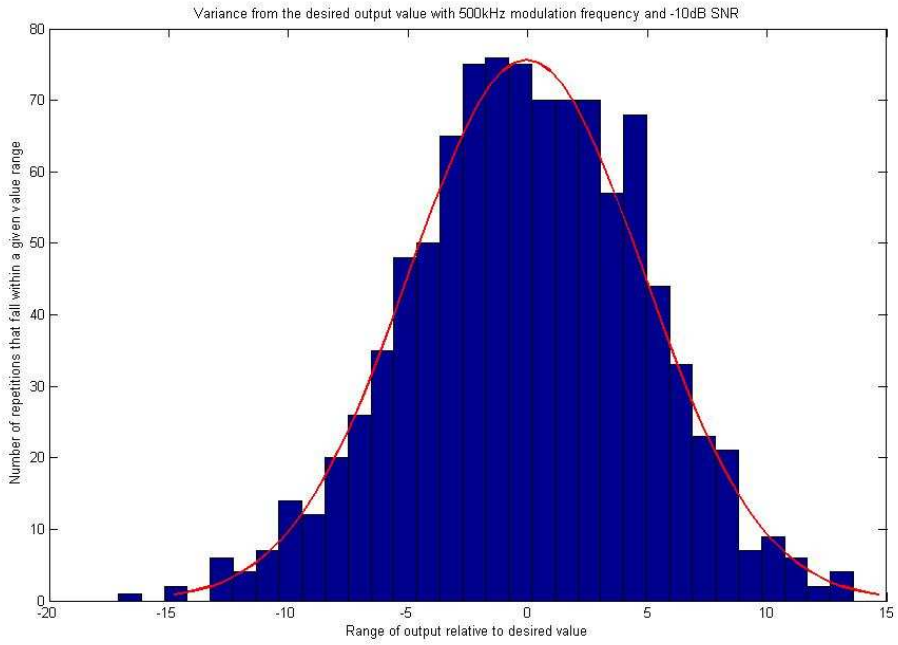


Figure 3-17 - Distribution for 500kHz

Noiseless output value = 157.06

Standard deviation (σ) = 4.8842

Input SNR = **-10dB**

Output SNR = Noiseless value / σ = 157.06 / 4.8842 = 32.16 = **30.1dB**

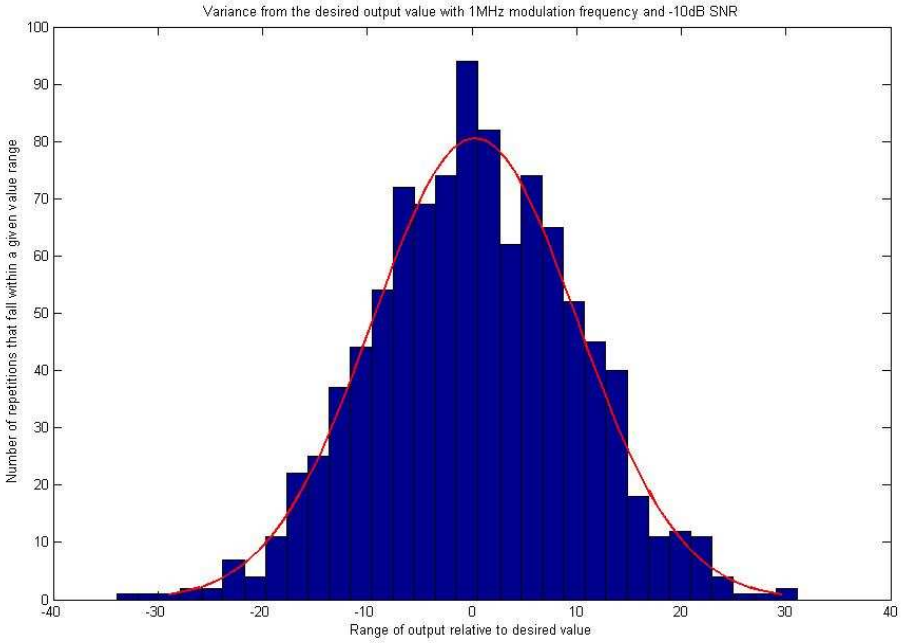


Figure 3-18 - Distribution for 1MHz

Noiseless output value = 314.13

Standard deviation (σ) = 10.3407

Input SNR = **-10dB**

Output SNR = Noiseless value / σ = 314.13 / 10.3407 = 30.38 = **29.7dB**

The previous plots show that the increase in SNR is relatively independent of the modulation frequency. This relieves a lot of constraints on the rest of the system. At first we believed that the highest modulation frequency we could handle would give us the best performance. The high frequency constraint made the transimpedance amplifier design difficult and noisy. The discovery that the modulation frequency does not have to be extremely high makes the amplifier design easier, allowing us to focus more on decreasing the total noise. This discovery is a much-needed "break" in the constraints we had previously placed on the amplifier.

### 3-2-2: Phase Offset Simulations

Another factor to consider while working with modulated waveforms is the phase offset that occurs naturally when the waves propagate across any length of space. The phase at the point of arrival will never be the same as at the point of departure. In order for the matched filter to function properly, the demodulating waveform must be in phase with the incoming waveform. Because the phase of the incoming waveform is unknown at the point at which it is demodulated in our system, we must demodulate twice with waves that are 90 degrees out of phase. When those two demodulated signals are combined, the output will resemble what we would have acquired with a demodulating signal in phase with the incoming signal.

This technique can be proven through simulation of the matched filter with different phase shifts at the demodulation stage. Figure 3-19 shows the audio signal as a step function in order to more clearly show the audio signal within the modulation scheme. The modulating signal is a sine wave with amplitude ranging from 0 to 1 which scales the amplitude of the audio signal. The time axis was intentionally left out because all figures are to equal scale.

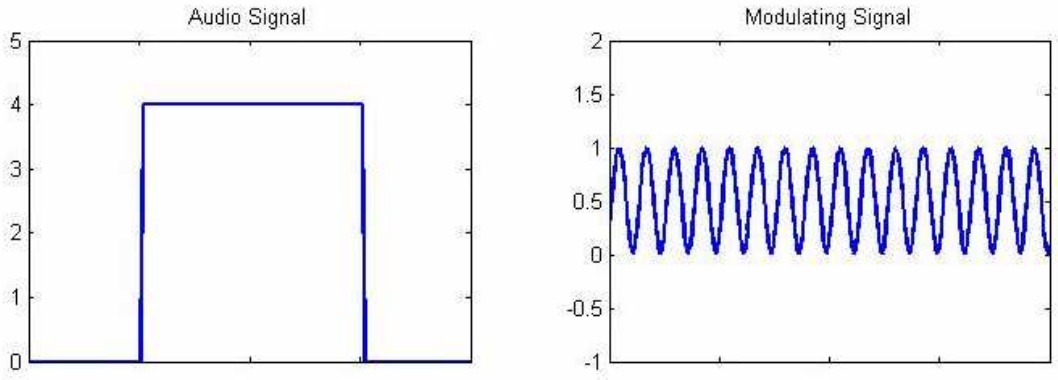


Figure 3-19 – The Audio Signal and Modulating Signal

The control for this experiment is the waveform demodulated with a signal that is in phase with original modulating signal. It is shown in figure 3-20. To the right of the demodulated signal is a graph of the integrator output over time as is sums the demodulated signal and is then reset. The demodulated signal graph also contains a number which is the total area underneath the demodulated waveform as a percentage of the area underneath the original audio signal. That percentage is for illustrative purposes only and during implementation we will apply a hold function which holds the respective peaks of the demodulated signal after integration. The sum of the signal after integration is given on the integrator output diagram.

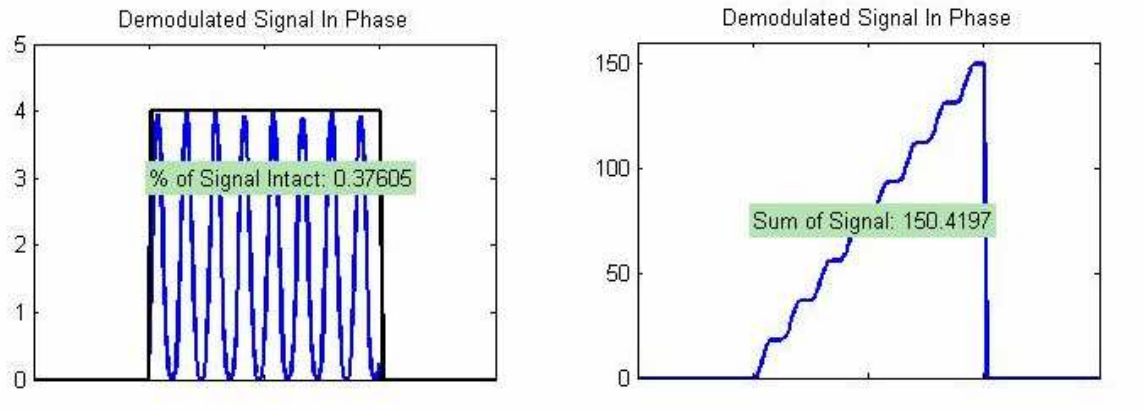


Figure 3-20 – Demodulated Audio Signal with In Phase Modulating Wave

To simulate a Quadrature demodulation scheme we first generate two sine waves that are 90 degrees out of phase with one another. Then the same phase offset is added to both waveforms. The next step is to demodulate the modulated audio signal with both waveforms and produce graphs of integration over time for both demodulated signals. Then the two demodulated signals are combined by squaring each signal, adding them together and taking the square root of the sum. The process is shown in figure 3-21 for a phase offset of 0.1 $\pi$ radians.

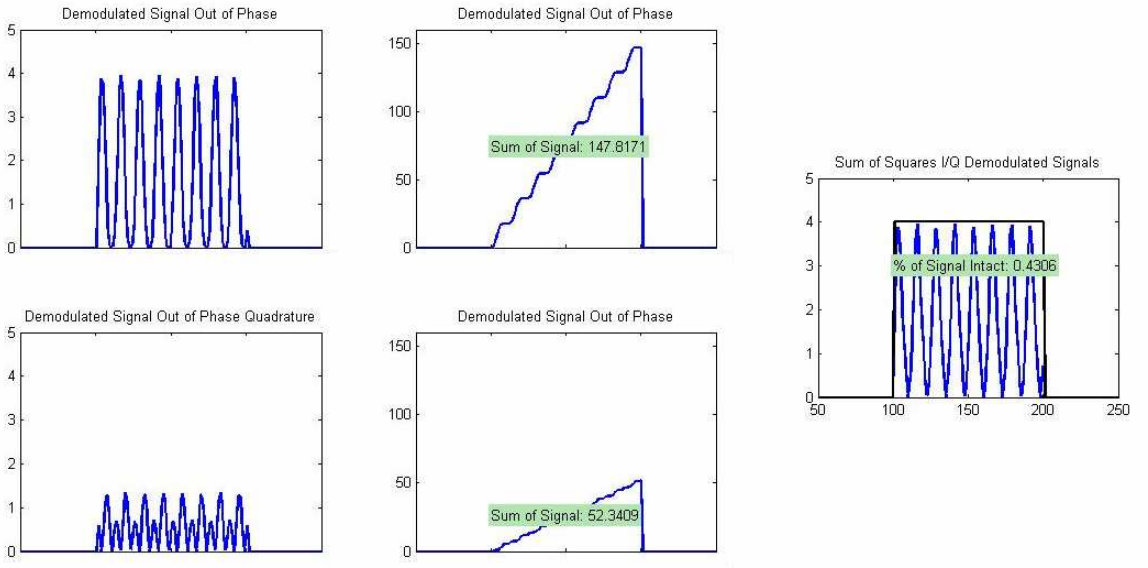


Figure 3-21 – Quadrature Demodulation Scheme with a Phase Off-set of 0.1π Radians

The small phase offset becomes apparent when observing the I or In-phase signal versus the Q or Quadrature-phase signal. The sum of the I-phase signal is 147.8 which is close to the sum of the in-phase signal of 150.4 in figure 3-20. The Q-phase signal only integrates to a value of 52.34 because when it is multiplied by the modulated audio signal it depresses the peaks. When both signals are added together, the percentage of the signal intact is 43% which is approximately equal to the in phase value of 37% from figure 3-20.

As the phase offset is increased, the Q-phase signal amplitude increases and the I-phase signal amplitude decreases. At a phase offset of 0.5 π radians both signals are nearly equal in amplitude which is shown in figure 3-22.

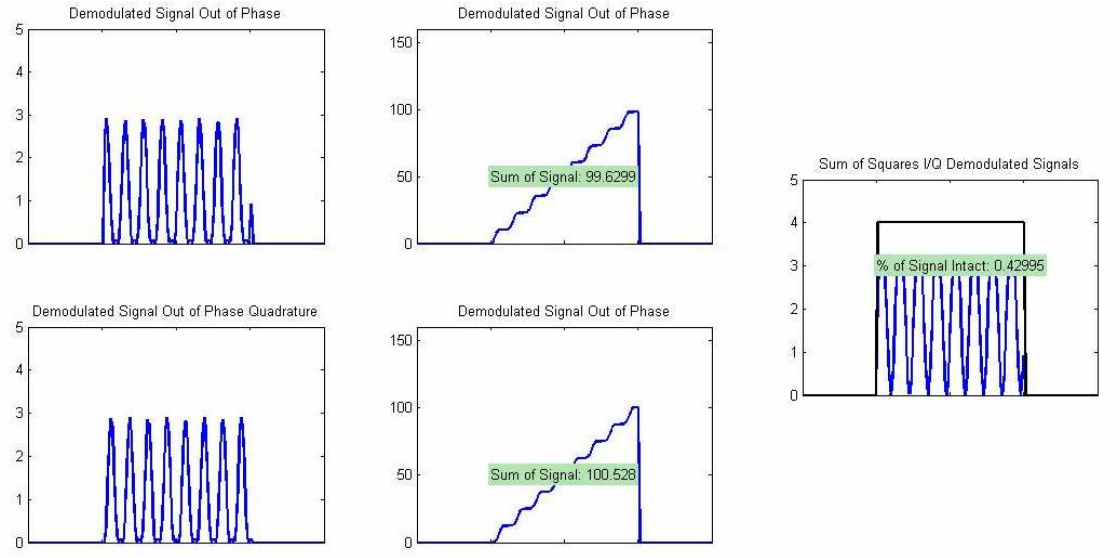


Figure 3-22 – Quadrature Demodulation Scheme with a Phase Off-set of 0.5π Radians

Increasing the phase offset all the way to 0.9 π radians is shown in figure 3-23. This is the opposite extreme case from the example shown in figure 3-21. The phase offset makes the modulated audio signal more closely resemble the Q-phase demodulating waveform then the I-phase demodulating waveform and as a result the output for the Q-phase signal integrates to a value nearly identical to the I-phase signal from the 0.1π phase offset simulation in figure 3-23. What is most important is that the sum of the signals remains fairly constant throughout the entire spread of phase changes given. Implementing a Quadrature demodulating scheme should keep the output constant even with changes in phase while the device is operating.

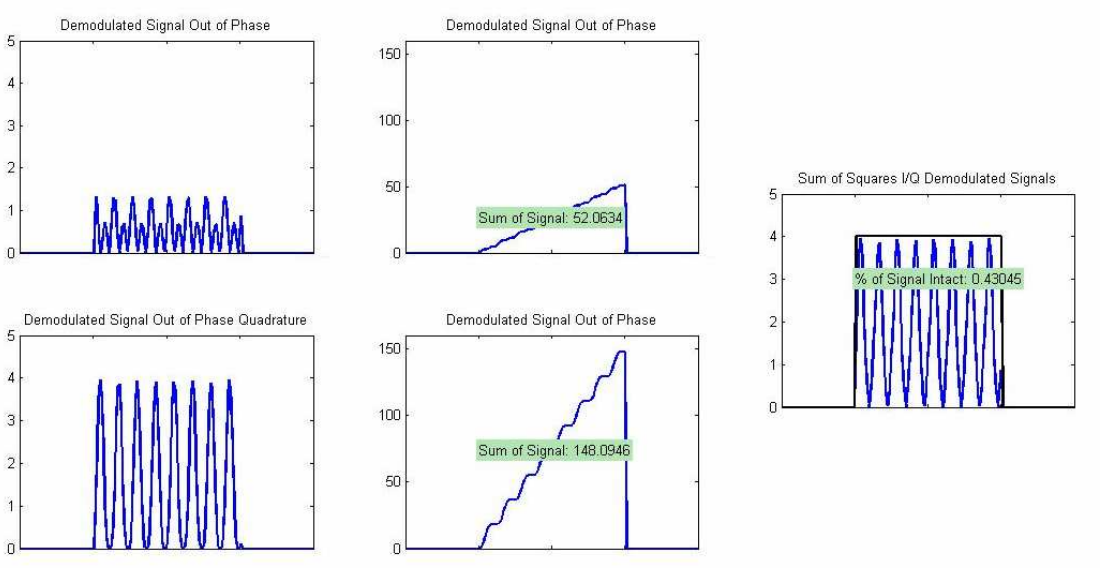


Figure 3-23 – Quadrature Demodulation Scheme with a Phase Off-set of 0.9π Radians

# 4: Implementation

## 4-1: Transimpedance Amplifier

As stated earlier in the design section, we decided on a cascode configuration for the transimpedance amplifier. The block diagram for the TIA stage is shown below in figure 4-1.


Figure 4-1 – TIA Block Diagram

The actual transimpedance amplifier design is shown next in figure 4-2.

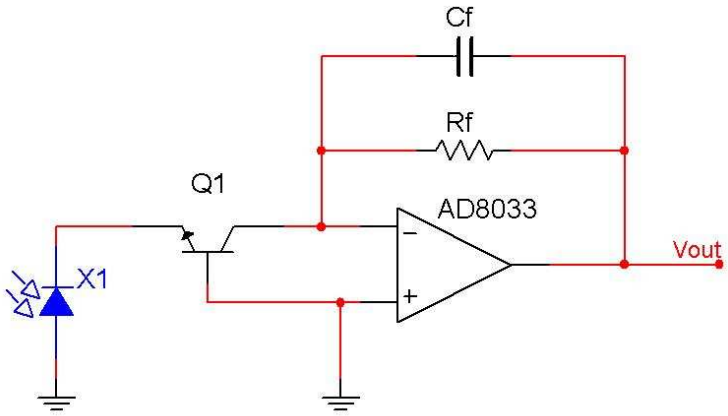


Figure 4-2 - Cascoded TIA

### 4-1-1: Op-amp Selection

The op-amp is the core of the TIA circuit, so it is the first component we researched. There are many op-amps to choose from, so we had to determine the most

important specifications for our application and narrow our search based on those properties.

Any noise that is added to the signal from the TIA stage will be amplified by successive stages and carried throughout the entire system. Thus the most important parameter we focused on was the noise characteristics of the op-amp. From the point of view of the op-amp, the voltage gain of the system is unity. Even if the value of $R_f$ is large, the input impedance of the op-amp is much larger. This means there is no voltage attenuation from the output to the op-amp input because the potential between the op amp and ground is maximized. Thus the voltage noise of the op-amp is not a noteworthy factor.

Instead, the current produced by the photodiode is the most significant contributor to noise in the op-amp. Thus we were able to narrow our search to op-amps with JFET inputs, which are known to have moderate voltage noise and very low current noise. After searching the Analog Devices website, we discovered the AD8033. The pinout is shown below in figure 4-3.

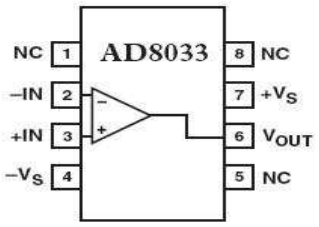


Figure 4-3 - AD8033 Pinout[1]

The AD8033 has an extremely low current noise parameter. The datasheet says:

$$I_{noise} = 0.6\,fA/\sqrt{Hz} \quad @\,100kHz$$

[1] Taken from Analog Devices AD8033 datasheet

The measurement for the datasheet was taken at exactly our proposed modulation frequency, so no conversion or estimation was needed. For this particular op-amp, the noise introduced by the op-amp to our input current signal is extremely small compared to similar op-amps. Though the matched filter will maximize the SNR in a later stage, it is still important to minimize the noise at the front end of the system.

The second most important factor in our op-amp choice was its bandwidth. Though our requirements were more relaxed then we originally thought, sufficient bandwidth is still essential to pass our 100kHz modulated signal to the rest of the system. The AD8033, along with its low current noise, has a bandwidth of 80MHz, which is more than enough to accommodate our signal. Finally, the input capacitance, which affects the bandwidth of the system, is only 2.3pF. Since it fit our system specifications so well, the AD8033 was chosen to be the component to implement the TIA.

### 4-1-2: Transistor Selection

The next component we decided upon was the cascode transistor. The transistor had to contribute as little noise as possible while "mirroring" the current from the emitter to the collector. The most important factor, though, was the collector-base capacitance. The cascode essentially shields the op-amp from the photodiode capacitance, replacing that capacitance with its own collector-base capacitance. To narrow our search, we researched only RF transistors, which are known for being low capacitance devices. After some searching we settled on the FMMTH10 RF transistor. It is designed specifically for common-base amplifier applications, so it has a very low collector-base capacitance. The maximum collector-base capacitance is $C_{CB} = 0.7pF$, which is extremely small.

### 4-1-3: Feedback Component Selection

For the TIA to function properly, the feedback components must be properly selected. The feedback governs the stability, bandwidth and amplification of our TIA output. The values were determined largely by experimentation, but for our starting point and for checking our values we used the equation to determine what the pole of the feedback network would be.

$$f_p = \frac{1}{2\pi(R)(C)}$$

This gives us a general estimation of the bandwidth of our TIA. The actual bandwidth will be higher, because the op-amp gain will be able to compensate for the additional attenuation of the feedback network. But as a rough estimate, the pole frequency will tell us what our minimum possible bandwidth would be.

Again, our values were determined mostly through trial and error and experimentation. Trying to be conservative, we reduced the gain to prevent instability. We decided upon the following resistance and capacitance values for our implementation:

$$R_f = 200k\Omega \qquad C_f = 1pF$$

In the calculation for the pole frequency, though, several circuit capacitances affect the actual frequency value. In the calculation, we have to account for the feedback capacitance, transistor collector-base capacitance, and the op-amp input capacitance. All these capacitances appear in parallel, so their values add together. This makes the overall pole frequency lower than if the feedback capacitance alone were responsible for generating the pole. Using our chosen values along with the capacitance value of the transistor and op-amp input we get a pole frequency of:

$$R_f = 200k\Omega \qquad C_f = 1pF$$
$$C_{op-amp} = 2.3pF \qquad C_{CB} = 0.7\,pF$$
$$f_p = \frac{1}{2\pi(R_f)(C_f + C_{op-amp} + C_{CB})} = \frac{1}{2\pi(200k\Omega)(1pF + 2.3\,pF + 0.7\,pF)} = 200kHz$$

Our goal modulation frequency is 100kHz, so our feedback attenuation will be negligible at our desired frequency. Knowing that our actual bandwidth will be higher than simply the pole bandwidth gives us a reassurance that our modulated signal will pass safely through the TIA.

When this circuit was implemented, we used a square wave input to the photodiode and measured the rise-time of the output signal. We were then able to use the rule-of-thumb equation:

$$(BW)(t_r) = 0.35$$

We measured a 10% to 90% rise time of about $t_r$ = 1uS. Solving the equation for bandwidth we get:

$$BW = \frac{0.35}{t_r} = \frac{0.35}{1\mu S} \approx 350kHz$$

Thus as expected, our actual bandwidth is higher than the pole frequency because of the op-amp gain compensating for the feedback attenuation. At the end of it all we have a stable TIA with moderate signal gain and more than enough bandwidth to accommodate our incoming signal.

### 4-1-4: High-Pass Filter

Our modulation signal will eventually drive the laser diode. Since our diode will not produce any light output if reverse biased, we will have to add a DC bias and our modulation signal will vary the intensity of the laser. This DC bias will produce a constant light output from the diode, which will be received as a DC offset in our TIA. There is no way to tell exactly what the offset value will be, so before we do any other processing of the signal the DC bias must be removed. Thankfully our modulation signal is relatively high, so a simple single-pole high-pass filter will work. Figure 4-4 below shows the high-pass filter we implemented.

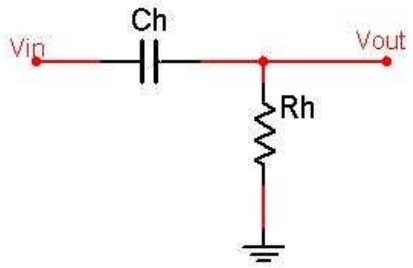


Figure 4-4 - High-pass Filter

Knowing the problems that capacitances can cause for op-amps, we were at first worried that the filter capacitor might cause instability with the op-amp. After redrawing the circuit to show the feedback, though, we were able to see that since the filter does not affect the feedback signal, it does not play a role in the stability of the TIA. As a background exercise, we modeled the high-pass filter circuit to show that it will act essentially as a short circuit at our modulation frequency. Figure 4-5 below shows the magnitude response of a high-pass filter with three different capacitance values and a resistor value of 10k ohms.

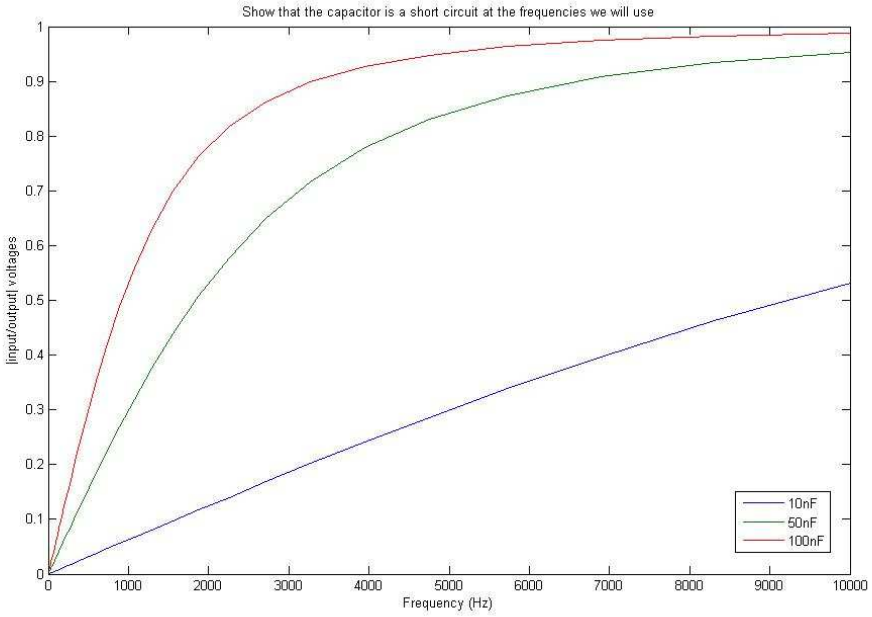


Figure 4-5 - High-pass Filter Magnitude Response

For our actual capacitance value, we used a 0.1uF capacitor, which is equivalent to 100nF. As can be seen from the magnitude response plot, the filter completely rejects the DC component, but at a frequency of 10kHz the gain is almost unity. If the graph were extended to include our 100kHz modulation frequency, the transfer function magnitude would be unity. Thus our high-pass filter will center our incoming signal near ground, allowing us to further process the signal in later stages.

## 4-1-5: Gain Stage

To take some of the amplification strain off of the TIA, we added a simple non-inverting gain stage directly after the TIA. We also added a potentiometer across one of the feedback resistors so we could have adjustable gain. The gain circuit is shown below in figure 4-6.

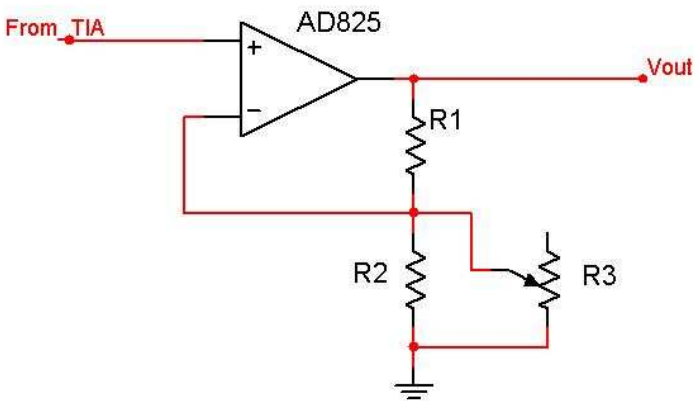


Figure 4-6 - TIA Gain Block

We used the AD825 for the gain stage because of its low voltage noise, high bandwidth, and low cost. By adding this gain stage with an adjustable gain we can effectively adjust the final amplitude of the output signal. This is important as the target reflection surface is moved further away from our system. Rather than risk instability by increasing the gain in the TIA itself, a simple non-inverting amplifier is very effective at increasing the signal amplitude without adding much additional noise.

## 4-1-6: Block Schematic

The schematic for our final implementation of the TIA block is shown below in figure 4-7.

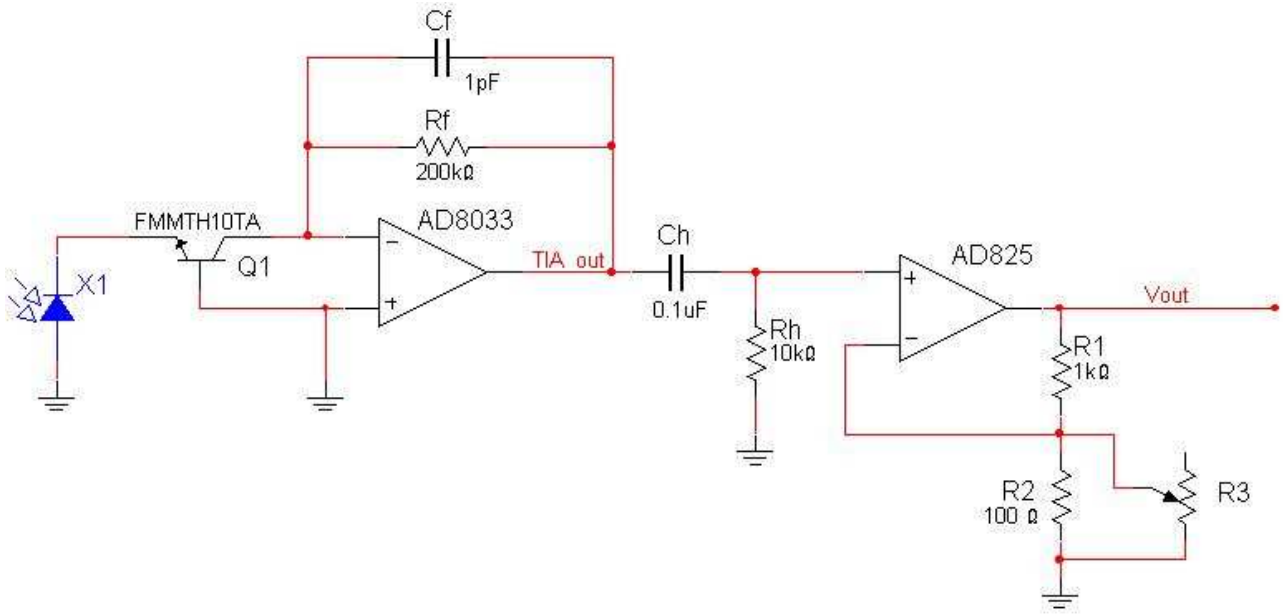


Figure 4-7 - Final Schematic for TIA Block

## *4-2: Matched Filter*

The goal of the matched filter stage is to maximize the SNR of the signal we received during the detection stage. The output of the matched filter at the time of sampling is essentially a sample of the input signal at that instant. The matched filter thus accomplishes two functions at once; increasing the SNR and converting the input signal to a pseudo-digital output.

The two main parts of our matched filter design are the multiplier and integrator stages. A basic block diagram of the matched filter portion of our system is shown below in figure 4-8.

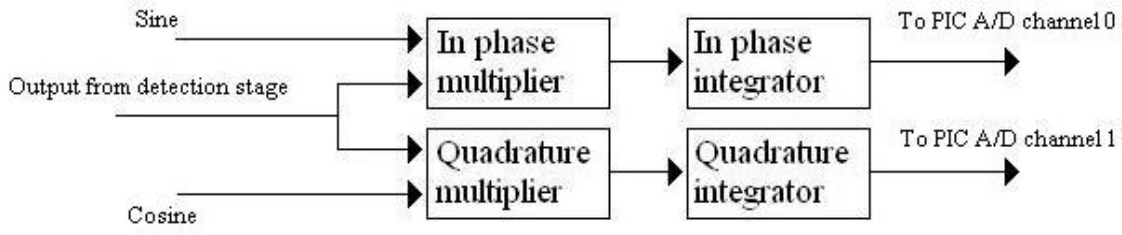


Figure 4-8 - Matched Filter Block Diagram

### 4-2-1: Multipliers

The filter requires two multipliers and two integrators to account for phase differences between the incoming carrier signal and the multiplying signals. These phase differences come from the propagation delay of our laser beam. Though light does travel quickly, it is not an instant propagation. A delay of only 2.5uS from time of transmission to receiving the signal would result in a phase shift in the carrier of 90 degrees. If we

used only a sine wave reference, for example, and the incoming carrier signal was 90 degrees out of phase, the multiplication would result in an output of zero.

To compensate for the possibility of a zero output, we also multiply the incoming signal with a cosine, which is 90 degrees out of phase with a sine wave. Using two demodulating signals, each 90 degrees out of phase, ensures that there will never be a zero output from the multipliers. The down side is that there will ultimately be two outputs that we must combine together to obtain the original input signal. The combination of the two output signals is done within the PIC processor using the following algorithm:

$$V_{out} = \sqrt{V_{\text{int}1}{}^2 + V_{\text{int}2}{}^2}$$

Where $V_{out}$ is the actual desired sample, and $V_{int1}$ and $V_{int2}$ are the outputs from the two integrators.

Unfortunately we were not able to make it to the point of using pure sine and cosine inputs for the laser modulator and multipliers. Instead we made use of the capabilities of the PIC to continually generate PWM outputs. Though they were not filtered to pure sine waves, the square wave outputs from the PIC were out of phase by exactly 90 degrees, something that would have been difficult in the analog world.

## 4-2-2: Integrators

The integration part of the matched filter is essential to achieve the maximum SNR at the output. The time constant must be chosen to allow for the maximum possible voltage change across the integration time. The basic integrator we started with is shown in figure 4-9 below.

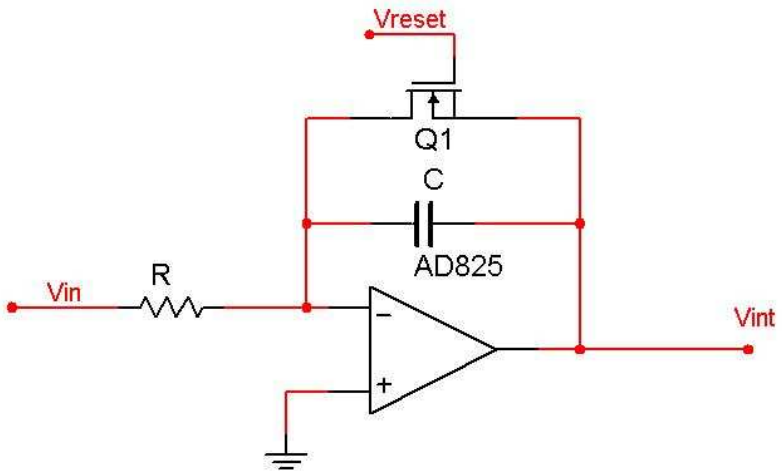


Figure 4-9 - Basic Op-amp Integrator

The multiplication of the input signal by the demodulating signal reinforces the input and diminishes the noise magnitude. The frequency of the noise within the signal is different from the multiplication signal frequency, so the noise does not receive the squaring effect that the signal does. The signal power is thus increased, while the noise power is decreased, and the random noise voltage remains centered at zero. The integrator then acts as an averaging circuit. The reinforced signal causes the integrator output to move away from zero, where the ground-centered noise is averaged to almost zero, causing little change in the output.

Our original intent was to be subject only to the constraints of the power supply rails, and use the complete +/- 5V range. If the integrators have a large range, then the final signal output can be very large compared to the low noise floor, making for a much larger SNR. Thus the maximum possible input voltage should integrate to +5V over our integration time. Doing some simple circuit analysis of the integrator yields the following equation, where the values of R and C can be extracted:

$$V_{in(\max)} = RC\frac{dV}{dt}$$

Knowing that the output voltage from the multipliers will be small, we assumed a maximum input voltage of 1V. We also chose a value of C = 0.01uF, which is large enough for us to neglect integration offset problems, but small enough to prevent the op-amp from hitting its output current limit. Knowing that our integration time is 100uS and our maximum desired output voltage is 5V, we can solve for the resistance:

$$1V = R(0.01\mu F)\left(\frac{5V}{100\mu S}\right)$$

$$R = 2k\Omega$$

While this approach was legitimate, and it functioned exactly as planned, we could not use it in our application. After some datasheet digging, we found that the A/D converter we planned to use on the PIC could not convert negative voltages. We thus had to devise a way to reduce the amplitude of the output signal, and add a DC offset so that the entire signal was above ground. By changing the resistance and modifying the integrator slightly, as shown in figure 4-10, we were able to add a 2.5V DC offset, using that as our center point for the integrator output.

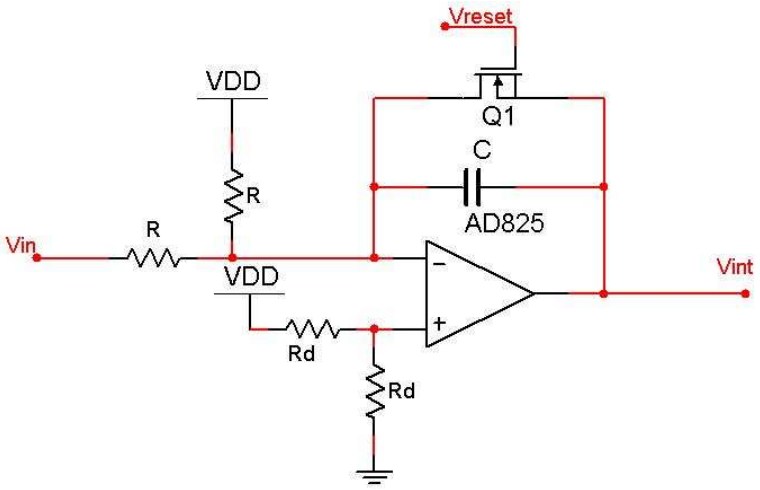


Figure 4-10 - Offset Integrator

Using a voltage divider, we set the voltage at the non-inverting terminal of the op-amp to 2.5V. Then, when the integrator is reset, the output will force the inverting terminal to be equal to the non-inverting terminal, thus setting the output to our 2.5V starting point. Knowing that the non-inverting terminal will also be constantly set at 2.5V, we added a resistor from the non-inverting terminal to the power rail as compensation. Now, when the input voltage is zero, current will flow through the two R resistors with no current flowing into the capacitor. Thus an input voltage of zero will cause no change in the output voltage, which is exactly what we want.

Once again doing the circuit analysis, we get the following equation where we can solve for the resistance R:

$$\frac{dV}{dt} = \frac{V_{in(\max)}}{RC}$$

Knowing that the maximum input voltage will be 1V, the maximum voltage change is 2.5V, the capacitor value is 0.01uF and the integration time is 100uS, we can solve for the resistance as follows:

$$\frac{2.5V}{100\mu S} = \frac{1V}{R(0.01\mu F)}$$
$$R = 4k\Omega$$

Unfortunately, due to time constrains on the processing side, we had to increase our integration time to 250uS. The increase in integration time will increase the SNR, but greatly reduces the usable bandwidth of the system. Using this new integration time, we finally settled on a resistance value of:

$$\frac{2.5V}{250\mu S} = \frac{1V}{R(0.01\mu F)}$$
$$R = 10k\Omega$$

## 4-2-3: Components

### *4-2-3-1: Multiplier IC*

Though we did not meet our goal, our original intent was to use pure sine waves for the carrier and demodulating waveforms. Thus we needed an analog multiplier that was fast enough to multiply our 100kHz carrier and demodulating waves. The best device we found that fit our primary constraints was the AD835. The functional block diagram is shown below in figure 4-11.

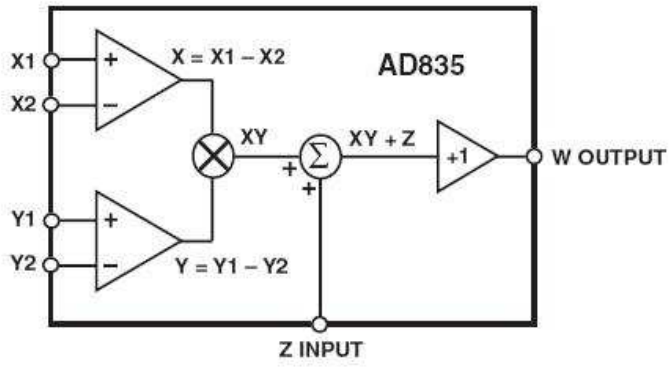


Figure 4-11 - AD835 Functional Block Diagram[2]

The AD835 is a fully functional four-quadrant analog multiplier. It has a bandwidth of 250MHz, which is more than enough to handle the 200kHz output frequency that results from the multiplication of our signals. The AD835 has a transfer function of:

$$W = (X_1 - X_2)(Y_1 - Y_2) + Z$$

Thus to use this multiplier in our application, we tied $X_2$, $Y_2$, and Z to ground, or zero volts. For our application the transfer function simplifies to:

$$W = (X_1)(Y_1)$$

This is exactly the functionality we desire, where $X_1$ is our modulated input signal and $Y_1$ is our demodulating signal.

The biggest downside to this particular multiplier is the voltage ranges of the input and output. The input range is limited to +/- 1V and the output range is limited to +/- 2.5V. The original circuit implementation assumed a close to full-scale range of the input and output, so adjustments needed to be made. To ensure the range was not exceeded, the demodulating waves had to be put through a voltage divider, and the input

[2] Picture taken from Analog Devices AD835 datasheet

waveform had to be checked to ensure it was within the range. The integrator RC time constant also had to be adjusted to account for the smaller than expected output range of the multiplier.

### *4-2-3-2: Integrator Op-Amps*

The integrator op-amp had to meet two main specifications: sufficient bandwidth and very low input current. For this component, we chose the AD825, with the pinout shown in figure 4-12 below.

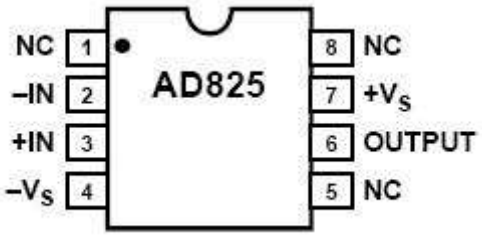


Figure 4-12 - AD825 Integrator Op-amp[3]

With a bandwidth of 41MHz, an input current of only 20pA and a low cost, this op-amp fit our specifications perfectly. The low input current is the most important factor for an integrator op-amp. The integration is a result of current flowing into the feedback capacitor. If the input current to the op-amp is too high, some of the signal current will be diverted away from the capacitor and into the op-amp, causing offsets or other integration errors. The input current of 20pA of the AD825 is extremely small compared to the signal current, so its effect is negligible.

[3] Picture taken from Analog Devices AD825 datasheet

## 4-2-4: Block Schematic

The schematic for our matched filter implementation is shown below in figure 4-13. It is only half of the total implementation, but the second half is identical except for the demodulating signal.

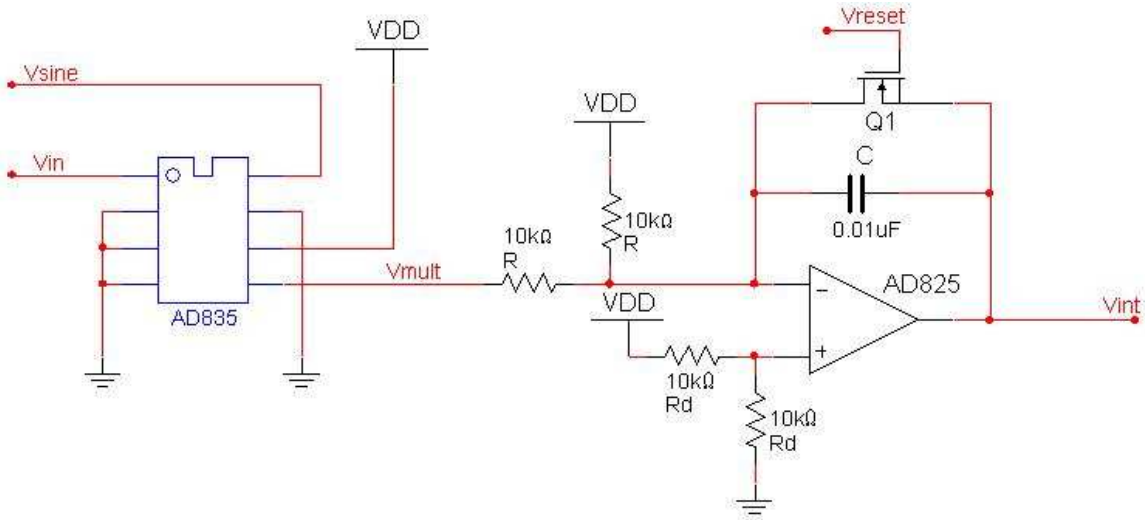


Figure 4-13 - Matched Filter Half Schematic

## 4-3: PIC Implimentation

The heart of our circuit controls is the dsPIC30F3013 PIC microcontroller. It is responsible for all timing, control signals and integrator output sampling. We chose the 30F3013 for several reasons. First of all, I was most familiar with the operation and implementation of PIC microcontrollers and already had a programmer for this particular chip. Second, we were able to find a free C compiler for this model microcontroller so we could use a higher level programming language rather than assembly. Finally, when compared to other microprocessors we looked at, the 30F3013 had a relatively small pin count and package size which would make it the best choice for our compact PCB.

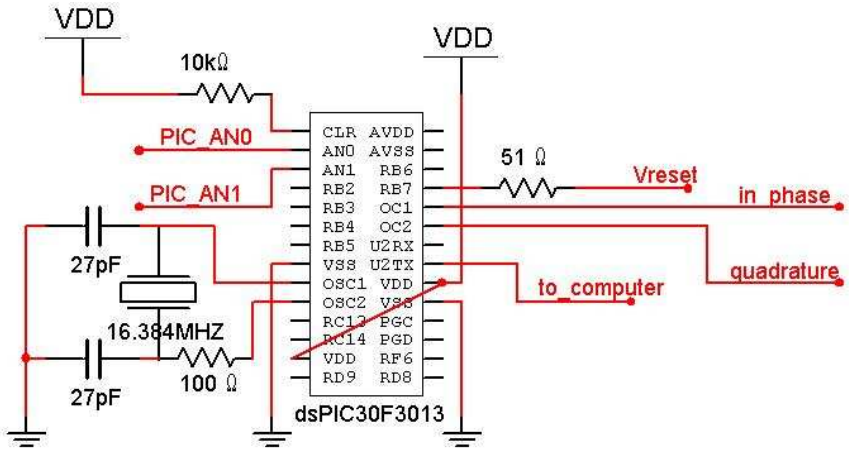


Figure 4-14 - PIC implementation circuit

Figure 4-14 above shows the circuit we used for using our PIC microcontroller. It is a straightforward circuit with relatively few components because the majority of the work is done internally by the program. For our test circuit we added a push button to tell the PIC to begin transmitting data to the computer, but that was omitted from the circuit shown because all transmission will be done in real time in the final design.

As was mentioned previously, the PIC is responsible for the timing of our entire system. Our original goal was to allow for an audio bandwidth of 4kHz, and in order to satisfy the Nyquist criteria we would have to sample at a minimum of 8kHz. To give ourselves some headroom, we decided to sample at 10kHz. Thus our original sampling period goal was:

$$F_{sample} = \frac{1}{T_{sample}}$$
$$10kHz = \frac{1}{T_{sample}}$$
$$T_{sample} = 100\mu S$$

Initially we wanted to sample the outputs of the integrators every 100uS. But we soon ran into a bottleneck with our PIC to PC communication link. Our A/D converter generates a 12 bit value that must be transferred to the computer. Because of the RS232 standard we were using, we could only transfer 8 bits of data at a time plus the 3 control bits. On top of that, the serial port on the computer could only support data transfer rates up to 115.2kbits/second, meaning it would take approximately 8.5uS to transfer each bit. Thus the time to transfer the data for just one sample would be:

$$T_{transfer} = 2\left(\text{sec}\,onds/_{bit}\right)(bits\ \ per\ transfer)$$
$$T_{transfer} = 2(8.5\mu S)(8_{data} + 3_{control}) \cong 190\mu S$$

Our transfer thus takes about 190uS, which is much longer than our sampling time of 100uS. If we continued to sample every 100uS, data would get "backlogged" because we could not transfer information out as quickly as it comes in. Also, along with the transfer time, we had to account for conversion time for the A/D which turned out to be about 11uS. So ultimately our sample time would have to be greater than 200uS to account for all the delays within the PIC.

To take care of any other delays we may have overlooked in our previous calculations, we increased the sampling time to 250uS. Unfortunately, this reduced our audio bandwidth available to a maximum of only **2kHz**, which is half of our desired bandwidth. This could be solved in the future by choosing a different communication protocol, but we did not have enough time to work around this constraint.

### 4-3-1: Demodulating Signals

As discussed in the matched filter sections, we need to demodulate our incoming signal with both an in phase and a quadrature phase signal of the same frequency. Originally we were going to simply use a function generator to provide the in phase signal and use an all-pass filter to generate a 90 degree phase shift to that signal. For a phase shift of exactly 90 degrees, though, the components on the all-pass filter would have to be extremely accurate.

Once we got familiar with the PIC, though, we realized that it could output two PWM signals at 100kHz that were exactly 90 degrees out of phase. The resulting output is shown in figure 4-15 below.

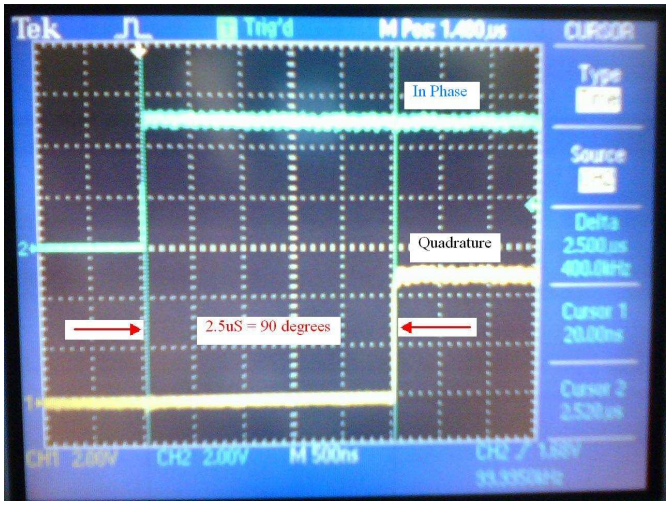


Figure 4-15 - In phase and Quadrature demodulation signals

By delaying the start up time for each output by 2.5uS, we were able to get two square waves that are exactly 90 degrees out of phase. In the future if we wanted to demodulate with a sine wave, we would simply need to low-pass filter each PWM output.

## 4-3-2: Sampling and Reset Signals

Perhaps the most important role of the PIC is to sample the output of our matched filter. The model that we chose has an internal A/D converter, but only one sample and hold channel. We thus have to convert one channel first, then quickly sample and convert the other. Ideally we would sample both channels simultaneously, but the lack of a second sample and hold channel makes that impossible. Thankfully, a small delay in sampling should not have a big effect on the final output results.

Every 250uS the PIC samples each channel, and then resets the integrators to their starting value. One of the challenges we encountered earlier was designing an integrator that has an output centered on 2.5V to accommodate the PIC A/D range. Figure 4-16 below shows the reset timing generated by the PIC.

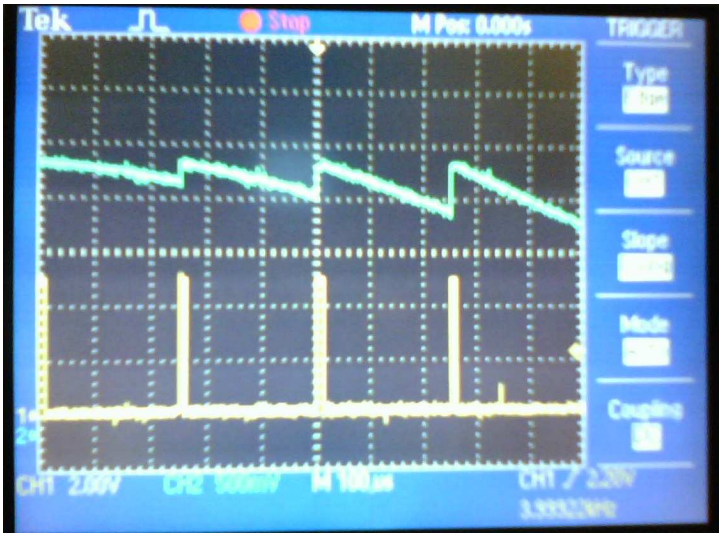


Figure 4-16 - PIC Reset Timing

In figure 4-16, the blue waveform on the top is the output from the integrators and the yellow pulses are the reset signals from the PIC. The reset signal has a frequency of 4kHz, which corresponds to our desired integration period of 250uS.

### 4-3-3: MATLAB Output Plots

Once the sampling and reset procedure was functioning properly, we began to transfer that information to the computer. We wrote a MATLAB script to read values from the serial port and then to plot those values. We also plotted the FFT of the signals to see if we were getting the correct frequency content. Figure 4-17 below shows the output of our system when a modulated 300Hz sine wave is incident on our photodiode.

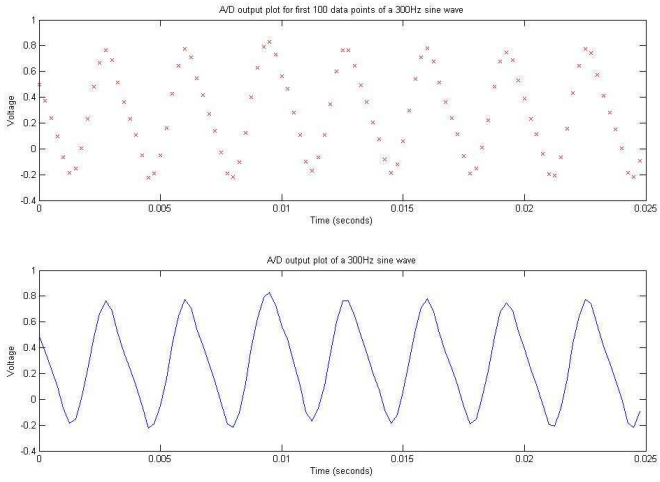


Figure 4-17 - Plot of 300Hz Sine Wave Output

The sine wave shown above is quite distorted. This is because we did not have time to properly implement the algorithm to combine the two integrator outputs. Thus the values transferred to MATLAB are the samples of only one output individually. The output is not a clean sine wave because of frequency and phase differences between the modulating and demodulating waveforms. The distortion of the sine wave can also be seen in the frequency domain. Figure 4-18 below shows the FFT of the waveform above.

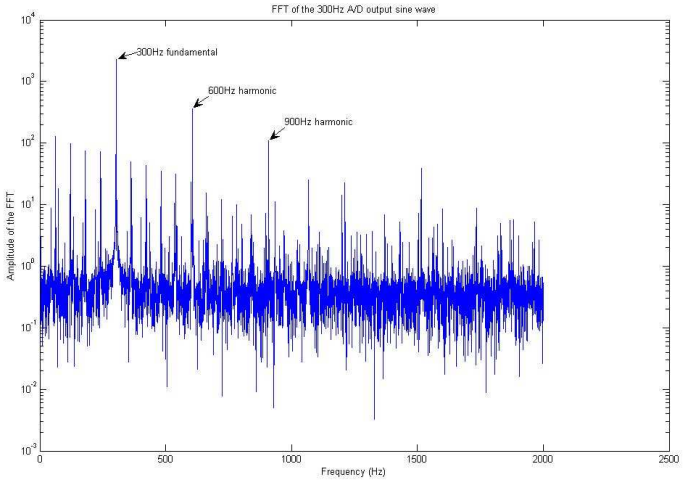


Figure 4-18 - FFT of 300Hz Sine Wave

The frequency content of the sine wave does show our 300Hz signal as being the dominant component, but it also has large harmonics and noise content. These unwanted frequencies are added by both the phase difference between modulating and demodulating waveforms and the inherent noise of the A/D converter.

The final code for both the MATLAB and the PIC script can be found in Appendix B and Appendix C at the end of this report.

# 5: PCB Testing

One of the shortfalls of the previous projects was that they used a breadboard for building and testing their circuits. Unfortunately for them, the breadboard leads acted like antennas and coupled noise into their circuit. To avoid this, we decided to go directly to a printed circuit board. We used a program called Eagle PCB to design the board shown in figure 5-1 below.

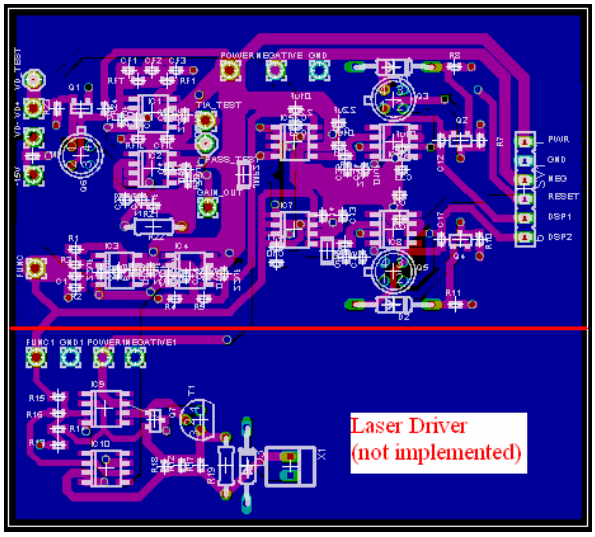


Figure 5-1 - PCB Layout

We decided to use surface mount components and short traces to limit the amount of noise that is added to our system from external sources. Once we received the PCB we tested each sub circuit individually and the results are outlined in the following sections.

## *5-1: TIA Testing*

For our test plan, we decided to simply follow the signal through the circuit by testing each block in order. We first started with the transimpedance amplifier. To test the TIA block, we simply hooked up a white LED to a function generator and aimed the LED at our photodiode. Our first output is shown below in figure 5-2.

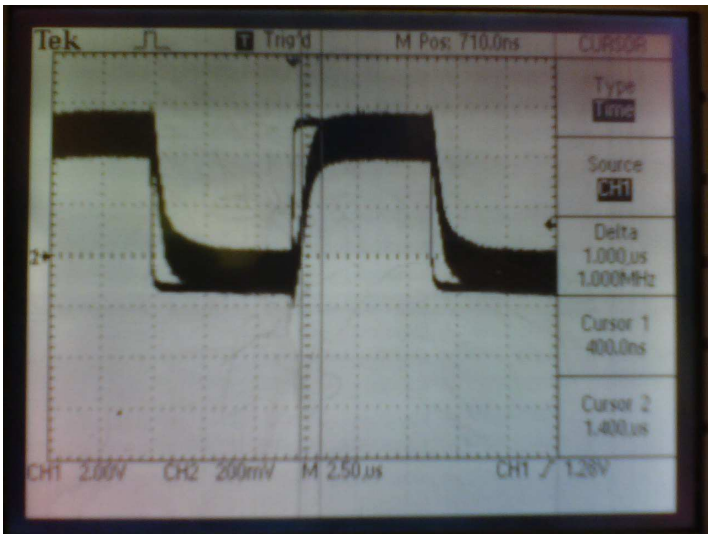


Figure 5-2 - TIA Test Output with Oscillations

When we first tested the TIA, we saw that the output was very "noisy". When we zoomed in on the noise, though, we saw that it was an oscillation at about 7MHz rather than noise. After some further testing we realized that we had forgotten the supply rail bypass capacitors. The inductance in the power lines was causing our op-amp to oscillate. Once the bypass capacitors were added, our output was very clean, as shown in figure 5-3 below. The square wave on the top is the waveform that we used to modulate our LED and the bottom waveform is the output of our TIA.

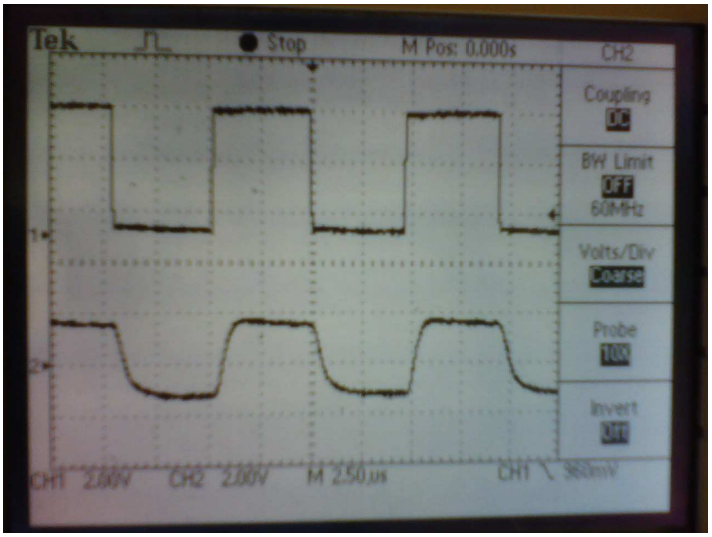


Figure 5-3 - TIA output with bypass capacitors

By using a square wave we were also able to get an estimate of our bandwidth by using the relationship between the bandwidth and rise time. To estimate the bandwidth we measured the rise time to be 1uS and used the following relationship:

$$BW \times t_r = 0.35$$

$$BW = \frac{0.35}{1\mu S} = 350kHz$$

The modulation frequency we are using is only 100kHz, so our actual bandwidth gives us plenty of excess bandwidth to accommodate our modulation signal. Initially we wanted to use a sine wave modulator, but even if we use a square wave the excess bandwidth will allow the square wave to pass relatively undistorted.

## *5-2: Multiplier Testing*

The AD835 multiplier chips are the most sensitive components in our circuit. They have a limited input voltage range of +/- 1V, and an output range of only +/- 2.5V. To test that they were working properly, we put a pure sine wave into each of the input terminals.

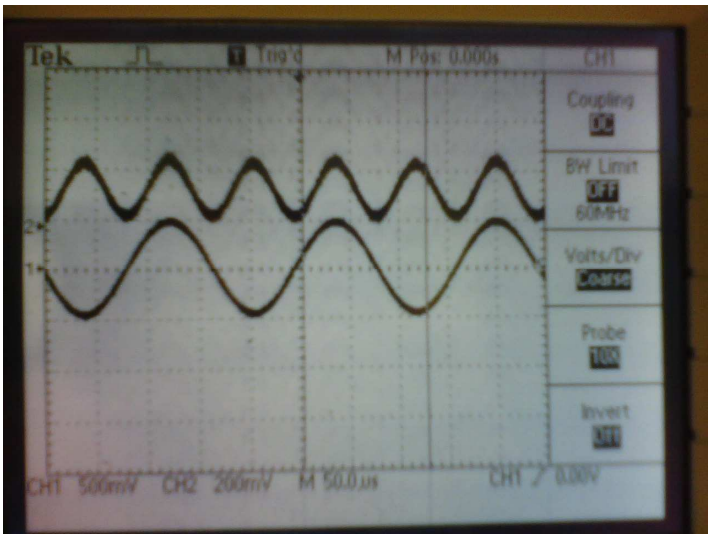


Figure 5-4 - Sine Wave Multiplication

Figure 5-4 above shows how we tested each multiplier circuit. At each input we connected a sine wave and at the output we observed a squared sine wave. The output is exactly what we expected, so the multipliers were working properly.

## *5-3: Integrator Testing*

The final portion of our PCB was the integrator portion of the matched filter. Without the PIC working as a reset circuit, we had to use the function generator to activate the reset switches. Figure 5-5 below shows the modulation waveform and the integrator output.

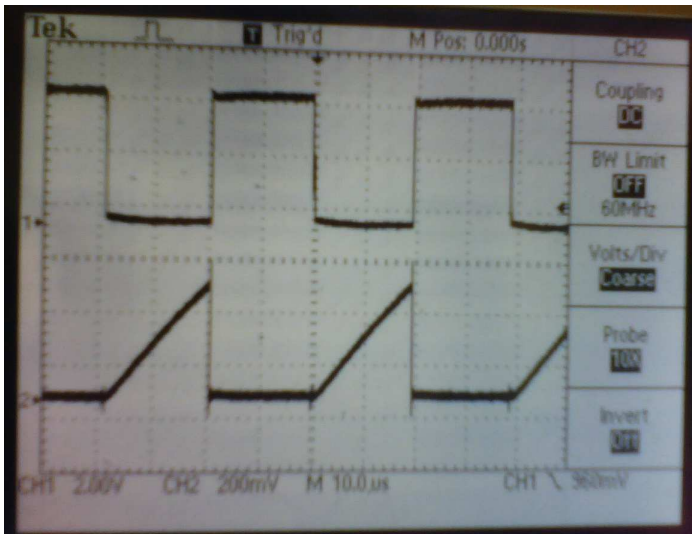


Figure 5-5 - Integrator Outputs

The top waveform is the square wave that we used to modulate the LED. The bottom triangle waveform is the output of the integrators. The rising edge of the modulation wave was used to activate the reset switches. The integrators work exactly as expected with a square wave input. This also shows that the entire PCB works from end to end, as the square wave is used to modulate the LED and the integrator output is taken from the final part of the matched filter.

## *5-4: PIC Testing*

After all the individual blocks were working properly, we began programming the PIC for the timing functions. Once the PIC was programmed and connected properly, the system began to function properly in regard to the analog portion of our system.

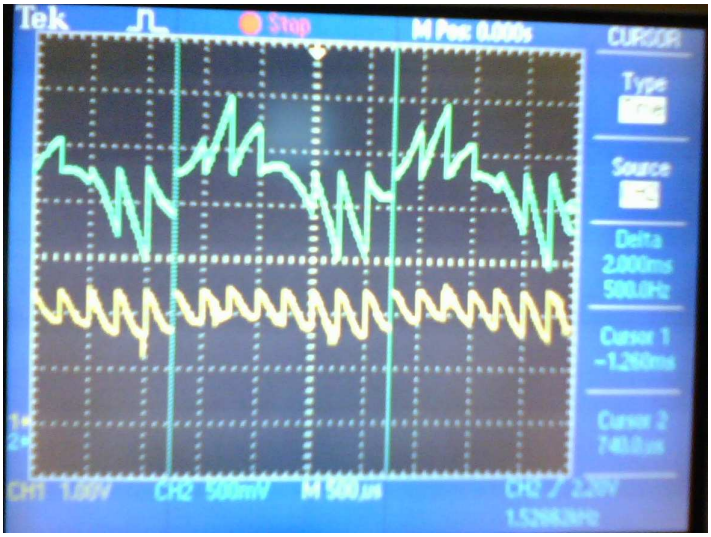


Figure 5-6 - Full System Output

To test the final system, we modulated a 500Hz sine wave with our 100kHz carrier wave. We demodulated it with our in phase and quadrature phase signals and observed the output of the integrators. The integrators were reset every 250uS according to the signals from the PIC. The outline of a sine wave is clearly seen in figure 5-6 above, with a frequency of 500Hz. Once the PIC was added, individual blocks began functioning as a single system.

# 6: Conclusion

Although we made a great deal of progress towards designing a working laser audio surveillance device, there were a number of restrictions in our system that prevented us from achieving that goal. The greatest constraint in our system is that the system bandwidth was not large enough to produce an audio signal. The choke point within our design that restricts our bandwidth the most is the PIC RS232 microprocessor. This is a functional limitation of the chip itself, specifically the data rate on the analog to digital converter is too slow to convert our integrator outputs to a digital audio signal for further processing. Given that the PIC is the final subsystem we worked on, a longer project cycle would have given us the opportunity to choose a more suitable PIC for signal processing.

Another limitation to our project is that we were unable to implement a laser diode driver and used an LED for testing. Although the LED is more flexible within the framework of our system, a diode laser would have provided a more stable and coherent source of light that would greatly improve performance at the photodiode and TIA.

The remaining subsystems functioned as expected. The TIA needed only two modifications beyond its basic design to function properly with our desired bandwidth. The first was the addition of a FMMTH10 RF transistor to reduce the effect of the photodiode capacitance and increase our system bandwidth. The second was the addition of an adjustable gain stage immediately following the TIA which added a much needed flexibility to its output as well as reducing strain on the TIA AD8033 op amp.

The matched filter underwent a number of changes throughout the project cycle. Due to the addition of the PIC RS232, both the modulating and the I and Q demodulating

signals could be produced by the same source and streamlined that part of the design. The multiplier and integrator were designed with simplicity in mind and did not present any real problems during system testing. The first modification to the matched filter was due to the fact that the AD835 analog multiplier could not accept an input beyond +/- 1V and the output from the TIA as well as the PIC modulator needed to be scaled. Another modification to the integrator stage was due to the fact that the PIC could not receive an input less then 0 V and the integrator stage needed a positive voltage offset to function properly. The matched filter functioned as expected and fell within our desired system bandwidth.

There are a few recommendations we can make to future project groups that may work on this or similar designs. The first would be to spend a significant amount of time investigating different microprocessors for digital signal processing. We implemented a PIC that we were most familiar with but its input conversion limitations were the final constraint in an otherwise functional system.

Another recommendation we can make is to use prepackaged ICs for signal multiplication, driving the laser diode and any other subsystems were noise is a concern. The primary reason we did not implement a laser diode driver is that the one we designed overloaded the laser diode and did not produce a stable output. Additionally, designing an analog multiplication circuit using op amps would have produced a large amount of noise and would not function as effectively as a predesigned module. ICs are relatively easy to select if you know your system's performance expectations, are inexpensive and often available at no cost if you request samples and take up much less circuit space which is critical for reducing the circuit loop to reduce interference.

Although our overall system did not function well enough to produce an audio output, most of our subsystems functioned as expected for our desired performance. This project taught us a great deal about light based communication systems, reducing interference in electrical circuits and different techniques for signal processing and increasing a circuit's bandwidth. As a result, this project was a successful capstone design experience for our undergraduate career at the Worcester Polytechnic Institution.

# Appendix A: Works Cited

[1] "Vocal Folds" *Wikipedia*, Available at HTTP:
http://en.wikipedia.org/wiki/Vocal_chords

[2] “Parabolic Microphone” *Wikipedia*, Available at HTTP:
http://en.wikipedia.org/wiki/Parabolic_microphone

[3] “Audio Frequencies” *Wikipedia*, Available at HTTP:
http://en.wikipedia.org/wiki/Audio_frequency

[4] “Microphones” *Wikipedia*, Avaliable at HTTP:
http://en.wikipedia.org/wiki/Microphone

[5] Ott, H. W., Noise Reduction Techniques in Electronics Systems, AT&T Bell Laboratories, 1988.

Siegman, A. E. , Lasers, University Science Books, 1986.

Hobbs, P. C. , Building Electro-Optical Systems Making It All Work, John Wiley and Sons Inc, 2000.

# Appendix B: Matched Filter Phase-Offset Simulation

```
% Kevin Manelski
% 2/20/07

% Start by creating a psudosoundwave at the TIA input (all positive
values)
% In this case it is a soundwave created by random number generation
% With potential voltage values between 0 and 4.25

% Maximum speach frequency 1 kHz, Modulation Frequency 100 kHz
% 100 Samples per Hz

phOFF = .9;

dt = 0.5;       % Step size in all waveforms (less than or equal to 1)
                % Lower values decrease the carrier frequency
%a = rand(1, 10).*4.25;
a = zeros(1,10);
a(1) = 4;

% Create an array that creates a smooth curve with the 10 random values

b = zeros(1, 2000/dt+1);
c = zeros(1, 2000/dt);
d = zeros(1, 2000/dt);
e = zeros(1, 2000/dt);
int_e = zeros(1, 2000/dt);
f = zeros(1, 2000/dt);
g = zeros(1, 2000/dt);
int_g = zeros(1, 2000/dt);
h = zeros(1, 2000/dt);
k = zeros(1, 2000/dt);
int_k = zeros(1, 2000/dt);
l = zeros(1, 2000/dt);
n = zeros(1, 2000/dt+1);
max = 0;

for i = 1:9
    for j = 1:200/dt
        b(i*200*dt+j+1) = a(i);
    end
end

x = 0:dt:2000;          % A low resolution sine doesn't hurt because
were
c = sin(x)./2+0.5;      % only multiplying every once in a while

d = b.*c;       % Modulated Signal

e = d.*c;       % Demodulated Signal In Phase
```

```
for i = 100:200
    int_e(i) = sum(e(1:i));
end
sum_e = sum(e);
max = sum_e;

f = sin(x+phOFF*3.14159)./2+0.5;   % Arbitrary Phase Offset

g = d.*f;

for i = 100:200
    int_g(i) = sum(g(1:i));
end

sum_g = sum(g);
if(sum_g>max)
    max=sum_g;
end

h = sin(x+phOFF*3.14159+3.14159)./2+0.5;    % Quadrature Phase to
Arbitrary Phase Offset Signal

k = d.*h;

for i = 100:200
    int_k(i) = sum(k(1:i));
end

sum_k = sum(k);
if(sum_k>max)
    max=sum_k;
end

l = sqrt(k.*k+g.*g);

subplot(3,3,1);
plot(b);
title('Audio Signal');
axis([100*dt 500*dt 0 5]);

subplot(3,3,2);
plot(c);
title('Modulating Signal');
axis([100*dt 500*dt -1 2]);

p = 1-(sum(b)-sum(e))/sum(b);

subplot(3,3,3);
plot(x/dt, e, x/dt, b, 'k', 'Linewidth', 1);
title('Demodulated Signal In Phase');
text(100, 3, ['% of Signal Intact: ', num2str(p)],
'BackgroundColor',[.7 .9 .7]);
axis([100*dt 500*dt 0 5]);
```

```
subplot(3,3,6);
plot(int_e, 'Linewidth', 2);
title('Demodulated Signal In Phase');
text(100, sum_e/2, ['Sum of Signal: ', num2str(sum_e)],
'BackgroundColor',[.7 .9 .7]);
axis([100*dt 500*dt 0 max+10]);

subplot(3,3,4);
plot(g);
title('Demodulated Signal Out of Phase');
axis([100*dt 500*dt 0 5]);

subplot(3,3,5);
plot(int_g, 'Linewidth', 2);
title('Demodulated Signal Out of Phase');
text(100, sum_g/2, ['Sum of Signal: ', num2str(sum_g)],
'BackgroundColor',[.7 .9 .7]);
axis([100*dt 500*dt 0 max+10]);

subplot(3,3,7);
plot(k);
title('Demodulated Signal Out of Phase Quadrature');
axis([100*dt 500*dt 0 5]);

subplot(3,3,8);
plot(int_k, 'Linewidth', 2);
title('Demodulated Signal Out of Phase');
text(100, sum_k/2, ['Sum of Signal: ', num2str(sum_k)],
'BackgroundColor',[.7 .9 .7]);
axis([100*dt 500*dt 0 max+10]);

p = 1-(sum(b)-sum(l))/sum(b);

subplot(3,3,9);
plot(x/dt, l, x/dt, b, 'k', 'Linewidth', 1);
title('Sum of Squares I/Q Demodulated Signals');
text(100, 3, ['% of Signal Intact: ', num2str(p)],
'BackgroundColor',[.7 .9 .7]);
axis([100*dt 500*dt 0 5]);
```

# Appendix C: Interface from PIC to Serial Port

```
%
% MATLAB code
% Attempt at reading in serial data from the PIC
%

%create an object using the serial command
port = serial('COM1', 'BaudRate', 115200, 'DataBits', 8);

port.InputBufferSize = 20000;    %take in 1000 values to start with for
now i guess

% open the port and read in 20000 values
fopen(port);
[input_250, count_250] = fread(port, 20000);
fclose(port);

%initialize the vector to zeros so it's not grown in the for loop
corrected_input_250 = zeros(1,10000);


%this loop is needed to combine the two packets that i get from the
% PIC output
i = 1;
for j = 1:10000

    corrected_input_250(j) = bitshift(input_250(i), 8) + input_250(i+1);

    i=i+2;

end

%%use this M-file to manipulate and plot all the data that i just took
% in in the serial_comm file

% t = linspace(0, 125e-3, length(corrected_input_100));
% ts = linspace(0, 125e-3, 50000);
t = linspace(0, 2.5, length(corrected_input_250));
ts = linspace(0, 2.5, 100000);
%N = 5000;
N = length(corrected_input_250);
delta_f = 1/(N*250e-6);

xmin = 0;
xmax_50 = 0.0125/2;
xmax_100 = 0.0125;
xmax = 0.0125;
ymin = -2.5;
ymax = 2.5;

%
% These plots are for for the 250Hz sine wave
```

```
%
theta_250 = 4.126;
sine_250 = 2.*sin(2*pi.*250.*ts + theta_250);
figure(3);
subplot(2,1,1);
plot(t(1:100), corrected_input_250(1:100), 'rx');
% hold on
% plot(ts, sine_250);
% axis([xmin xmax_100 ymin ymax])
xlabel('Time (seconds)');
ylabel('Voltage');
title('A/D output plot for first 100 data points of a 250Hz sine wave
with best fit sine wave');

subplot(2,1,2);
plot(t(1:100), corrected_input_250(1:100));
%axis([xmin xmax ymin ymax])
xlabel('Time (seconds)');
ylabel('Voltage');
title('A/D output plot of a 250Hz sine wave');

figure(4);
frequency_plot_250 = abs(fft(corrected_input_250,N));
f = 1 : delta_f : delta_f * (length(frequency_plot_250)-1);
semilogy(f(1:length(frequency_plot_250)/2),
frequency_plot_250(1:length(frequency_plot_250)/2));
title('FFT of the 250Hz A/D output sine wave');
xlabel('Frequency (Hz)');
ylabel('Amplitude of the FFT');
```

# Appendix D: PIC Algorithm

```
/*****************************************************************************
* Code for PIC 30F3013 for the Laser Audio Bug MQP group
*
* The PIC controls the timing, sine wave generation, and data sampling.
*  We use the outputs to generate a sine and a cosine wave at a
*  frequency of 100kHz.  We also use the PIC to sample the final integration
*  value, and then reset the integrator circuit every 100uS.
*
* REVISION HISTORY:
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Author                    Date              Comments on this revision
*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Cody Brenneman     1/15/07
*
*****************************************************************************/

#include <p30f3013.h>

/* Macros for Configuration Fuse Registers (copied from device header file):*/
//_FOSC(CSW_FSCM_OFF & XT_PLL8);  /* Set up for Crystal multiplied by 8x PLL
*/
_FOSC(CSW_FSCM_OFF & XT_PLL4); //Use 4xPLL for 16.384MHz 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 */


# define RESET LATBbits.LATB7
# define HOLD LATBbits.LATB6
# define ALTERNATE ADCON2bits.ALTS
# define CHANNEL ADCHSbits.CH0SA
# define CONVERT ADCON1bits.SAMP
# define TRANSMIT U2STAbits.UTXEN

/* Global Variables and Functions */
int main (void);
void port_init(void);
void interrupt_init(void);
void tmr1_init(void);
void tmr2_init(void);
void tmr3_init(void);
void PWM_init(void);
void AD_init(void);
void UART_init(void);
```

```
void phase_delay(void);
void delay_2_5uS(void);
void delaymS(int time);
void __attribute__((__interrupt__)) _T1Interrupt(void); /*Declare interrupt ISRs */
void __attribute__((__interrupt__)) _T2Interrupt(void);
void __attribute__((__interrupt__)) _T3Interrupt(void);
void __attribute__((__interrupt__)) _ADCInterrupt(void);

int sine_value;
int temp;
float cosine_value;
char toggle = 0;
char low_output;
char high_output;
char value_present = 0;


int main (void)
{
        port_init();
        interrupt_init();
        PWM_init();
        AD_init();
        UART_init();
        tmr1_init();
        tmr2_init();
        tmr3_init();

        phase_delay();


        while(PORTDbits.RD8);           //waits here until a button is pressed,
        delaymS(50);                    // then it begins transmitting sampled values

        while (1)
        {

                if(value_present)
                {
                        value_present = 0;
                        sine_value = ADCBUF0;       //This sequence separates the high
                        temp = sine_value;          // and low values of the A/D
                                                    //conversion so
                        temp = temp & 0x00FF;       // that it can be transferred in two
                        low_output = temp;          // packets
                        temp = sine_value;
```

```
                        temp = temp & 0xFF00;
                        temp /= 256;
                        high_output = temp;

                        U2TXREG = high_output;      //fill the output buffer with the
                        U2TXREG = low_output;       // high and low output values
                }
        }
}

void phase_delay(void)
{
        T2CONbits.TON = 1;              //Turn on the "sine" wave
        delay_2_5uS();                  //Need a delay of 2.5uS so that they're 90
                                        //degrees out of phase
        T3CONbits.TON = 1;              //Turn on the "cosine" wave

        T1CONbits.TON = 1;

        return;
}

/*
        This function initializes the two PWM outputs we will use for the
         "sine" and "cosine" square waves.  Timer 2 and Timer 3 are used for
         the time bases for OC1 and OC2 respectively
*/

void PWM_init(void)
{
        OC1CONbits.OCSIDL = 0;          //PWM will still work during idle
        OC1CONbits.OCTSEL = 0;          //For output 1, timer 2 is the timer used
        OC1CONbits.OCM = 6;             //PWM mode on OC1, Fault pin disabled

        //OC1RS = 49;                   //For 5MHz crystal
        OC1RS = 82;

        OC2CONbits.OCSIDL = 0;          //PWM will still work during idle
        OC2CONbits.OCTSEL = 1;          //For output 2, timer 3 is the timer used
        OC2CONbits.OCM = 6;             //PWM mode on OC2, Fault pin disabled

        //OC2RS = 49;
        OC2RS = 82;

        return;
}
```

```
void AD_init(void)
{
        ADCON1bits.ADSIDL = 0;          //Continue in idle mode
//      ADCON1bits.FORM = 3;            //Set output to be signed fractional
//      ADCON1bits.FORM = 1;                    //Signed integer output
        ADCON1bits.FORM = 0;                    //Unsigned integer output
        ADCON1bits.SSRC = 0;            //Clearing SAMP bit starts conversion
        ADCON1bits.ASAM = 1;            //Sampling begins right after conversion
                                        //ends, SAMP bit automatically set
        ADCON1bits.SAMP = 1;            //Begin by sampling
        //Monintor the ADCON1bits.DONE bit to see when the conversion is done
        // The bit is 0 when the conversion is in process, is set to 1 when completed


        ADCON2bits.VCFG = 0;            //Use AVdd and AVss as references
        ADCON2bits.CSCNA = 0;           //Do not scan inputs, whatever that means
        //ADCON2bits.BUFS is the buffer fill status bit, look in the reference
        // manual on page 466 to figure out if i want this or not
        ADCON2bits.BUFM = 0;
        ADCON2bits.SMPI = 0;            //Set it so that the A/D converter generates an
                                        // interrupt after every conversion
        ADCON2bits.ALTS = 0;            //Always use MUX A

        ADCON3bits.ADCS = 14;           // A/D conversion clock = 10 * Tcy
        ADCON3bits.SAMC = 31;           //Auto Sample time = 31 * Tad
        ADCON3bits.ADRC = 0;            //Clock derived from system clock

        ADCHSbits.CH0NA = 0;            //Use Vref- as the negative input
//      ADCHSbits.CH0NA = 1;            //Use AN1 as negative input to S/H channel
        ADCHSbits.CH0SA = 5;            //Start by first converting AN5, will be
                                        //switched later to convert AN1 for the
                                        //"cosine"
        ADPCFG = 0xFFFF;                //Sets only AN0 and AN1 to be analog input
                                        //pins, the rest are under digital control
        ADPCFGbits.PCFG1 = 0;
        ADPCFGbits.PCFG0 = 0;       //Set channels AN0 and AN1 to be analog inputs
        ADPCFGbits.PCFG5 = 0;

        /*
                Tad = Tcy(ADCS+1)/2 = 1uS
        */

        /*
        There is only one S/H amplifier within the PIC!!!  Which means that we cannot
        do both A/D conversions that we need simultaneously.  We will need to somehow
```

```
        sample the integrator outputs with an external S/H circuit, then use the PIC to
        convert one integrator output, then switch the internal PIC S/H amplifier to the
        other input and convert that.  The total conversion time as set up right now is Tad
        = 1uS, so there should be plenty of time to convert one, switch, convert the other,
        and do any math necessary within a 250uS span.
        */

        ADCON1bits.ADON = 1;            // A/D converter is now on

        CONVERT = 0;

        return;
}

void UART_init(void)
{
        U2MODEbits.USIDL = 0;           //Continue during idle
        U2MODEbits.ALTIO = 0;           //Use U2TX and U2RX pins, not U2ATX
                                        //and U2ARX
        U2MODEbits.WAKE = 0;            //wake-up disabled
        U2MODEbits.LPBACK = 0;          //Loopback mode is disabled
        U2MODEbits.ABAUD = 1;           //Not going to use autobaud, but it's now on
                                        //the U2RX pin
        U2MODEbits.PDSEL = 0;           //8-bit data, no parity
        U2MODEbits.STSEL = 0;           //Use one stop bit

        U2STAbits.UTXISEL = 1;      //Interrupt when transmit buffer becomes empty
        U2STAbits.UTXBRK = 0;           //no break? what's a break?
        U2STAbits.URXISEL = 0;          //interrupt on every character recievied
        U2STAbits.ADDEN = 0;            //Address detect mode disabled

        //U2BRG = 10;         //57600 for 5MHz crystal with 8xPLL
        U2BRG = 8;            //115200 for 16.384MHZ crystal with 4xPLL

        U2MODEbits.UARTEN = 1;          //UART is now enabled

        U2STAbits.UTXEN = 1;            //Enable transmitting

        return;
}

void port_init(void)
{

        TRISB = 0xFFFF;             //Set only RB6, 7, 8, 9 to be outputs
```

```
                                        //  PWM outputs are RB8 and RB9 for OC1 and
                                        //OC2 respectively
                                        //  RB7 is the 100uS reset output
        TRISBbits.TRISB6 = 0;
        TRISBbits.TRISB7 = 0;
        TRISBbits.TRISB8 = 0;
        TRISBbits.TRISB9 = 0;

        TRISC = 0;              //Set all of PORTC to be outputs for LED testing
        TRISD = 0xFFFF;
        TRISF = 0;

        return;
}

void interrupt_init(void)
{
        INTCON1bits.NSTDIS = 0;         //Enable nesting of interrupts
        INTCON1bits.OVATE = 0;
        INTCON1bits.OVBTE = 0;          //Turn off whatever these are

        IEC0 = 0;                       //Turn off all interrupts, turn on the ones i
want individually later
        IEC0bits.T1IE = 1;          //Turn on timer 1 interrupt
        IEC0bits.ADIE = 1;          //Turn on the A/D converter interrupt
        IEC1 = 0;                   //turn off all other interrupts
        IEC2 = 0;

        //All inerrupts are automatically set to same priority, so don't change that

        return;
}

void tmr1_init(void)
{
        T1CONbits.TON = 0;              //Make sure the timer is off for now, it will
                                        //be turned on later
        T1CONbits.TSIDL = 0;            //Continue in idle mode
        T1CONbits.TGATE = 0;            //Turn off gated time accumulation,
                                        //whatever that is
        T1CONbits.TCKPS = 0;            //No prescalar
        T1CONbits.TCS = 0;              //Internal clock (Fosc/4)

        /* The following values are based on Fosc being 40MHz, which is
                a 5MHz crystal with an 8x PLL */
```

```
	/*
		To have a time of 100uS:
		  The counter will increment every 4/(4MHz) = 100nS
		  So we need a count in the PR register of 100uS/100nS = 1000
	*/

	//PR1 = 5000; //500uS for 5MHz crystal with 8xPLL
	PR1 = 4096;		//250uS for 16.384MHz crystal with 4xPLL

	/*  So when I finally turn the timer on, the timer will interrupt every 5uS.
		When I get to the interrupt I need to then toggle the output pin that
		I am using for the "sine" output */

	return;
}



/*  Timer 2 is the time base for OC1 PWM, which is the "sine" wave  */
void tmr2_init(void)
{
	T2CONbits.TON = 0;		//Make sure the timer is off for now, it will
					//be turned on later
	T2CONbits.TSIDL = 0;		//Continue in idle mode
	T2CONbits.TGATE = 0;		//Turn off gated time accumulation,
					//whatever that is
	T2CONbits.TCKPS = 0;		//No prescalar
	T2CONbits.TCS = 0;		//Internal clock (Fosc/4)
	T2CONbits.T32 = 0;		//Use Timer 2 and Timer 3 as two separate timers,
					//disable the 32-bit timer mode

	/* The following values are based on Fosc being 40MHz, which is
		a 5MHz crystal with an 8x PLL */

	/* Desired PWM frequency is 100kHz
		PWM Period = (PR2 + 1) * Tcy * (Timer 2 prescale value)
		1/100kHz = (PR2 + 1) * 0.1uS * 1

		PR2 = 99
	*/

	//PR2 = 99;		//This is the right value for my 5MHz crystal
	PR2 = 163;	/	/This value is for the 16.384MHz crystal

	return;
}
```

```
/*  Timer 3 is the time base for OC2 PWM, which is the "cosine" wave  */
void tmr3_init(void)
{
        T3CONbits.TON = 0;          //Make sure the timer is off for now, it will be
                                    //turned on later
        T3CONbits.TSIDL = 0;                //Continue in idle mode
        T3CONbits.TGATE = 0;        //Turn off gated time accumulation, whatever that is
        T3CONbits.TCKPS = 0;                //No prescalar
        T3CONbits.TCS = 0;                  //Internal clock (Fosc/4)

        /* The following values are based on Fosc being 40MHz, which is
                a 5MHz crystal with an 8x PLL */

        /* Desired PWM frequency is 100kHz
                PWM Period = (PR3 + 1) * Tcy * (Timer 3 prescale value)
                1/100kHz = (PR3 + 1) * 0.1uS * 1

                PR3 = 99
        */

        //PR3 = 99;             //This value is right for the 5MHz crystal
        PR3 = 163;              //This value is for the 16.384MHz crystal

        return;
}

/*
_T1Interrupt() is the timer 1 interrupt service routine (ISR).
*/
void __attribute__((__interrupt__)) _T1Interrupt(void)
{
        IFS0bits.T1IF = 0;              //Clear the T1 interrupt flag or else
                                         //the CPU will keep vectoring back to the ISR

        CONVERT = 0;                    //Now start the A/D conversion (starts with a 0)

        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");

        RESET = 1;                      //Pulse the RESET MOSFET for 1uS

        delay_2_5uS();
        delay_2_5uS();
        delay_2_5uS();
        delay_2_5uS();

        RESET = 0;
}

void __attribute__((__interrupt__)) _ADCInterrupt(void)
{
        IFS0bits.ADIF = 0;              //Clear the A/D interrupt flag so I don't
come right back

        sine_value = ADCBUF0;
        value_present = 1;
}

void delay_2_5uS(void)
{
        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");
//from here on is needed only for the 16.384MHz crystal
        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");

        return;
}

void delaymS(int time)
{
        int i;
        int j;
        for(j=0; j<time; j++)
        {
                for(i=0; i<1042; i++);
        }
        return;
}
```