DESIGN, CHARACTERIZATION AND APPLICATION OF A MULTIPLE INPUT STETHOSCOPE APPARATUS

A Thesis

Presented to

The Faculty of

California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

Of the Requirement for the Degree

Master of Science in Electrical Engineering

By

Spencer Geng Wong

September 2014

COMMITTEE MEMBERSHIP

TITLE:	Design, Characterization and Application of a Multiple Input Stethoscope Apparatus
AUTHOR:	Spencer Geng Wong
DATE SUBMITTED:	September 2014
COMMITTEE CHAIR:	Wayne Pilkington, Ph.D. Associate Professor Electrical Engineering Department
COMMITTEE MEMBER:	Jane Zhang, Ph.D. Professor, Associate Dept. Chair Electrical Engineering Department
COMMITTEE MEMBER:	Tina Smilkstein, Ph.D. Assistant Professor Electrical Engineering Department

ABSTRACT

Design, Characterization and Application of a

Multiple Input Stethoscope Apparatus

Spencer Geng Wong

For this project, the design, implementation, characterization, calibration and possible applications of a multiple transducer stethoscope apparatus were investigated. The multi-transducer sensor array design consists of five standard stethoscope diaphragms mounted to a rigid frame for a-priori knowledge of their relative spatial locations in the x-y plane, with compliant z-direction positioning to ensure good contact and pressure against the subject's skin for reliable acoustic coupling. When this apparatus is properly placed on the body, it can digitally capture the same important body sounds investigated with standard acoustic stethoscopes; especially heart sounds. Acoustic signal inputs from each diaphragm are converted to electrical signals through microphone pickups installed in the stethoscope connective tubing; and are subsequently sampled and digitized for analysis. With this system, we are able to simultaneously interrogate internal body sounds at a sampling rate of 2 KHz, as most heart sounds of interest occur below 200 Hz.

This system was characterized and calibrated by chirp and impulse signal tests. After calibrating the system, a variety of methods for combining the individual sensor channel data to improve the detectability of different signals of interest were explored using variable-delay beam forming. S1 and S2 heart sound recognition with optimized beam forming delays and inter-symbol noise elimination were investigated for improved discernment of the S1 or S2 heart sounds by a user. Also, stereophonic presentation of heart sounds was also produced to allow future investigation of its potential clinical diagnostic efficacy.

Keywords: Beamforming, MATLAB, Multiple Array Stethoscope Apparatus, Heart signals

ACKNOWLEDGMENTS

For this project I would like to acknowledge some friends that helped me out: Hani Abidi for helping me choose the Audio Amplifier from Maxim Integrated. I would also like to thank Jiwon Han and Anthony Chan for helping document my project by taking pictures and drawing Multiple Stethoscope Apparatus on AutoCAD. And I would like to thank my family for their support. Lastly I would like to thank Professor Pilkington for giving me a thesis idea and helping me out greatly with my thesis throughout the year.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

1 Introduction

Heart disease is a pervasive problem in the United States, and the number one cause of death in our country. Every year about 600,000 people die of heart failure in the U. S. (Murphy, Xu, & Kochanek, 2013). This mortality rate could decrease if there were better early detection methods for the various conditions that lead to heart attacks and heart failure. Current cardiac screening methods rely on simple acoustic stethoscopes and the training of the physicians using them to discern subtle indicators of numerous different cardiac abnormalities or malfunctions. These simple stethoscopes allow only basic tuning of their frequency range sensitivity through the selection of one of two bell shapes for the acoustic pickup. Also, the diagnostic analysis of the resulting detected sounds by the physician is very subjective and requires extensive training and practice. To improve the sensitivity of the stethoscope system, and to provide an easier to discern signal for the diagnostician, a multiple-element stethoscope apparatus using beamforming techniques for signal aggregation will be studied in this report. The objective of this project is to design and explore the potential of using multiple-element electronic stethoscopes to better observe the internal functioning of the human body, to assist physicians with more accurately diagnosing diseases and dysfunctions in the body. The specific application I will by studying is the use of heart sounds to diagnose heart disease or abnormalities. Analysis will be done by storing a simultaneous recording of heart sound sample from 5 stethoscopes and then writing scripts in MATLAB to analyze the data. Stereophonic presentation of the sound collected will also be studied in this report.

1.1 Overview

There are several technical concepts that need to be understood for this project. The first such concept is beamforming theory. Beamforming is important here because it describes how to integrate multiple input signals from a transducer array in order to best isolate and amplify the sound of interest while attenuating unwanted, interfering sounds from other sources. Also important is an understanding of the characteristics of the body sounds of interest – where they are from; how they are produced; and what they are expected to sound like. Since this project is about listening to the heart, it is important to know how the heart generally works; what normal and abnormal sounds can be heard, and what they mean. In addition to knowing about the body, it is necessary to know how a stethoscope works, since a standard stethoscope will be a vital part of the data collection system used here, and the quality of the data signal acquired will depend on understanding the characteristics and proper application of the stethoscope pickup. Other concepts that will be explained are the algorithms used to determine the phase delay needed to align the five input signals received, and the particular signal to noise ratio used as a metric for the quality of the received and beamformed signals. These algorithms and metrics applied the statistic concepts of cross correlation, correlation coefficient, and variance which are also describe below.

1.2 Beamforming

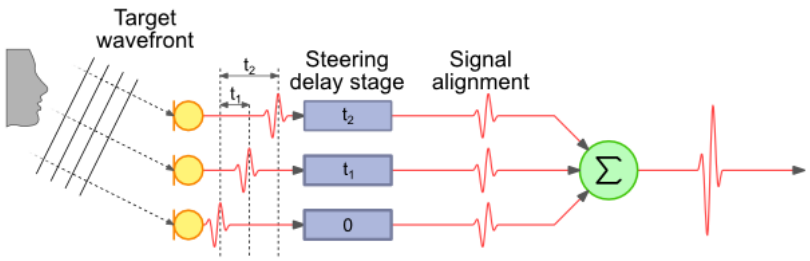


Figure 1.1 Beamforming Example (Greensted, 2010)

Beamforming is a technique used to combine multiple signals collected by a microphone array together into a single composite signal. The goal of beamforming is to amplify sounds that come from a specific source location and attenuate other sounds, including signals from other sources and random

noise. This is a very widely-used technique employed in many applications like robotics, communication, biomedical imaging, and more. There are many different beamforming implementation algorithms in use for different applications. Some algorithms are delay and sum algorithm, minimum variance, sidelobe cancelling, linearly constrained beamforming, adaptive beamforming (Synnevåg, Austeng, & Holm, 2007) (Steinhardt & Van Veen, 1989). The particular beamforming method studied in this project is the *delay and sum* algorithm.

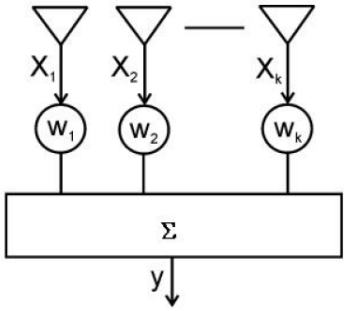


Figure 1.2 Simple Structure for Beamforming

1.2.1 Delay and Sum

The delay and sum algorithm is the most basic beamforming algorithm and easiest to implement. The structure of the Delay and Sum is shown above in Figure 1.2. As shown, there are two main components to the delay and sum, the delay buffer (elements $W_1..W_i$) and the summer. The purpose of the delay buffer is to properly align the individual received signals in time by delaying each signal differently. By adjusting the starting index of each signal in the buffer, different time delays can effectively be applied to each signal, so that the wavefront features emanating from the source point of interest received by each sensor are temporally aligned with the same features in all the other sensor signals before getting summed together into a single composite signal, as shown in Figure 1. This way, the desired signals from the source point of interest will reinforce each other as they are added together; and signals or noise from other sources will be attenuated by averaging.

To determine the exact delays needed, the spatial location of the source, the three-dimensional location of each sensor relative to each other or some reference point, and the speed of sound in the medium between the source and the sensors must be known.

$$Delay = \frac{Distance\ from\ Point\ of\ Interest\ to\ Sensor\ Element}{Speed\ of\ sound\ in\ the\ medium\ between\ POI\ and\ sensor}$$

Equation 1.1 Delay of Sound

Once the multiple signals are properly aligned in time, they are then simply summed together. If aligned properly, the signal transmitted from the point of interest will be reinforced and amplified by constructive interference as these signals are added together, and all other interfering sounds should average out of the summed signal, as their features will not be aligned in the different sensor signals. Figure 1.1 shown above illustrates the delay and sum beamforming. This figure shows a planar wave front from a distant source arriving at an array of sensors. For heart sounds, as in this application, spherical wave fronts emanating from a local region on the heart roughly 5 cm away from the stethoscope pickup are expected.

Determining the proper beamforming delays for each sensor signal in this project was accomplished by aligning the peaks of the signal in an excerpt surrounding the feature of interest. This is done using a cross correlation method described later in the "Determine Phase *Delay*" section.

1.3 Body Sounds

There are many different sounds created within the body, some of which include respiration, digestive, movement sounds, and heart sounds. Figure 1.3 shows the spectral intensity and frequency range of the different body sounds, along with our threshold of audibility (the level below which the sounds cannot be heard). It is also important to note that human's audible range is generally between 20 Hz to 20 KHz; while many internal body sounds exist below this range. This shows that a physician or cardiologist using a standard stethoscope is only able to hear a fraction of the spectral intensity of the heart sounds occurring, and an even smaller portion for respiratory and digestive sounds. Therefore it would be beneficial to have a method to better analyze all types of body sounds. A particular body sound

of interest is the heart, which I will be beamforming in this project. Heart murmur will also be discussed because it is a potential application for this project as an indicator for heart problems.

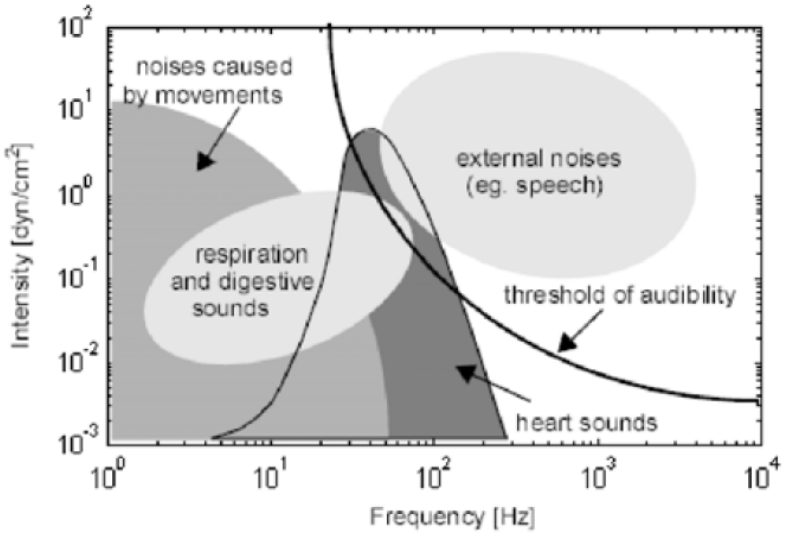


Figure 1.3 Spectral intensity of heart sound record (Jatupaiboon, Pan-ngum, & Israsena, 2010)

1.3.1 Heart Sounds

The normal heart sounds we are accustomed to hearing are created from the successive closing of the four valves in the heart. The frequency spectral distribution of such normal heart sounds is shown in Figure 1.3, and an example of the acoustic waveform of normal heart sounds is shown in

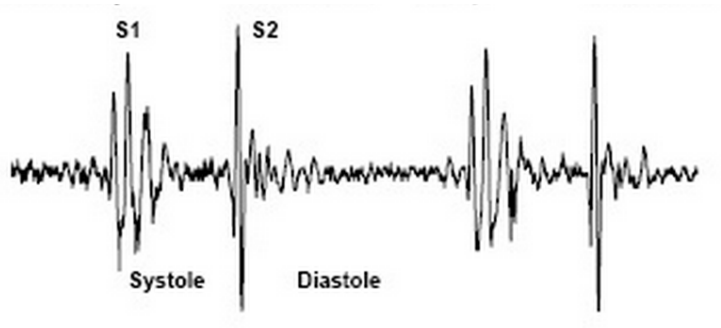


Figure 1.5. The anatomy of the heart is also displayed in Figure 1.4 to provide a reference for the anatomical terms that follow.

The four valves in the heart are the Tricuspid, Mitral, Pulmonary, and the Aortic valves**.** The sounds resulting from the tricuspid and the mitral valves closing compose the characteristic “S1” heart sound. Similarly, the sounds caused by the pulmonary and the aortic valves closing in sequence compose the “S2” heart sound. Discerning these two characteristic signals (S1 and S2) will be the focus of this project.

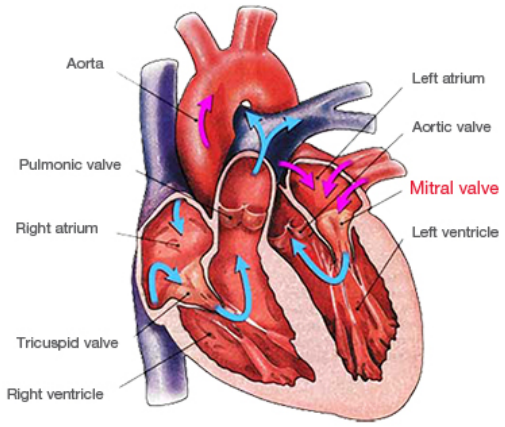


Figure 1.4 Location of heart valves (human heart picture with labels)

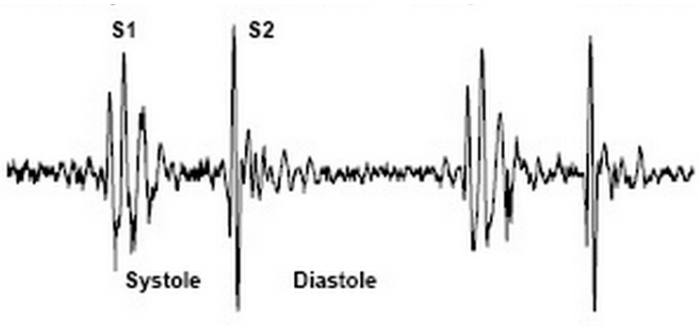


Figure 1.5 Normal Heart Sound S1 and S2 (144)

1.3.2 S1 and S2 Heart Sounds

Figure 5 shows typical S1 and S2 signals from a normal heart as they would appear on the output of an acoustic sensor. The S1 heart sound is caused by sudden cessation of blood flow caused by the closure of the mitral and tricuspid valves. The mitral valve opening is normally not heard except when there is Mitral stenosis (narrowing of the valve). The S2 sound is caused by the Aortic and Pulmonary valves closing. For the S2 sound the aortic valve precedes the pulmonary valve. The period between the

S1 and the S2 sound is the *systole*. This is the period of time when the blood is driven out of the heart. The period between the S2 and the S1 is the *diastole*. That is the period of time when the heart refills with blood. Flow of blood into the heart during this rapid filling diastole phase is not normally heard, except in certain pathological states where it constitutes a third heart sound (S3).

1.3.3 Heart Murmur (National Heart, Lung, and Blood Institute, 2012)

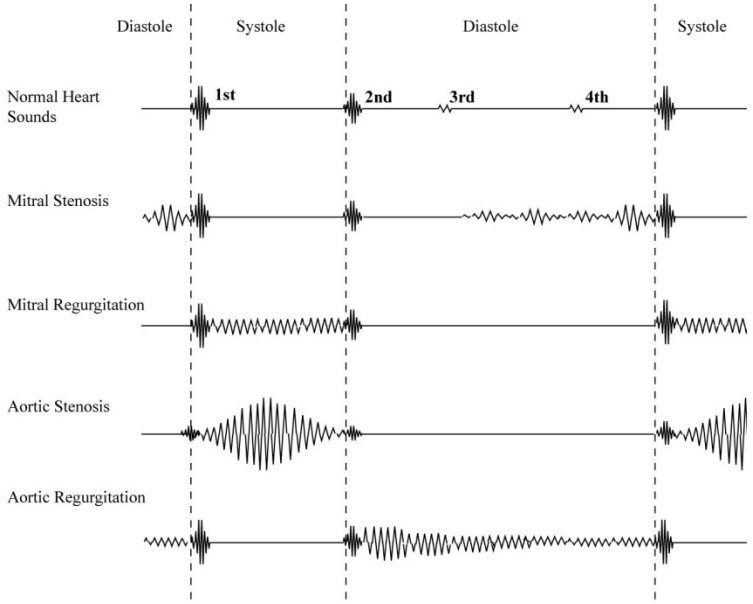


Figure 1.6 Examples of Heart Murmur (Cardiac Murmurs, 2006-2013)

Heart murmur is a general term referring to an extra or unusual sound heard during a heartbeat. It ranges in volume from faint to very loud. Heart murmurs are usually heard as a whooshing sound or a swishing sound caused by blood flow at unexpected times. Shown in Figure 1.6 are different types of heart murmur, each resulting from a different problem in the functioning of the heart.

There are two different classes of heart murmurs, *innocent* (or functional) and *abnormal* murmurs. *Innocent* heart murmurs are ones that aren't harmful, and often occur in children. They are caused when blood rushes through the heart quickly during normal function, while no heart disease processes exist.

There may be an underlying but benign medical condition that leads to the innocent murmur. On the other hand, an *abnormal* heart murmur is a symptom of a more serious heart valve defect or disease.

Heart valve disease is diagnosed when the heart isn't working as well as it needs to because of defects or deterioration in the heart valves from aging or disease. The five basic heart valve problems are *regurgitation, prolapse, stenosis, sclerosis,* and *artesia*. *Regurgitation* occurs when the valves don't close tightly enough, causing blood to backflow into the heart. Valve *prolapse* is a bowing of a valve that causes some leaking and most often involves the mitral valve. Valve *stenosis* is narrowing of a heart valve. This often occurs over time as the valve scars due to injury or infection such as rheumatic fever, or from a congenital birth defect. Calcification of a valve may also result in stenosis. Stenosis causes the heart muscle to work harder to push blood through the narrowed opening and can lead to heart failure. Valve *sclerosis* is a mild narrowing and stiffening of the valve due to aging. It is most often seen in the aortic valve, and is associated with atherosclerotic heart disease. Finally, *artesia* is a total closure or absence of a heart valve preventing blood to flow through. This is typically a congenital defect that must be corrected at birth.

Holes in the walls of the heart (the septum that divides the heart chambers) can also be the source of a heart murmur. Atrial septal defect (ASD) describes a hole in the wall that separates the collecting chambers of the heart while a ventricular septal defect (VSD) affects the wall dividing the pumping chambers.

While some abnormal heart sounds may not indicate serious problems, others do. Heart valve disease may result in heart failure, stroke, blood clots, or death due to cardiac arrest.

1.4 Stethoscope

The stethoscope is the primary acoustic medical device for *auscultation* - listening to the internal sounds of an animal or human body. The same standard two-sided chest piece acoustic stethoscope has essentially been in use since the 1940's, with several minor improvements made in its design since its inception. The acoustic stethoscope was an important component to this project because it remains the

acoustic signal acquisition device of choice. Thus, a high-quality acoustic stethoscope model was used for this project to provide sufficient signal sensitivity for accurate data acquisition.

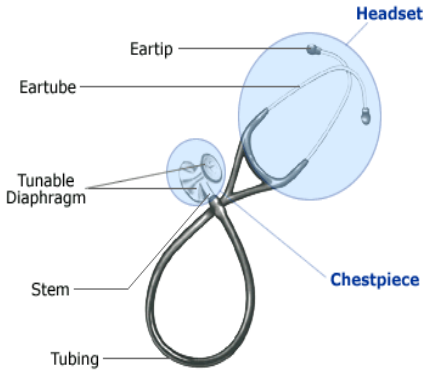


Figure 1.7 Stethoscope Anatomy

A stethoscope consists of three main components - the chest piece, tubing, and the headset. The chest piece has two sides to contact the patient and capture sounds from inside the body. One side of the chest piece is known as the *diaphragm*. This part of the chest piece is larger in diameter and covered with a flexible membrane that contacts the skin. This side of the chest piece is used to collect higher frequency sounds compared to the other side, known as the *bell*, which favors lower frequency sounds. The sound pressure wave captured by either side of the chest piece then couples into the column of air inside of a flexible tube. The tube is a Y-shaped rubber hose that directs the sound wave collected by the chest piece up to the headset, splitting it to provide sounds to both ears of the listener. The headset directs the sounds passed along by the flexible tubing in to the physician's ears. It consists of two metal tubes, each ending in a rubber ear tip to seal with the ear canal opening in order to block out other environmental sounds when used by the physician.

Placement of the stethoscope diaphragm is very important when listening to internal body sounds. In this project we will be observing heart sounds. There are four essential locations for stethoscope placement to listen to the heart as shown in the figure below (Figure 1.8). Here, the red dots indicate the locations to place the diaphragm to listen to each of the heart valves, and the black ovals

indicate the location of the associated valves. The stethoscope apparatus built for this project will be placed in the general vicinity of these auscultation sites.

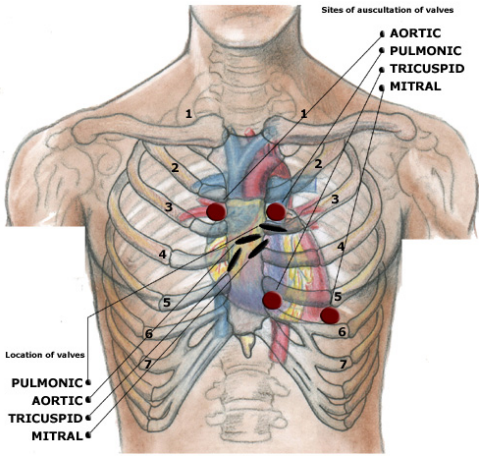


Figure 1.8 Auscultatory Sites on the body (Heart Anatomy)

1.4.1 Tunable Stethoscope

All modern stethoscopes have a tunable sensitivity frequency profile. The two-sided chest piece provides a course adjustment in the range of frequencies picked up by the stethoscope. The diaphragm side of the chest piece couples higher frequency sounds into the stethoscope, while the bell pickup is sensitive only to lower frequencies. The frequency response of the diaphragm pickup is also greatly affected by the amount of pressure applied to the stethoscope as it contacts the body. An example of this is shown in Figure 1.9 and Figure 1.10, where high frequency sounds will be detected when more pressure is applied and lower frequencies are coupled when less pressure is applied. This feature introduces a complication in this project as the five diaphragms used as acoustic pickups will have slightly different frequency responses depending on how much pressure they each produce when placed against the chest.

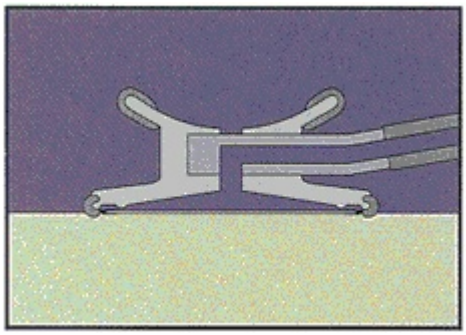


Figure 1.9 Bell Mode – Light pressure (low-frequency sensitivity) (Medical Department Store)

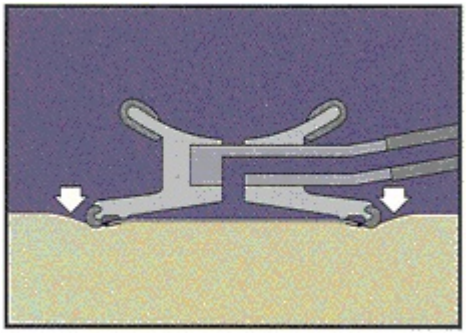


Figure 1.10 Diaphragm Mode – More contact pressure (high-frequency sensitivity) (Medical Department Store)

1.5 Statistical Methods

Statistical principles were used for time-aligning and analyzing the waveform data collected in this project. Key concepts used in this project were cross correlation and the correlation coefficient, used to determine the phase delay; and statistical variance applied in the computation of the signal to noise ratio.

1.6 Cross Correlation

Cross correlation was used to determine the relative phase difference between the five body sound waveforms collected by the multi-element transducer. The cross correlation is a measure of how similar two signals are to each other, as the sample position offset between the two signals is varied. The sample offset resulting in the highest positive cross-correlation summation indicates the optimum alignment between the features in two signals. The equation of the discrete-time cross correlation is shown below in Equation 1.2 .

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f^*[m]g[m+n]$$

Equation 1.2 Cross Correlation

In this equation, '*f* 'and '*g'* represent sampled signals from two different sensors, and '*n*' is the sample position offset introduced between the two signals with a sample rate of 2 KHz.

1.6.1 Correlation Coefficient

The Correlation Coefficient is a function that determines how similar two signals are. If the Correlation Coefficient has a value of 1, then there is perfect correlation between two signals. If the coefficient is 0, then there is no correlation (no similarity) between the signals; and if the coefficient is -1, then there is a negative correlation – indicating that the two signals have the same shape but the opposite polarity. The correlation coefficient equation is shown below in Equation 1,3.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Equation 1.3 Correlation Coefficient

For the correlation coefficient equation, '*cov(x,y)*' represents the covariance function, which is the measure the similarity between two variables; where:

$$cov(X,Y) = E[\,(X-E(X))\,(Y-E(Y))\,]$$

Equation 1.4 Covariance

And *E( )* is the expected value (or mean) of a random variable. The symbol 'σ' in the correlation coefficient computation represents the standard deviation of the random variable. Standard Deviation is a measure of how much the data points are spread out from the mean value. The division of the covariance by the individual variable standard deviations in Equation 1.3 provides a normalized measurement of the correlation between the two variables; with values between -1 to +1.

1.6.2 Determine Phase Delay

There were a number of steps needed to determine the relative phase delay between two compared signals. The MATLAB Code used to determine the phase delay is shown below in Table 1.1.The first step is to extract the samples in the vicinity of the feature of interest within the two signals. Then the cross correlation peak index is used to find the relative time shift needed to best align that

desired feature in the two signals. For this, the offset index producing the peak correlation value is subtracted from the center index of the cross correlation result. This difference will determine how many sample intervals one signal needs to be shifted to align the other.

Thephasedelay.m

```
function [numshift, timedelay] = thephasedelay(data1,data2,samplingfreq)
sampletime = 1/samplingfreq;
c = xcorr(data1, data2);
centerofxcorr = size(c,1)/2;
maxc = max(c);
numshift = find(c == maxc)-centerofxcorr;
timedelay = (numshift*sampletime;
end
```

Table 1.1 MATLAB Code used to determine the Phase Delay

1.6.2.1 Variance

A variance computation was used to determine the signal to noise ratio (SNR), for comparing SNR before and after signal compositing. Variance is the measure of how spreads out the numeric values are in a sample sequence. The variance equation is shown below in Equation 1.5.

$$Var(X) = E[(X-\mu)^2] = E[X^2] - (E[X])^2$$

Equation 1.5 Variance

Variance is defined using the expected value which is represented again as 'E[X]' or. In Equation 1.5, the symbol μ represents the average of the 'X' data; and 'X' is the array of data points from which the variance is computed. The variance is a special case of the covariance, where the two variables are the same.

1.6.3 Signal to Noise Ratio

The Signal to Noise Ratio was determined by using one of the equations shown below (Equation 1.6 or Equation 1.7), depending on the situation. The variables that needed to be determined were the variance of the signal and the variance of the noise. The signal variance was found by calculating the variance only for the samples occurring around when the S1 or S2 Signal feature occurred. The noise

variance was found by calculating the variance for the periods between the S2 signal portion and the S1 portion. For Equation 1.7, '$A_{peak\ signal}$' and '$A_{peak\ noise}$' represent the peak signal and noise amplitudes.

Equation 1.6 Signal to Noise Ratio using Variance

Equation 1.7 Signal to Noise Ratio using Peak of Signal

1.7 Stereophonic Sound

An additional application that was explored in this project was stereophonic sound. Stereophonic sound is sound reproduction that preserves the directional phase cues that enable a person to localize the position of the sound source for an audio signal. This is accomplished by having two or more independent speakers presenting the same basic sound with two different phase delays to our ears; as our brains are trained to interpret these relative phase differences as indicators of differences in distance from the sound source to each ear.

A simple stereophonic acoustic stethoscope is available commercially from Prestige Medical, Inc. (Model 131). It uses a patented (US Patent #D625804) dual-chamber chest piece and tubing to enable stereo sound channeling. It is intended mostly to improve the volume and clarity of auscultation sounds.

Figure 1.11 Prestige Medical Model 131 Stereo Stethoscope

Stereophonic reproductions of heart sounds using signals from different pairs of auscultation points were created in this project to study if this could provide an additional clinical advantage to the

multiple stethoscope apparatus developed for this project. The results here differ from those of the stereo acoustic stethoscope in that the sensing points for the two audio channels are much further spatially separated in the multi-stethoscope apparatus than the split diaphragm sensor of the commercial acoustic stethoscope. This should provide significantly more difference in sounds provided to each ear; which should provide better discrimination of the sound source location.

2 Electronic and Multiple-Pickup Stethoscopes - State of the Art

2.1 Multisensor Stethoscopes Beamforming

Previous research has been done on multi sensor stethoscopes by Anita McKee and Rafik Goubran at Carleton University. Anita McKee has published a journal article and a thesis report on that topic. In Anita's thesis report, she studied beamforming using a circular array of stethoscopes. Her study evaluates two different methods of localization of the source of sound. And of these two methods they studied how different parameters can affect beam forming performance, such as number of microphones, signal frequency, speed of sound. The two source localization methods studied were Method A: Maximum Power in Beam Shape and Method B: solving system equation.

Some conclusions that were found was that Method A and Method B are both effective at locating the source when the source is located in the center of the array; and that neither method is that effective when the source is located far away from the center of the array. Some unique qualities about Method A are that it can identify multiple source locations if they have the same power. And some unique qualities about Method B are that the more systems of equation used the better the localization is. One problem with Method B, is that depending on the system equations used different answers on localizations can result. These findings are helpful for source location to aid traditional geometric beamforming.

2.2 Multi-Channel Stethograph Apparatus

Figure 2.1 Images of the Multi-Channel Stethograph(Stethographics inc.)

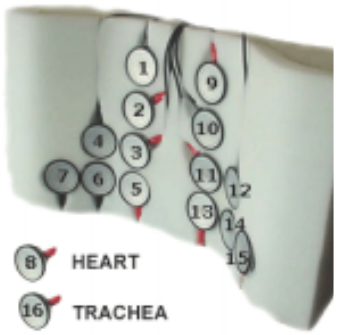


Figure 2.2 Images of the Multi-Channel Stethograph(Stethographics inc.)

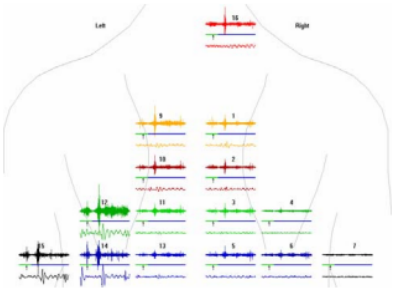


Figure 2.3 Visual Waveform (Stethographics inc.)

A product similar to the project discussed in this paper is the Multi-Channel Stethograph. The Multi-channel Stethograph is a product designed by Stethographics. As shown in Figure 2.3, it is a 16 digital stethoscope apparatus located in foam material so that it can form around the body. The application of this product is listening to the lungs, which is much quieter than the heart. The advantage of this product is that it is non-invasive, it records and display the body sounds as shown in Figure 2.3, and it provides specialized signal processing that enable localizing the source of the lung sounds. This product is not for sale in the United States, however indicating that it has most likely not received FDA approval as a diagnostic instrument. (Stethographics inc.)

2.2.1 Electronic Stethoscope

Electronic Stethoscopes work by converting acoustic sounds to an electrical signal; and then amplifying the signal to make them easier to hear. They may also provide noise filtering and sound recording capabilities. Electronic stethoscopes are available from ADC, ADS Global Telemedicine, Cardionics, 3M Littmann, Welch Allyn, Thinklabs,

In these products, there are three different methods used for the conversion of acoustic sounds to electrical signals. The simplest method is to place a microphone into the chest piece of the stethoscope, as was done in this project. This method has the shortcoming of susceptibility to ambient noise interference. The second, more commonly used method is to employ a piezoelectric crystal as the sound transducer. These directly convert the compression and expansion of the crystal caused by the acoustic pressure wave variations they receive into an electrical signal. 3M Littmann and Welch Allyn brands of electronic stethoscopes use a piezoelectric crystal transducer. The last conversion method is to use an electromagnetic diaphragm with a conductive inner surface to form a capacitive sensor. This is the approach used in the Thinklabs products. (National Telehealth TechnologyAssessment Resource Center)

Figure 2.4 Littmann Model 3200 Electronic Stethoscope

2.2.2 Littmann Electronic Stethoscope Model 3200

One of the highest rated electronic stethoscopes on the market right now is the 3M™ Littmann Electronic Stethoscope Model 3200. The key feature of this stethoscope is the ability to reduce noise by 85% on average and to amplify sound by 24 times. It has been shown that this electronic stethoscope makes it easier to detect S3 gallops, aortic regurgitation murmurs and abnormal lungs sounds compared with a non-electronic, cardiology-type stethoscope. This electronic stethoscope also has Bluetooth® connectivity for external digital recording devices. (3M Littman Stethoscope)

Electronic Stethoscopes would seem to have a big advantage over traditional acoustic stethoscopes. One advantage that electronic stethoscopes have is the ability to overcome low sound level problems or hearing impairment in the listener that acoustic stethoscopes can not address. Some can also reduce the ambient or motion noise in the signal. Another advantage is that the sound received by electronic stethoscopes can be digitally recorded and analyzed. Also the recorded data can be emailed to a physician at a different location for analysis.

Despite these key advantages, electronic stethoscopes are still not widely used by physicians. The primary reason for this is the higher cost of electronic stethoscopes compared to simpler acoustic stethoscopes. For example, the Littmann 3200 electronic stethoscope list price is currently $538, compared to the $115 cost of the Littmann Classic II S.E. acoustic stethoscope or $275 for the Littmann Cardiology stethoscope. Another impediment to the acceptance of electronic stethoscopes is that they produce different sounds (different spectral distributions) than acoustic stethoscopes; which means that physicians have to retrain their ear to listen to recognize heart sound artifacts with an electronic stethoscope. The other disadvantage that some electronic stethoscopes have is that they may over amplify some ambient or motion noise, which may potentially damage the clinician's hearing. This is another reason that noise reduction is an important feature that is becoming more common in newer electronic stethoscope models.

3 Multiple Digital Stethoscope System Design

3.1 Stethoscope System Architecture

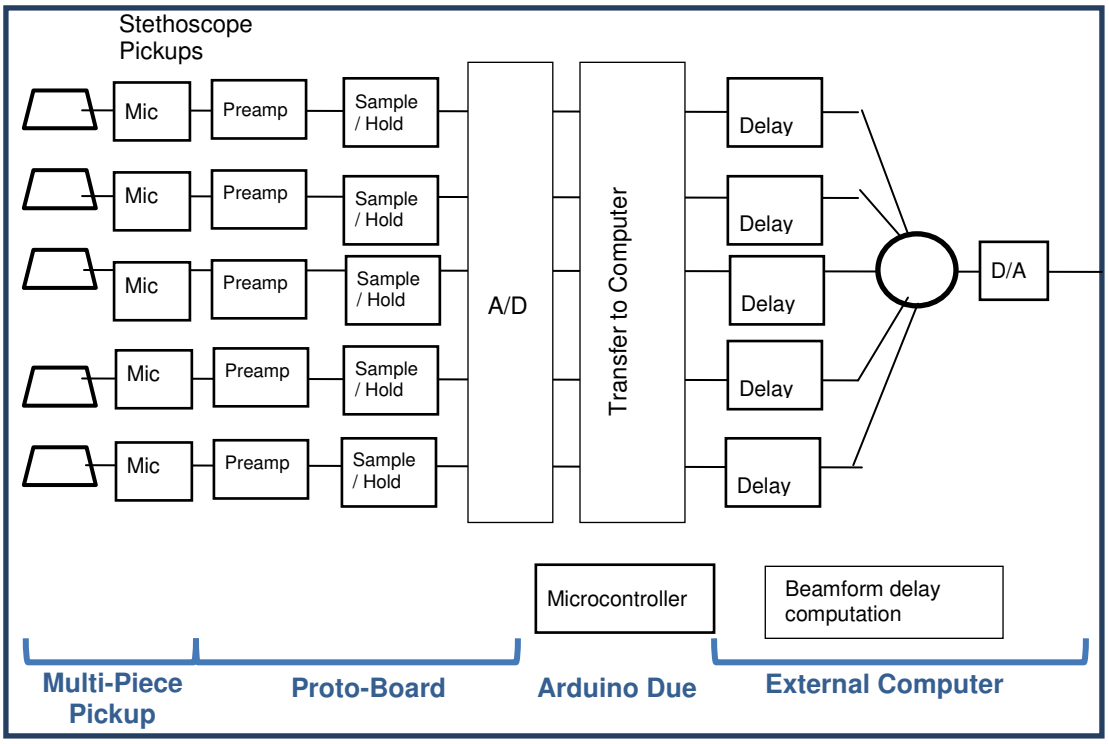


Figure 3.1 Stethoscope System Architecture

A multiple-sensor digital stethoscope was designed for this project to simultaneously acquire heart signals from multiple locations on the body surface. The architecture of the system is shown above. The system architecture is implemented in four major physical components. There is a multi-piece pickup, a prototype circuit board, an Arduino Due board, and an external computer. The multi-piece pickup converts the body sounds to electrical signals. Then the circuit board conditions the analog signals and converts the analog signals to a digital data stream. The microcontroller controls the data acquisition on the circuit board, and records the digital signals values to be transferred to the external computer. Then the external computer is used to analyze and combine the data collected by the multi-sensor digital system.

3.1.1 Multi-Piece Pickup

Each multi-piece pickup includes a stethoscope chest piece and a microphone. This subsystem collects body sounds and converts them into an electrical signal using the microphone. The electrical signals produced by the microphone are then processed by the Circuit Board.

3.1.2 Circuit Board

The circuit board has most of the electronics of the system. It holds the Preamplifiers, Sample and Hold Chip and the Analog to Digital converter (ADC). The signals received by the microphone are processes on this board. First the analog signal is boosted using preamps. After the preamp, the signal is then uniformly sampled by the sample and hold chip. The sampled signal is converted into digital data by the analog to digital converter. The digital signal is then sent to the Arduino Due via SPI communication.

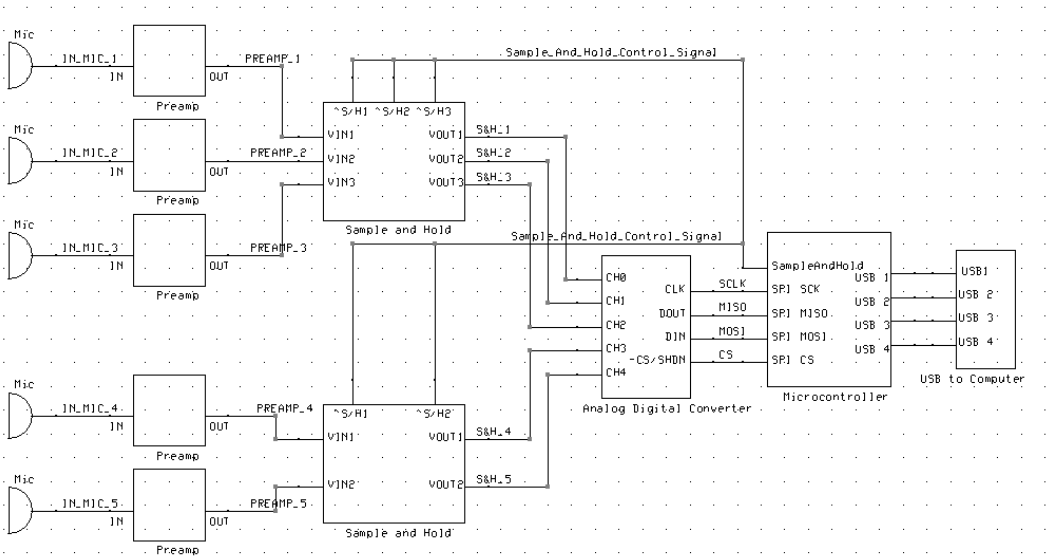


Figure 3.2 Basic control signal and data flow layout

3.1.3 Microcontroller

The Microcontroller coordinates the data acquisition on the circuit board of the system. It controls when the analog signal from the microphone should be held and what signals the analog to digital converter should convert. In addition to controlling the circuit board it also stores the digital values in

memory to be transferred to the external computer.

3.1.4 External Computer

The external computer is where the data processing occurs. The data collected by the Microcontroller is transferred to the computer via USB interface. Once the data is in the computer, the beamforming signal processing can be accomplished along with other measurements and characterizations needed for the various experiments performed here to demonstrate the potential usefulness of this apparatus.

3.2 Assumptions

The assumptions made for this project are:

- The different frequency responses of the Stethoscope pickups due to differences in contact pressure are neglected
- Air coupling of acoustic signals is a sufficient method for characterizing the stethoscope
- The droop rate of the sample and hold chip is negligible during the sampling interval
- 2 KHz is a sufficient sample rate for monitoring heart sounds, based on observing Figure 1.3

3.3 Electronic Design Architecture

The front-end electronic architecture of this project is shown in Figure 3.3. There are five stethoscope pickups simultaneously sampled by the microcontroller; with sampled waveforms temporarily saved in the on-board SRAM. The waveform data will then be sent to an external computer to be stored and analyzed.

The basic parts of the electronic design of the project include the microphone, amplifier, sample and hold chip, analog to digital converter and the microcontroller. The microphone will convert the body sound into an analog signal. That analog signal will be amplified by the microphone amplifier. The five analog signals will then be individually sampled by a sample and hold chip, and multiplexed to a single ADC to convert each analog signal to a digital signal for the microcontroller to store in memory.

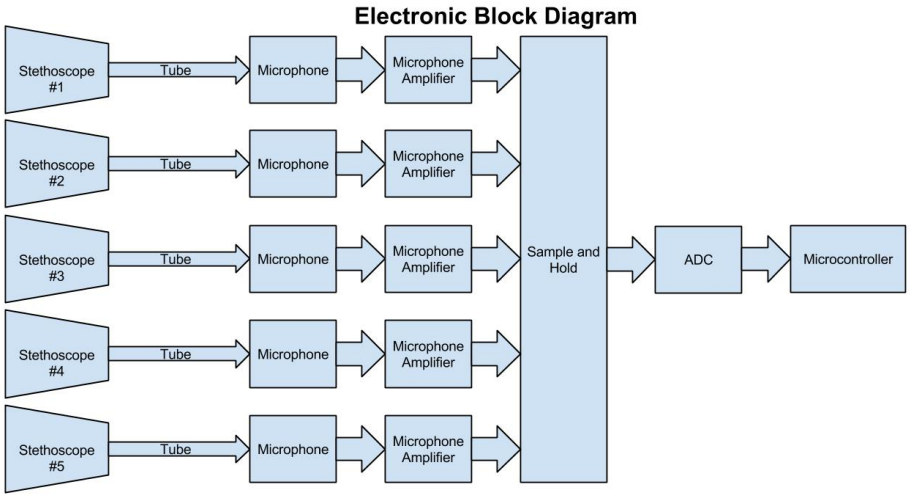


Figure 3.3 Electronic Block Diagram

3.4 Design Requirements

The design requirements for the front-end electronics in this project are shown below.

- Input waveform acquisition with no clipping
- Gather waveform data from all five stethoscope pickups simultaneously
- Sufficient Sampling Frequency
    - 2 KHz
- The body sounds are sampled synchronously from all channels.
- Acquired data is made available to an external computer for analysis
    - Communicate through USB

3.5 Component Designs

The components used for the multiple pickup digital stethoscope electronic design of the project will be described in detail below.

3.5.1 Stethoscope

Figure 3.4 Littmann Classic II S.E. Stethoscope

Manufacture: Littmann

Part Name: Littmann Classic II S.E. Stethoscope

Features

- Diaphragm Diameter 1.75” (4.4cm)
- Weight: 125 Grams
- Stainless Steel

A traditional acoustic stethoscope pickup (diaphragm and tubing) was used to collect the sounds produced by the body. Components from the Littmann Classic II S.E. Stethoscope were chosen because this stethoscope amplified heart sounds better than the other stethoscope examples that were under study for this project. The Littmann Classic II SE stethoscope is regarded as the industry standard for stethoscopes due to its high acoustic sensitivity and superior performance. (Medisave)

The sensitivity frequency response of the chosen Littman stethoscope compared to a sound pressure meter microphone pickup is shown below in Figure 3.5. The figure shows the human heart power spectra recorded from the diaphragm using both stethoscope earpieces attached to the microphone of a sound pressure meter. The control curve (labelled “Microphone”) was produced by the microphone pressed directly against the chest wall. The sounds were analyzed for sound pressure level in decibels as a function of frequency from 20 Hz to 20,000 Hz. (www.forusdocs.doc)

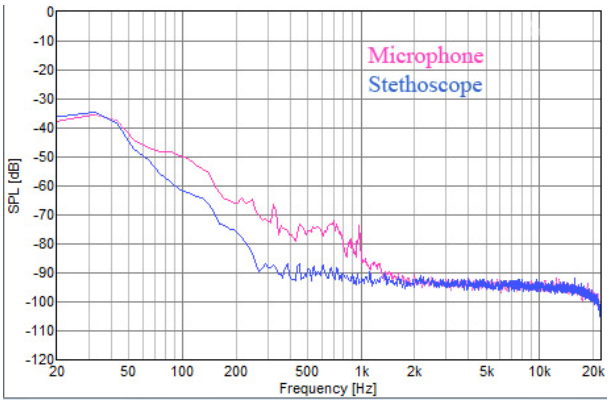


Figure 3.5 Littman Classic II SE stethoscope sensitivity frequency profile

3.5.2 Microphone

Figure 3.6 Image of the microphone used (POM-3535L-3-R)

Manufacture: PUI Audio inc.

Part Number: POM-3535L-3-R

Features

- Electrets Condenser Microphone
- Directivity: Omni
- Sensitivity: -35±4 dB
- Max Operating Voltage: 10 V_{DC}
- Current Consumption (MAX): 0.5 mA
- Impedance: 2.2 kΩ
- Diameter 6mm

The microphone is a key part of the system that converts the body sounds gathered by the stethoscope into an analog signal. An electrets condenser microphone was used to build the digital

stethoscope. Condenser microphones have a flatter frequency response and are known to be more sensitive than dynamic microphones. (MediaCollege.com) The part chosen for this project was the POM-2525L-3-R from PUI Audio Inc. The key feature of this microphone is its 6 mm diameter size; making it perfect for fitting the microphone into the rubber tube that is already coupled to the stethoscope diaphragm.

3.5.3 Amplifier

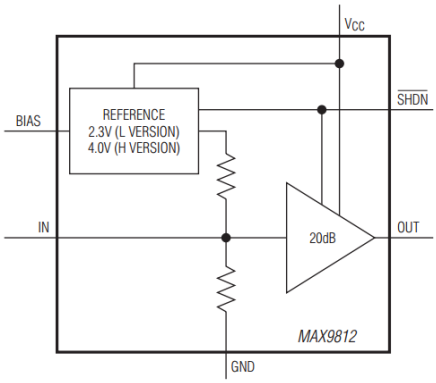


Figure 3.7 Block Diagram of the Fixed-Gain Microphone Amplifier.(MAXIM INTEGRATED)

Manufacture: Maxim Integrated

Part Number: MAX9812HEXT+T

Features

- Rail-to-Rail Outputs
- 20dB fixed Gain
- Voltage Supply Range: 4.5 to 5.5 V
- Amplifier Output Bias Voltage: 2.25 to 2.75 V
- Power Supply Rejection Ratio: 100 dB @ 217 Hz
- Small-Signal -3dB Bandwidth: 400 kHz
- Output Impedance 0.5 Ω

After the body sounds are converted into electrical signals, these signals need to be amplified. The amplifier used for this project was Maxim's MAX9812HEXT+T. This part was chosen because it is an integrated microphone amplifier designed to work with microphone transducers, and it has a fixed 20 dB

gain that is within the range required for the stethoscope signals. This part also supplies power to the microphone, has a voltage supply range of 4.5 to 5.5 volts, and includes an integrated bias. Its rail-to-rail output maximizes the useable A/D conversion range and reduces the chances of signal clipping. Such features made this part ideal to use in this project.

3.5.4 Sample and Hold

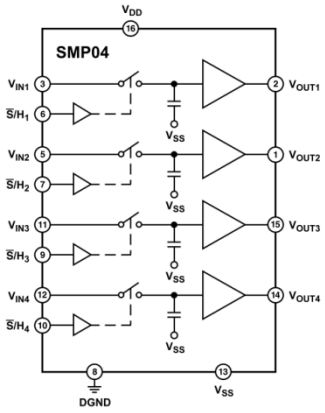


Figure 3.8 Block Diagram of the CMOS Sample-and-Hold Amplifier (SMP04)(ANALOG DEVICES)

Manufacture: Analog Devices

Part Number: SMP04

Features

- Four Independent Sample-and-Holds
- Internal Hold Capacitors
- High Accuracy: 12 Bit
- Very Low Droop Rate: 2 mV/s
- Max Acquisition Time: 4.25 µs

A sample and hold chip was added to the design to support simultaneous sampling of the five stethoscope signals. The specific part used for this project was an Analog Devices SMP04. The SMP04 contains four sample and hold circuits with a very low droop rate, and includes internal hold capacitors.

Two SMP04s were used for this project to sample the five stethoscope signals. This chip easily supported the required sampling rate of 2 KHz.

3.5.5 Analog to Digital Converter

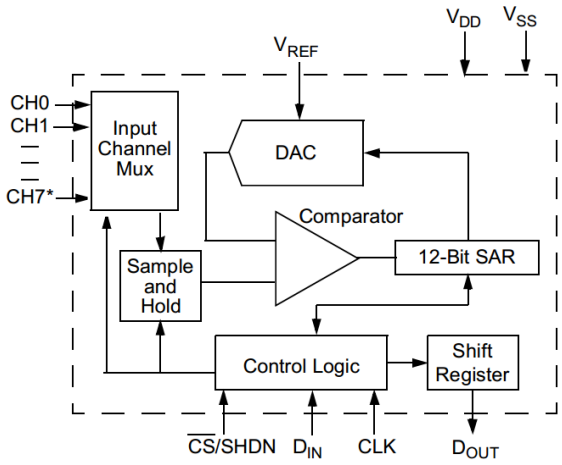


Figure 3.9 Block Diagram of the 8-Channel 12-Bit A/D Converter (MCP3208)(MICROCHIP)

Manufacture: Microchip

Part Number: MCP3208

Features

- Successive Approximation ADC
- 12-bit resolution
- ±1 LSB max differential non-linearity
- 8 multiplexed analog input channels
- Single-ended analog input
- SPI Serial interface
- Allowed Single Supply Operation: 2.7V to 5.5V
- 100k samples per second max. sampling rate at V_{DD} = 5V

A multiplexed-input Analog to Digital Converter was used to convert the analog input waveforms to digital signals. This device provides 12 bit precision with a maximum sampling rate up to 100 kilo-samples per second when used with a 5V supply voltage; which was more than adequate to support the required

sampling frequency of 2 KHz. In addition this part has an SPI Serial interface to transfer the digital signal values gathered to the Microcontroller.

3.5.6 Microcontroller

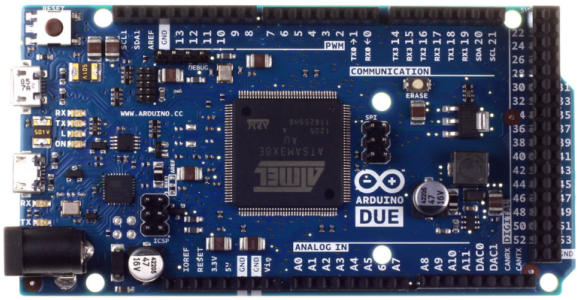


Figure 3.10 Arduino Due(Arduino)

Features

- ARM Cortex®-M3 revision 2.0 running at up to 84 MHZ
- Operating Voltage: 3.3V
- 512 KB Flash
- 96 KB SRAM
- SPI Communication Port
- 4 USART
- 2 DAC

The Arduino Due was used for this project because of its extensive features such as having built-in SPI and UART communication ports, multiple digital outputs, and the ability to store 96 KB of data. It was also selected because of its compactness and ease of firmware development. The functions carried out by the Arduino Due in this project were control of the sample and hold chip and the analog to digital converter, along with collecting and storing data in real time, and sending the data to the external computer for analysis. It was important to note that the Arduino Due was used instead of the Arduino Uno because it had 96 KB SRAM compared to the 2 KB SRAM on the Uno. The larger SRAM on the Arduino DUE allowed storage of 4.75 seconds worth of data from all five channels at a sampling rate of 2 KHz.

3.6 Acquisition Electronic Design

Using the parts described above, an acquisition circuit was designed to simultaneously acquire body sounds and convert them into digital values that were stored in the microcontroller. After much design, integration, testing and refinement, the final circuit built for this project is shown in Figure 3.11. This circuit was first tested on a breadboard and then wired on a perforated proto board, as seen in Figure 22. As this project is still in the proof of concept stage, and the signal frequencies were relatively low, it was not necessary to build a printed circuit board for the acquisition hardware.

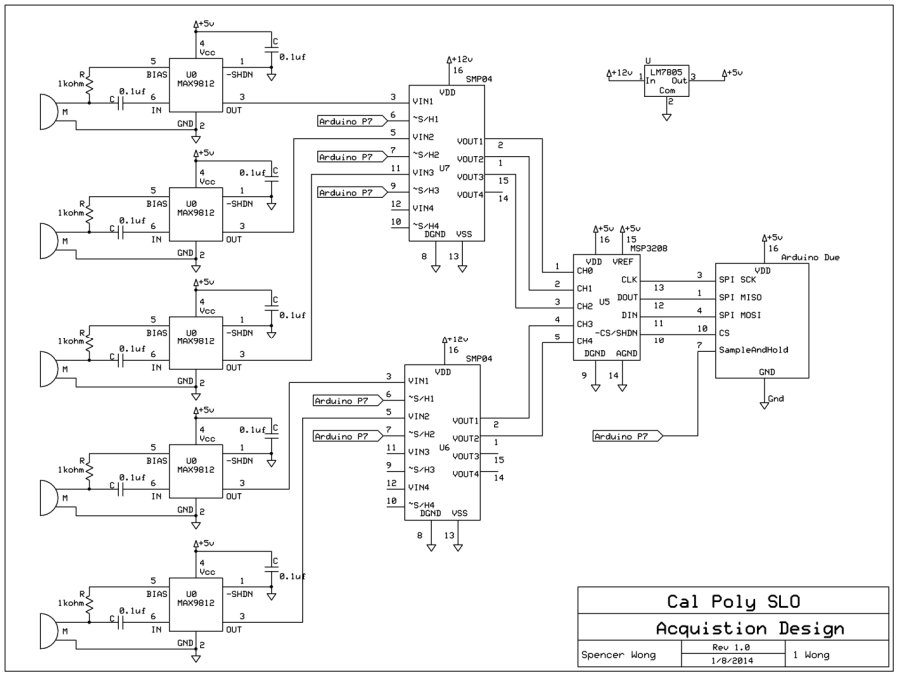


Figure 3.11 Acquisition Circuit Design

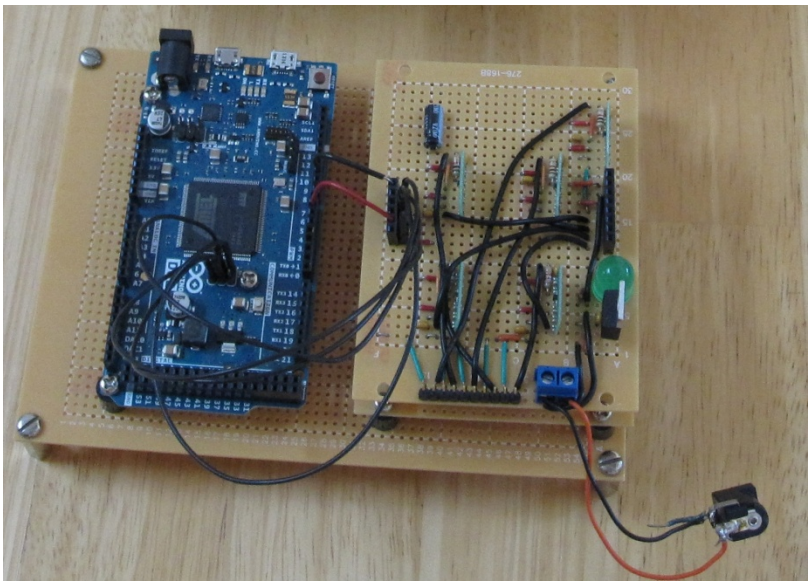


Figure 3.12 – Physical Design of the Acquisition Hardware

3.7 Embedded Software Design

Embedded software was designed for the microcontroller to control the data acquisition process. In the hardware design, there are five sample and hold stages and an analog to digital converter that were used to collect waveform data. In addition, communication to an external computer is required to store the data for later analysis and off-line processing. Figure 3.13 shows the general flowchart and Figure 3.14 is shows how data is collected in more detail. The full microcontroller code is shown in Appendix E.

The software flowchart shows the overall operations that the microcontroller is constantly performing. As shown in Figure 3.13, the microcontroller waits for a command from the computer to start collecting data; and after 4 seconds the sound data is collected and sent to the computer to be stored. It takes about 10 seconds for all the data to be transferred over to the computer. After transferring the data, it then waits until commanded again to collect data.

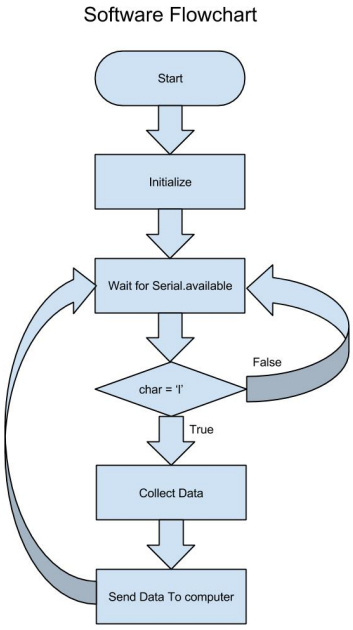


Figure 3.13 Software Flowchart

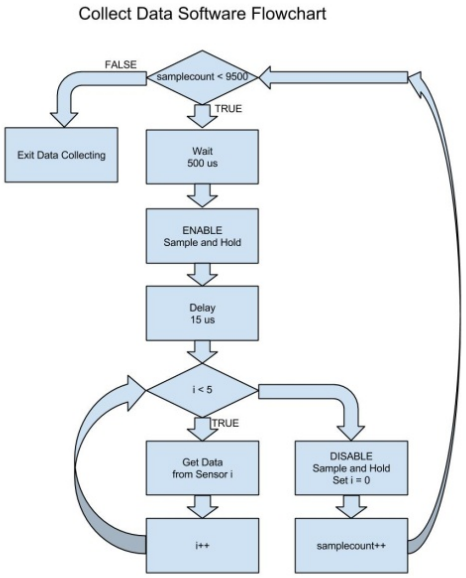


Figure 3.14 Collect Data Software Block Diagram

3.7.1 Collect Data code

The data collection code was one of the more challenging parts of the software. This section of the code had to interface with the sample and hold chips and the analog to digital converter and coordinate the timing of waveform samples and data conversions. Interfacing with the sample and hold chip was relatively easy; requiring simply setting a single bit low to hold the signal and then high to stop holding the signal. Interfacing with the ADC (MCP3208) required setting up an SPI communication channel. Fortunately the microcontroller used was equipped with an SPI configured communication port, so this made the interfacing simple. More detail in interfacing the microcontroller with the ADC will be covered below.

Figure 3.14 shows the flowchart of the data collection software. This code initiates when the microcontroller receives the command to collect data. Once it has received this command, it enables the sample and hold chips to simultaneously hold their present signal levels; thus effecting sampling of all

channels synchronously. Then the code sequentially digitizes all five held sensor signals consecutively; by selecting each signal's input channel and initiating a conversion of the A/D converter. Once the code finishes collecting the data, the sample and hold chips are commanded to stop holding the data. Then, the remaining time in the 500 microsecond sampling interval is waited, before the process is repeated again. This process is completed when 9,500 samples are collected from each sensor at a sampling rate of 2 KHz; producing about 4 seconds of data per channel.

3.7.2 Communication with ADC

Communication with the ADC is accomplished using the SPI protocol. Figure 3.15 shows an example of the interface signal sequence details for how the microcontroller communicates with the ADC. Table 3.1 shows the commands used to collect the data from the five sensors.

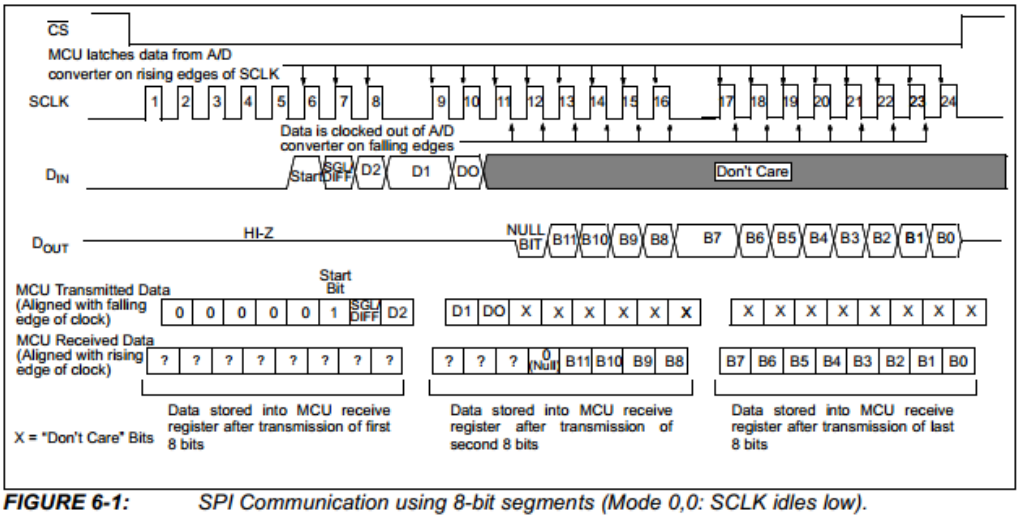


FIGURE 6-1: *SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).*

Figure 3.15 MCP3208 SPI Communication Example (MICROCHIP)

	Sensor Command 1	Sensor Command 2	Sensor Command 3
Sensor 0	0x06	0x00	0x00
Sensor 1	0x06	0x40	0x00
Sensor 2	0x06	0x80	0x00
Sensor 3	0x06	0xC0	0x00
Sensor 4	0x07	0x00	0x00

Table 3.1 MCP3208 Transmitted Data

To collect data from the ADC, one of the sequences of commands listed in Table 3.1 must be

sent to the ADC. For example to receive the digital value of sensor 1's signal, the chip enable needs to be set high; and then the command sequence 0x06, 0x40, and 0x00 needs to be sent to the ADC. After these commands are sent, the chip enable signal needs to be sent low again. In the process of sending the commands, the ADC is sending serial data back, beginning part way through the second command byte. Within the received data is the digitized value of the sensor signal, which needs to be extracted by parsing. This process needs to be repeated for all of the sensor signals, using the appropriate code sequence for each different channel.

3.8 Mechanical Hardware Design

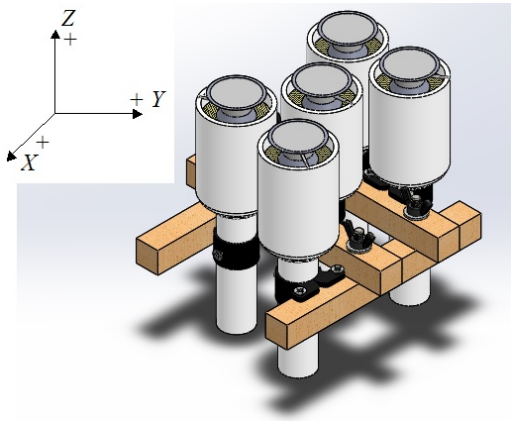


Figure 3.16 Side view of the Stethoscope Apparatus

One of the problems faced in this project was the need to hold all five stethoscope pickups firmly against the subject's chest to produce good acoustic signal coupling. It was also highly desirable to know the exact coordinates of each stethoscope pickup in order to simplify geometric delay estimation for beam-steered, directed listening of specific areas of the heart. The apparatus shown in Figure 3.16 was the solution to the first challenge. The top left side of this Figure shows the direction of the x, y and z coordinates. The plane formed by the x and y axis is parallel with the stethoscope diaphragms, and the z axis is the depth direction for the stethoscope. This apparatus was designed to be simple, sturdy, and to hold all 5 stethoscope pickups on an uneven surface in close vicinity of each other. The shortcoming of this solution is that it is difficult to know the exact Z position of each pickup, as it changes to conform to the body of the subject being examined. More work is needed to provide a method for knowing the exact

Z position of each sensor. However, this design does achieve the more critical goal of allowing every stethoscope diaphragm to make good contact with the body. In addition, the exact relative positions of the five stethoscopes in the X and Y plane is known and firmly constrained.

3.8.1 Mechanical Frame

As seen in Figure 3.16, each stethoscope pickup was surrounded by a piece of PVC pipe to hold it in place. A simple 4-piece wooden frame was built to locate and hold the PVC pipes in a known x-y placement. The frame was constructed to have the stethoscopes placed as close together as possible. This was important because the farther away the stethoscopes were from the desired auscultation point, the weaker the received heart sounds would be. Shown below in Figure 3.18 and Figure 3.18 are the details of the frame that was built.

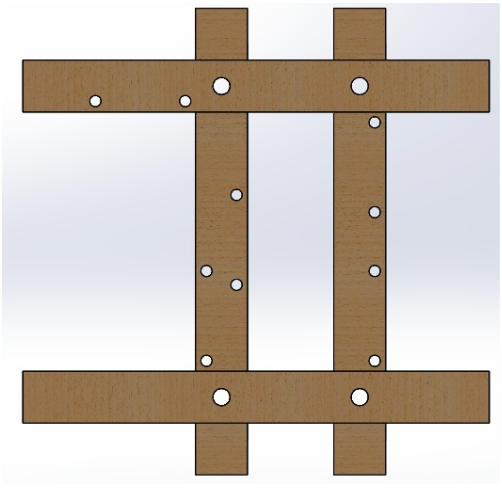


Figure 3.17 Top view of the Stethoscope Apparatus Frame

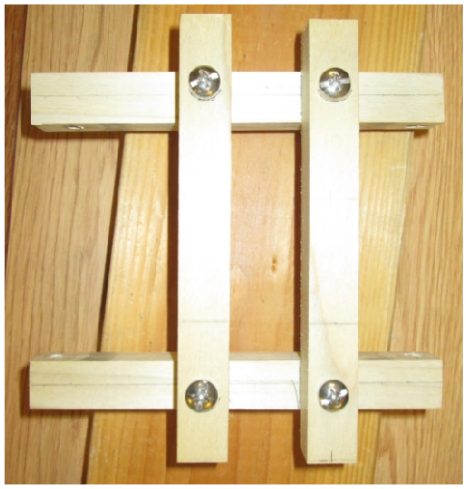


Figure 3.18 Completed Top view of the Stethoscope Apparatus Frame

3.8.2 Stethoscope Holders

The stethoscope holder was one of the more challenging problems in the mechanical design. The stethoscope holder had to provide a conformable z-axis position with a mild downward pressure for each stethoscope, so that when the stethoscope apparatus was strapped to the patient each stethoscope would conform to the contours of the body and still make firm contact with the chest wall. Also, the

integrity of the stethoscope pieces had to be protected to preserve their acoustic sensitivity and because they were relatively expensive. The solution to this problem is shown below in Figure 3.19. The solution consisted of PVC pipes, sponges, strings and the stethoscopes.

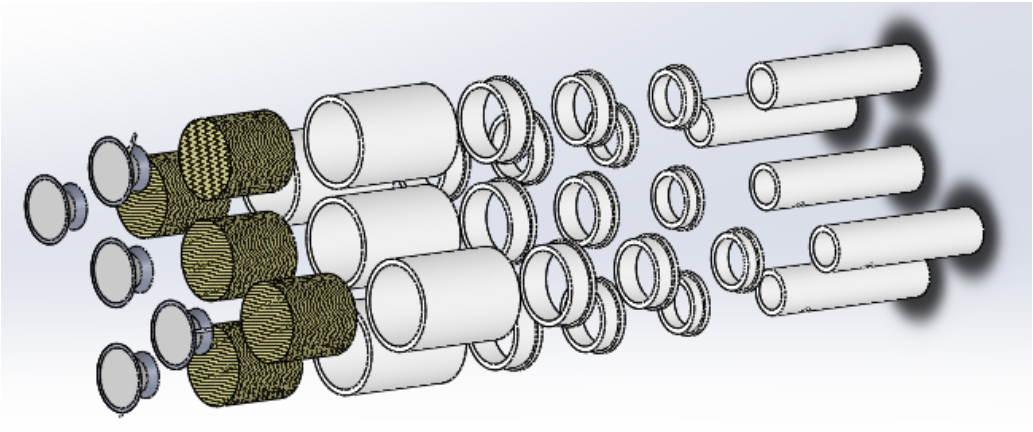


Figure 3.19 Exploded view of the Stethoscope Holders

Shown above in Figure 3.19 is the stethoscope holder design. The bell of each stethoscope head was pressed against a sponge, in order to give the stethoscope flexibility to articulate in the z direction and to tilt slightly to conform to the shape of the patient's chest. The sponge also provided a spring force pressing each stethoscope diaphragm to the maximum z-direction position possible. The spring force applied was controlled by the amount of sponge used. The sponge was then enclosed inside a PVC pipe that was a little bit larger than the diameter of the stethoscope. This bound the sponge and gave structural support for the stethoscope diaphragm to be secured. With the sponge sandwiched together with the stethoscope inside the PVC Pipe, the apparatus allows each stethoscope the flexibility to move independently in the Z direction.

Holes were drilled into the PVC pipe so that the stethoscope could be secured to it using wires. Two opposing wires, tied as shown in Figure 3.20, constrained the x-y position of the stethoscope neck at its narrowest diameter, while still allowing limited z-axis articulation. Figure 3.21 shows the actual stethoscope with the wires in place. Slots cut in the PVC pipe prevented the flexible tubing of each stethoscope from interfering with the z-axis motion of the diaphragm.

Narrower diameter PVC tubing was coupled to the stethoscope-holding PVC pipe segments using multiple PVC pipe size adapters, as shown in Figure 3.19. Together these created a sturdy mounting and locating fixture for each stethoscope.

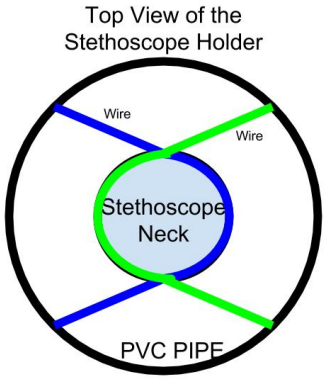


Figure 3.20 View of Stethoscope View

Figure 3.21 Stethoscope Holder

3.8.3 Apparatus Integration

Assembly and integration of the multiple stethoscope apparatus was relatively simple. The five individual stethoscope holders were secured to the wooden locating frame by clamps, as shown in Figure 3.22. Note that a screw was drilled through the PVC pipe holder and the PVC to keep the PVC pipe from sliding. The final end product of this design is shown in Figure 3.23.

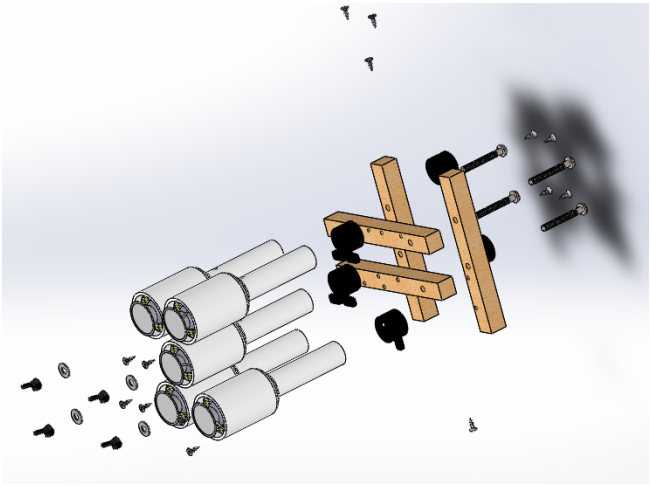


Figure 3.22 view of the Stethoscope Apparatus

Figure 3.23 Completed Stethoscope Apparatus

3.8.4 Stethoscope to Microphone Acoustic Coupling

The sound gathered by the stethoscope was directed to the microphone by a segment of the standard rubber tubing included with the stethoscopes. An electret microphone was then securely fitted into each of the rubber tubes. The tubing length between the stethoscope chest piece port and the microphones for all five stethoscopes was made as nearly identical as possible, so that the sound propagation time from each diaphragm to its corresponding microphone was the same; thus introducing one less variable in the beamforming time delay estimates computed later.

3.8.5 Stethoscope Position

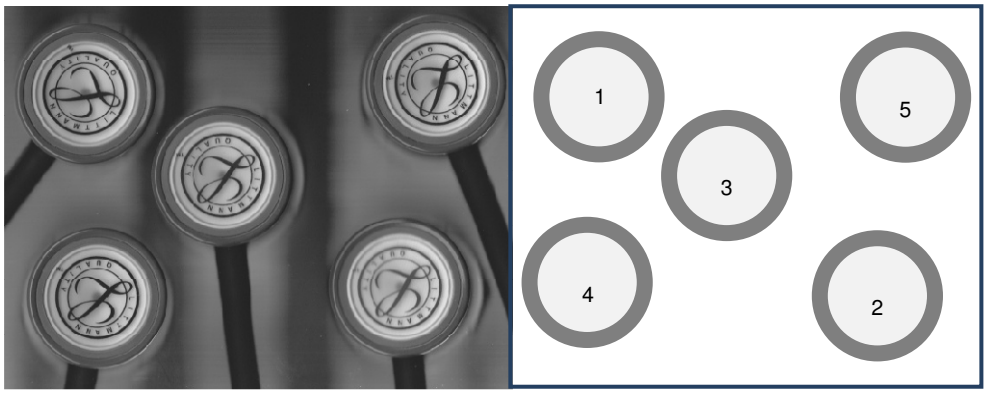


Figure 3.24 Image of the stethoscope apparatus faced down on a scanner, with channel numbers shown

Knowing the relative spatial positions of the five stethoscope diaphragms is mandatory for geometric beamforming. Therefore precise measurements were needed to determine the location of each stethoscope once mounted into the multiple stethoscope apparatus. Standard mechanical measuring methods for determining the position of the stethoscopes relative to each other seemed tedious and potentially hazardous to the valuable stethoscope pieces. Instead, an imaging process was developed to greatly simplify this task.

A MATLAB script was written to analyze the digital image received from a standard photo/document scanner when the stethoscope apparatus was scanned. An example digital image is shown above in Figure 3.24. The script used the MATLAB function '*imfindcircle*' to find the circular stethoscope diaphragms in the image. The '*ObjectPolarity*' and '*Sensitivity*' function parameters have to be given the appropriate settings for this function to work properly. With the correct parameters, the function will output the center location of any circles found by the function. With this information, a reference point is chosen and the circle center locations are determined and plotted. The result of this script is shown below in Figure 3.25 and Figure 3.26. The MATLAB script used can be found in Appendix O.

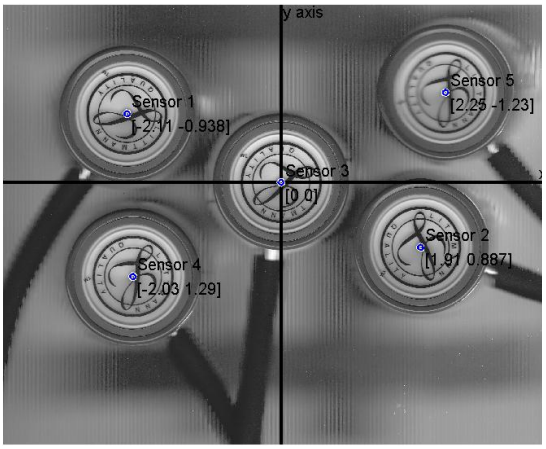


Figure 3.25 Coordinates of each stethoscope relative to sensor 3 in Inches.

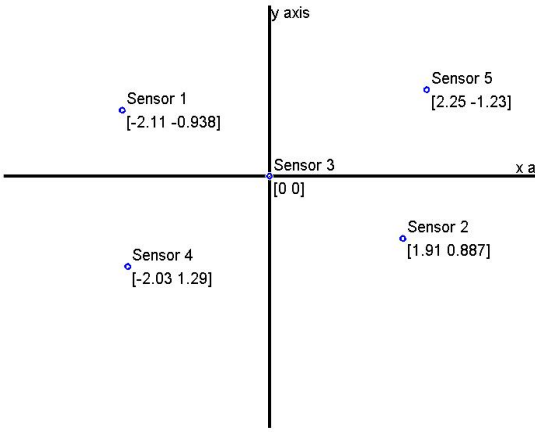


Figure 3.26 Coordinates of each stethoscope relative to sensor 3 without the stethoscope image.

4 Characterization and Verification

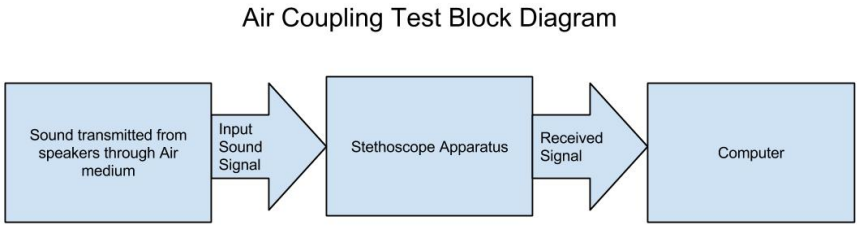


Figure 4.1 Air Coupling Test Block Diagram

Two *ex vivo* tests were performed on the stethoscope apparatus to characterize and verify the project system design. The tests performed used air as the medium between the acoustic source (signal generator output speakers) and the device under test (stethoscope apparatus), so the sound coupling into the stethoscopes is different than when used *in vivo*. The frequency-dependent signal attenuation that occurs for wave propagation in human tissue will not be experienced in the test setup, so that the frequency characteristics of the stethoscope pickup can be more accurately measured.

The block diagram of the test configuration is shown above in Figure 4.1, where the Stethoscope apparatus is the device under test. The basic test setup is shown in Figure 4.2; where the speaker is on the left side of the image and the multiple stethoscope apparatus is on the right side with the electronic hardware and the computer to command and received data. Details of the procedures and MATLAB scripts used in the three system characterization tests can be found in the Appendix I.

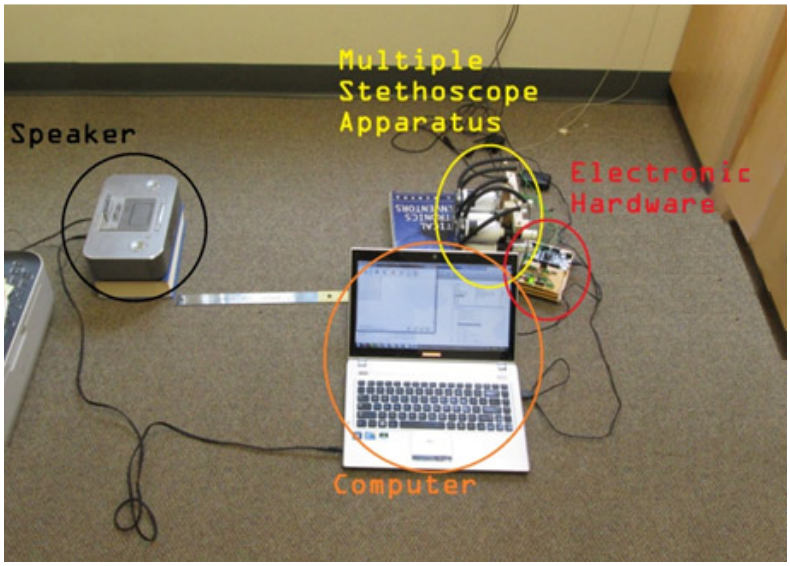


Figure 4.2 Characterization and Verification Test Setup

The two tests that were performed to characterize the transient and frequency responses of the system were the impulse and chirp signal tests; distinguished by the source signal used to measure the system response. The impulse test was performed to determine if there were any significant time delay differences between the five input channels due to differences in lengths of the tubes used in the stethoscope apparatus. The chirp tests measured the frequency response of the stethoscope apparatus. Again, it should be noted that the acoustic sensitivity frequency response of the stethoscope pickup varies depending on the contact pressure that the stethoscope applies to skin. This contact pressure is very hard to control, and so no pressure was applied (or needed for the air coupling of the test configuration) to provide a uniform testing condition for all pickups. Although we neglect any frequency response differences of the stethoscopes when sampling body sounds, it is still relevant to verify the frequency response similarity of the stethoscope input channels under these controlled conditions.

4.1 Impulse Experiment

The purpose of the impulse experiment was to determine if there was any variation in lengths of the tubes that connect the stethoscopes to their associated microphone, leading to a difference in arrival

times for acoustic signals received in each input channel. This was done by transmitting a sound impulse from a speaker that was positioned a foot away from the stethoscope apparatus. With the width of the speaker, the spacing of the stethoscope pickups and the distance between the speaker and the stethoscope apparatus, the impulsive acoustic wave front should be close to being planar and perpendicular to the x-y plane of the apparatus. The five received signals from the microphones were observed and compared to determine if there were any phase differences between the signals. The signal generated for this test is shown below in Figure 4.3. The signal received by each microphone will be more spread out in time due to the finite frequency response of the amplifier and speakers producing the test signal.

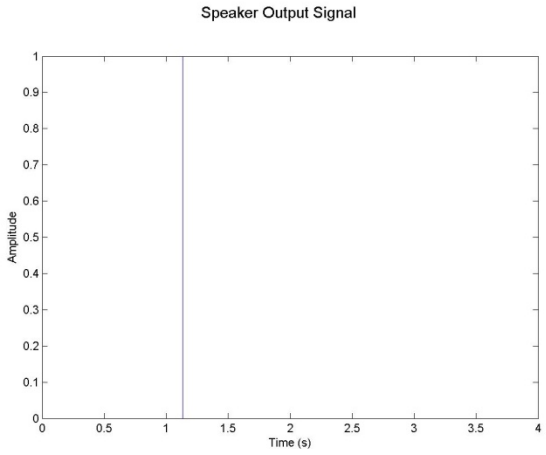


Figure 4.3 Signal generated for impulse test

4.1.1 Impulse Test Results

The received signal results from this test are shown below in Figure 4.4, Figure 4.5, and Figure 4.6. As seen in Figure 4.4, the transient responses of the input channels are similar in shape, but have different amplitudes, indicating a difference in overall gain or in the high-frequency sensitivity of the input channels. A close up view of the peaks is shown in Figure 4.5and Figure 4.6 shown below.

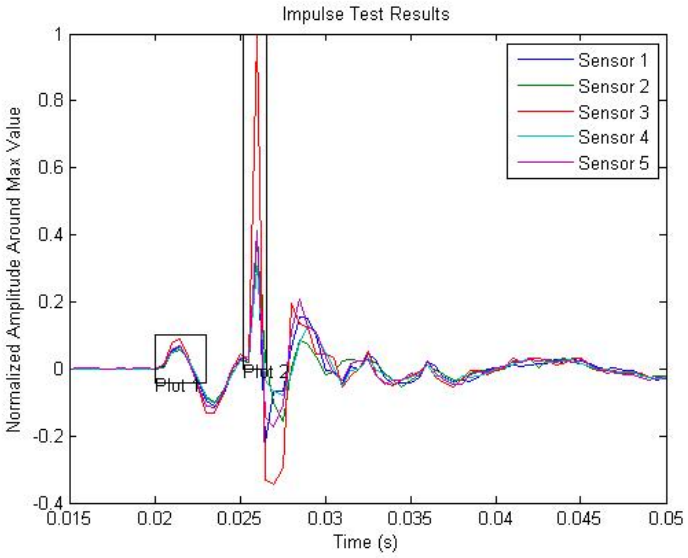


Figure 4.4 Impulse Response Result

The first peak in the received waveform, a preamble to the main impulse peak, is shown in Figure 4.5. Note that the signals peak at the same time sample for all five input channels. The second peak shown in Figure 4.6 is the main lobe of the impulse sound created by the speaker. This figure shows that the shapes of the signals are similar, and that they again peak at the same time sample, meaning that there is not a significantly large phase delay difference between channels. The only observed difference between the signals is the amplitude of the signals. It is also important to note that the third sensor has significantly higher peak amplitude than the other sensors.

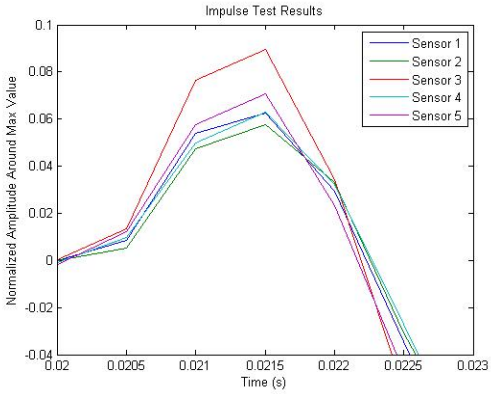


Figure 4.5 Plot 1 of the Impulse Response Results 1st peak

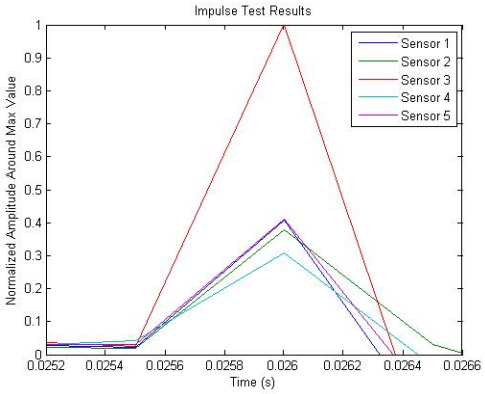


Figure 4.6 Plot 2 of the Impulse response result 2nd peak

To show how similar the impulse responses are in each channel, correlation coefficients were computed for the received impulse signals from all possible sensor pairs. These results are shown in Table 4.1. The smallest correlation coefficient is 0.84 and the largest correlation coefficient is 0.95. This shows that the signals were very similar to each other as a correlation coefficient of 1.0 would indicate that they are exactly identical.

Correlation Coefficient of the Impulse Response					
	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5
Sensor 1	1.000	0.840	0.902	0.909	0.950
Sensor 2	0.840	1.000	0.879	0.912	0.882
Sensor 3	0.902	0.879	1.000	0.848	0.921
Sensor 4	0.909	0.912	0.848	1.000	0.917
Sensor 5	0.950	0.882	0.921	0.917	1.000

Table 4.1 Correlation Coefficient for the data receive in the impulse response.

To show how the peak values from the five sensors varied, the peak values for both the preamble and the main peaks were normalized to a reference sensor's values (3rd sensor). Table 4.2 below compares both the peak values and the normalized peak values (ratio in comparison to the peak of the 3rd sensor) for each sensor impulse signal. As shown in the table, sensor 3 consistently has a higher peak than the other sensors for both peak 1 and peak 2. Also, the peak 2 amplitude of sensor 3 was nearly three times the size of the lowest sensor peak. This confirms the importance of the frequency response analysis on the stethoscope.

Max Peak Value Analysis				
	1st Peak Value Normalized around Max Value	2nd Peak Value Normalized around Max Value	1st Peak Value / Sensor 3 1st Peak	2nd Peak Value / Sensor 3 2nd Peak
Sensor 1	71.716	465.716	0.702	0.407
Sensor 2	65.982	432.982	0.646	0.379
Sensor 3	102.197	1143.197	1.000	1.000
Sensor 4	71.922	351.922	0.704	0.308
Sensor 5	80.719	470.719	0.790	0.412

Table 4.2 Shows the peak values and ratio versus sensor 1 of the signals received from the 5 sensors

The data gathered has shown that the phase delays in the five sensor channels are roughly the same (within the resolution of one sample period); but it is unlikely that they are exactly the same as the distances from the diaphragms to the microphones cannot be exactly identical. To get a more accurate measurement of the phase delay between each channel, a 'spline' interpolation was applied to the

normalized impulse test data with an up sampling by a factor of 300, resulting in a sampling frequency of 600 KHz. Below in Figure 4.7 and Figure 4.8 are the results of the interpolation. Figure 4.8 shows a close up view of the normalized peaks for the primary impulse peak 2. As could not be distinguished in Figure 4.6 due to the low temporal resolution, Figure 4.8 shows more clearly that there were small differences in phase delay for each of the signal channels. The numerical values for the different phase delays in each channel determined by this method, expressed as relative time delays, are given in Table 4.3.

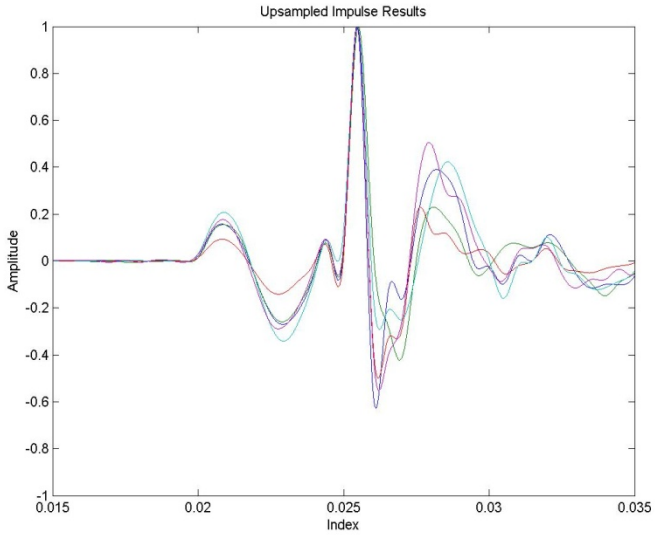


Figure 4.7 Upsample Impulse Result

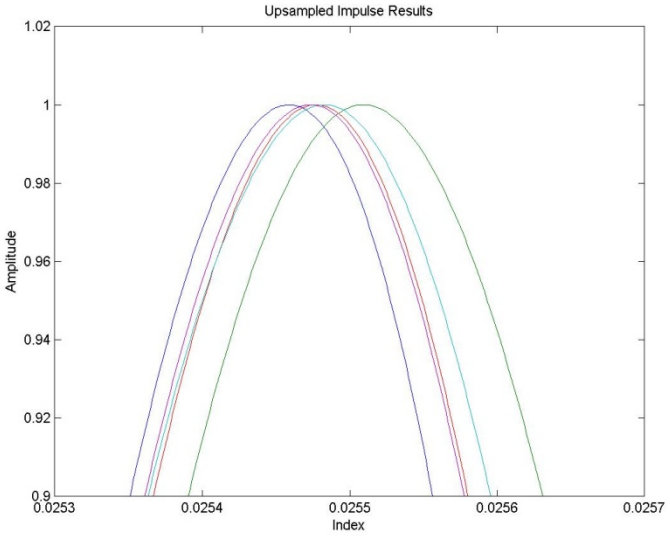


Figure 4.8 Peak of Upsample Impulse Result

	Relative Channel Arrival Time Delay (seconds)				
	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5
Phase Delay in Reference to Sensor 3	-18.3×10^{-6}	31.7×10^{-6}	0	6.67×10^{-6}	-3.3×10^{-6}

Table 4.3 Relative Channel Arrival Time Delay

4.2 Chirp Experiment – Frequency Response

The chirp characterization is one method for determining the frequency response of the stethoscope apparatus. This is accomplished by outputting a chirp signal (increasing frequency sinusoidal signal with a constant amplitude) from the speakers. A linear increase in frequency from 0 to 1 KHz over an interval of 4.5 seconds was used. The chirp signal generated is shown below in Figure 4.11 along with the frequency vs. time (Figure 4.9) and amplitude vs. time (Figure 4.10) of the chirp generated.

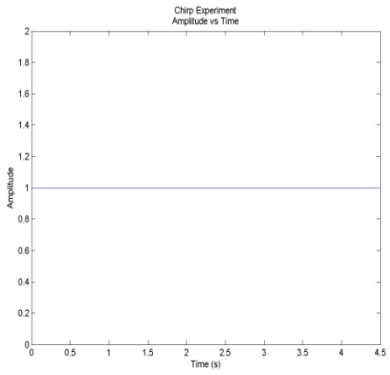


Figure 4.9 Chirp Characterization Amplitude vs Time

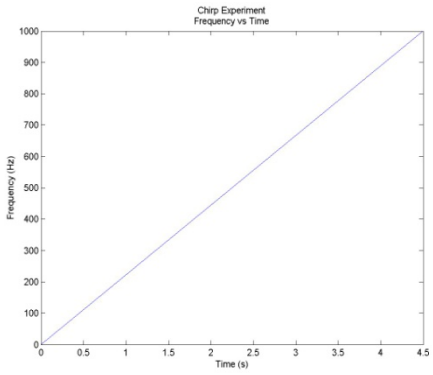


Figure 4.10 Chirp Characterization Frequency vs Time

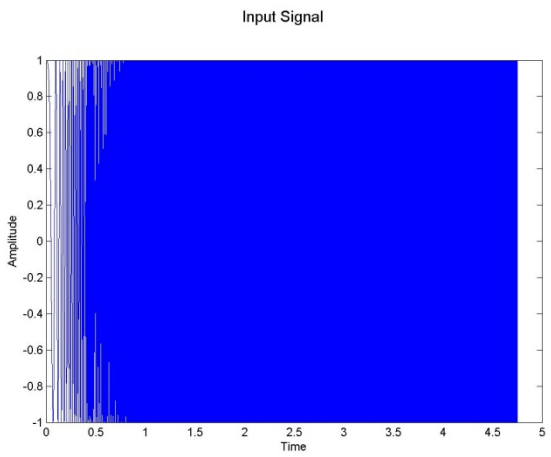


Figure 4.11 Chirp Test Input Signal

The procedure and scripts for this experiment are again shown in Appendix I. The test results showed below required some post processing to arrive at the frequency response, as described below. Several repetitions were also run on this experiment to get a more accurate measurement of the system response.

4.2.1 Post Processing

To achieve consistent results for the frequency response from the chirp test, some post processing had to be performed on the data collected. The shape of the magnitude response of the system is given by the amplitude of the envelope of the received signal, plotted against the frequency of the chirp signal for the corresponding time instant measured. The first signal processing step was to align the data received so that the waveforms from all trials started at the same time. This was needed because there was a random delay in the external computer in starting the test sound. The next step was to remove any DC offset from the data collected. This was done by subtracting the mean signal value for a complete sweep from the individual sample values in each waveform record. Next, a Hilbert Transform of the data was used to find the analytic envelop of each signal. The Hilbert Transform is a method often used in ultrasound signal processing for medical imaging to find the envelope of a sinusoidal signal. The next step was to remove some residual variations in the envelop using a 10-point convolution averaging

of the data around each envelop point. Finally, the envelopes from the multiple repeated sets of data for each channel were averaged together to further reduce noise and variation on the results. An example of the results of this processing is shown in the diagram shown below in Figure 4.12 to Figure 4.16. For more detail please refer to Appendix L.

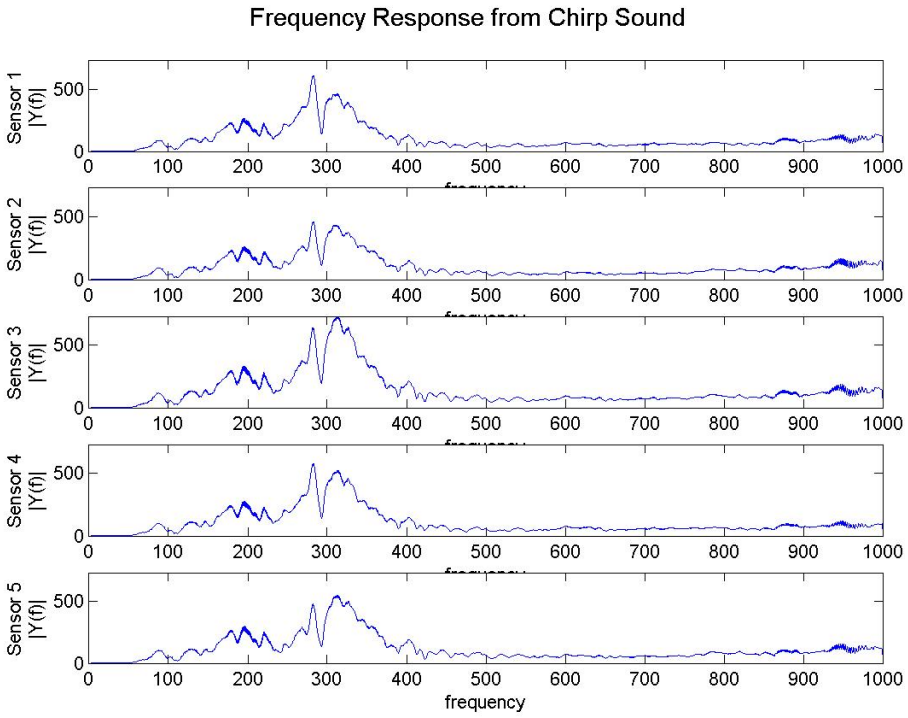


Figure 4.12 Frequency Response of the 5 Sensors from Chirp Sound

4.2.2 Frequency Response Results

Once the post processing was completed, the five frequency response plots were overlapped so that they could be more easily compared, as shown in Figure 4.13. The full frequency range of interest for this system is between 0 Hz to 1000 Hz (Nyquist rate). As you can see in Figure 4.13, the magnitude responses of all the sensors have similar shapes, but with different amplitudes. Three segments of this diagram have been zoomed in for a closer view of how the responses vary in different frequency bands around the three primary peaks in the magnitude responses (Figure 4.14 to Figure 4.16).

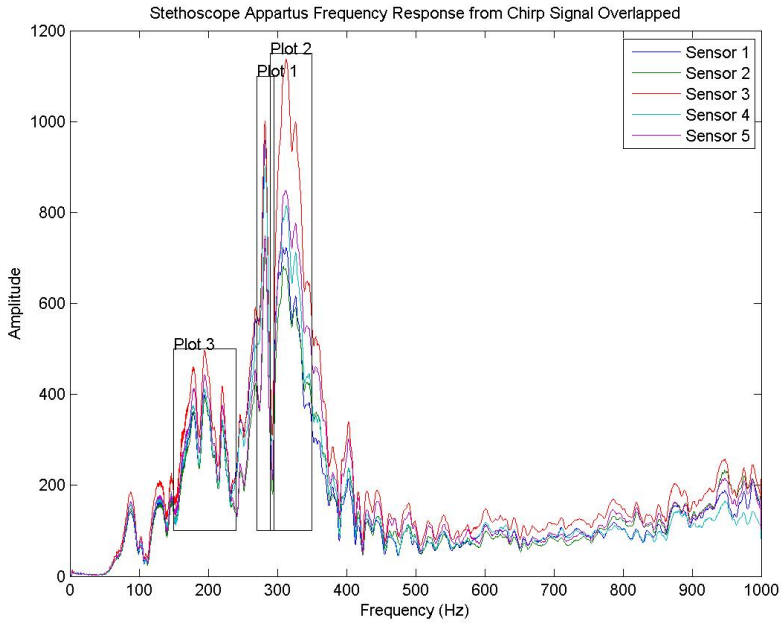


Figure 4.13 Frequency Response of the 5 stethoscope overlapped

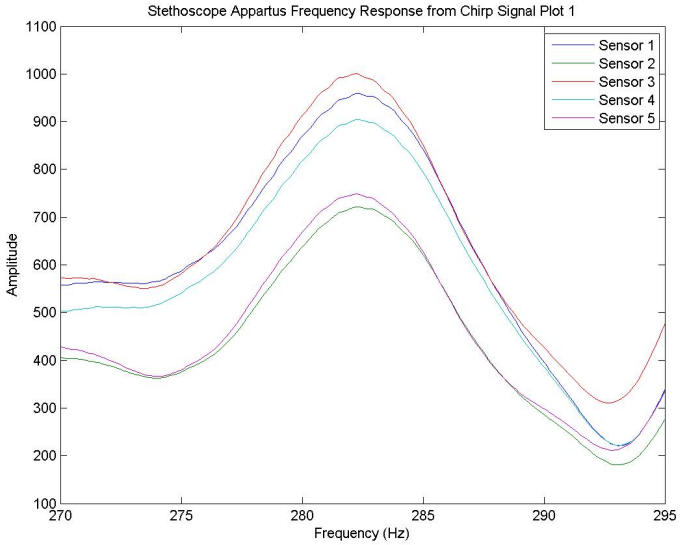


Figure 4.14 Close up view of Plot 1 from the Frequency Response

Shown above is a zoomed in view of the second peak of the frequency response, just less than 300 Hz. This shows the different sensor channels to have very similar response shapes between the frequencies of 270 Hz to 295 Hz. The ratio of the smallest peak to the largest peak in this range is 0.74.

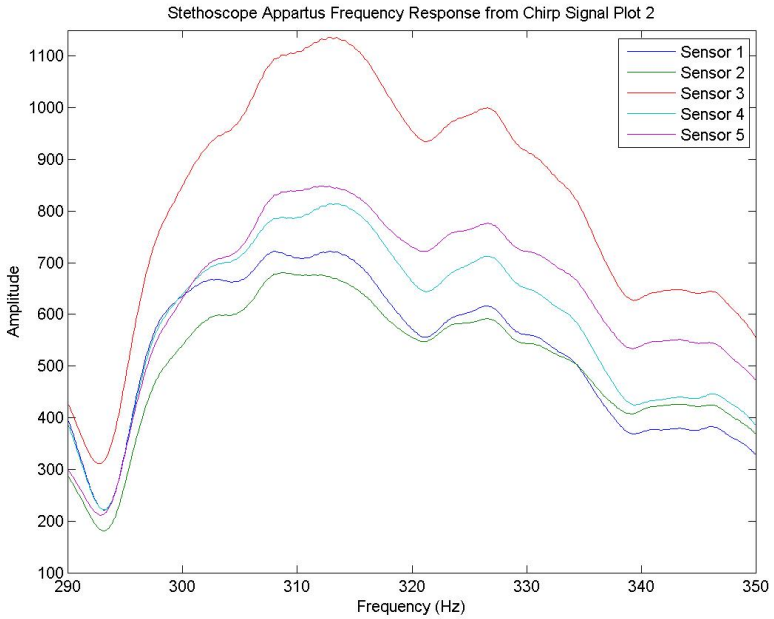


Figure 4.15 Close up view of Plot 2 from the Frequency Response

Figure 4.15 shown above is the zoomed in view of the largest peak in the chirp frequency response. It shows the frequency range between 290 Hz to 350 Hz. Again the responses of the different channels have a very similar shape but with different amplitudes. Sensor 3 again has significantly higher amplitude than the other channels, as was also the case with the impulse transient test. The ratio of the smallest peak to the largest peak is 0.59 at about 310 Hz.

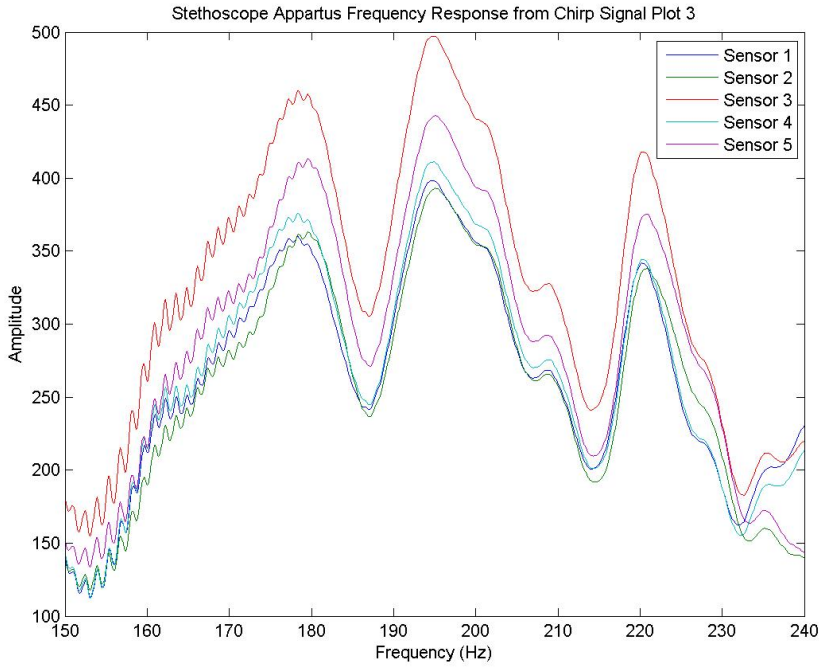


Figure 4.16 Close up view of Plot 3 from the Frequency Response

Figure 4.16 above shows the 3rd highest peak. This curve shows the lower frequency response between 150 Hz to 240 Hz. Again all channels seem to have a very similar shape in this range as well.

An additional narrower-range chirp test was done to get a more accurate understanding of the lower frequency response of the stethoscope. This was done because the heart sounds of interest are strongest in the frequency range between 20 to 100 Hz. Below in Figure 4.17 are the results of the narrow-range test, found using the same post processing method described above. Figure 4.18 shows the frequency responses for each of the five channels overlapped.

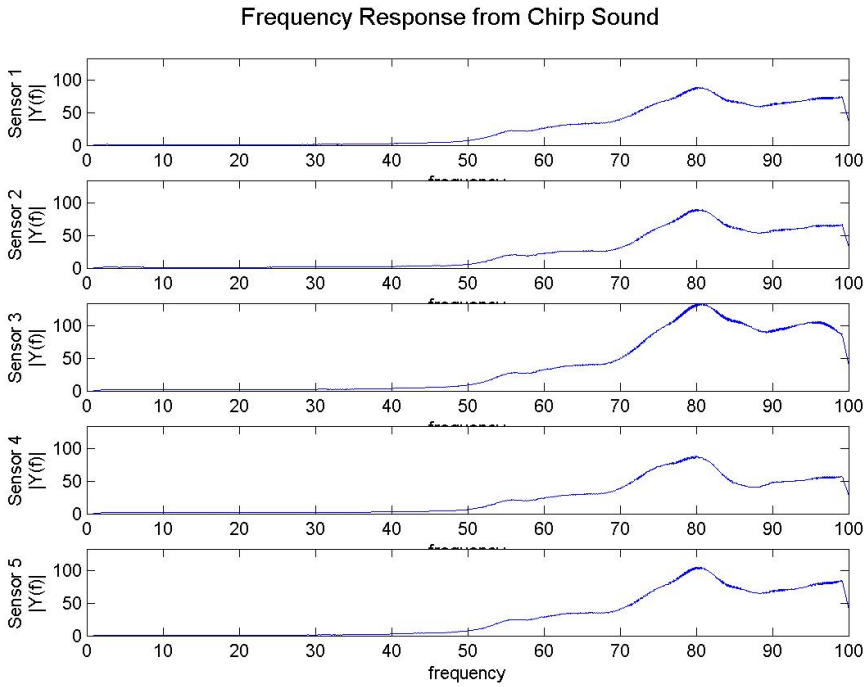


Figure 4.17 Frequency Response at a Lower Frequency

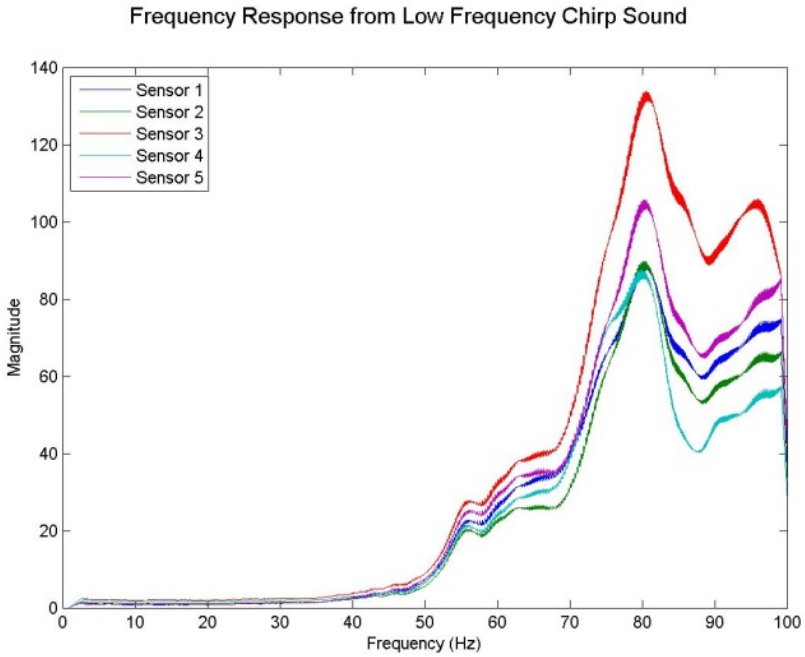


Figure 4.18 Frequency Response at a lower frequency overlapped

In Figure 4.18 the low frequency magnitude responses, all five sensors again have similar shapes, except for some peaking in the response around 95 Hz in Sensor 3. For frequencies between 0 and 50 Hz, there is very little signal measured in any of the channels. This is likely more of an issue with the frequency characteristics of the speaker producing the chirp sounds, as it is very hard to output such low frequency tones without a sub-woofer or very large speaker.

All of the chirp results show differences in overall sensitivity (gain) in each sensor channel, with Sensor 3 (the center sensor) always having the strongest signal. This most likely indicates that either or both the speaker producing the chirp sound and / or the stethoscope diaphragm receiving the sounds have angularly-dependent sensitivity (directivity pattern). Typical of acoustic signals, the strength of the sound wave reaching a sensor that is not directly in front of the speaker (an off-axis sensor, like sensors 1, 2, 4, & 5) will be weaker than the signal received on-axis (directly in front of the speaker – as with sensor 3 in the center of the apparatus) due to more absorption and dissipation along the greater path length to the outer sensors. Similarly, each stethoscope's acoustic pickup (diaphragm & bell) will be vibrated more in the perpendicular direction (z axis) by a pressure wave reaching it head-on than by a wave arriving at an oblique angle. Therefore, the strength of the sensed signal will vary with its angle of incidence, producing dependence in sensitivity on the wave front direction.

4.3 In-Vivo Verification of Multiple Stethoscope Apparatus

The previous experiments verified that the multiple stethoscope apparatus was able to properly receive signals generated from a speaker. The next step was to verify how well the stethoscope apparatus receives signals from the body. This was accomplished by the following procedures:

First, the stethoscope apparatus is connected to the electronics system, and the microcontroller board is connected to the external computer; as was done for the previous system characterizations. Then the stethoscope apparatus is secured to the test subject's body, with the stethoscope diaphragms located near to the typical auscultation listening points for heart sounds on the chest. (See Figure 1.8). The apparatus is secured in place to prevent lateral movements and to ensure there is downward

pressure on each diaphragm using a series of elastic straps shown in Figure 4.19. Once the apparatus is in place, the data acquisition cycle is initiated in the Arduino Due via the external computer. Collected waveforms are then uploaded to the computer, and analyzed as described in the following sections. For more detail about the procedures to collecting data from a human subject, refer to Appendix C.

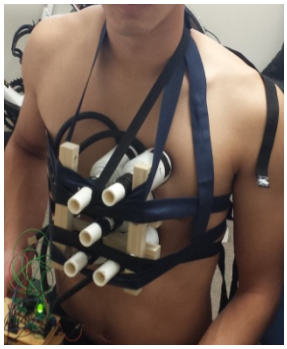


Figure 4.19 Multiple Digital Stethoscope Apparatus Mounted to the body

Shown in Figure 4.20 is the data collected simultaneously through the five sensor channels in the Stethoscope apparatus. It shows the digitized signals plotted versus time; with the amplitudes normalized to make it easier to compare the heart signals.

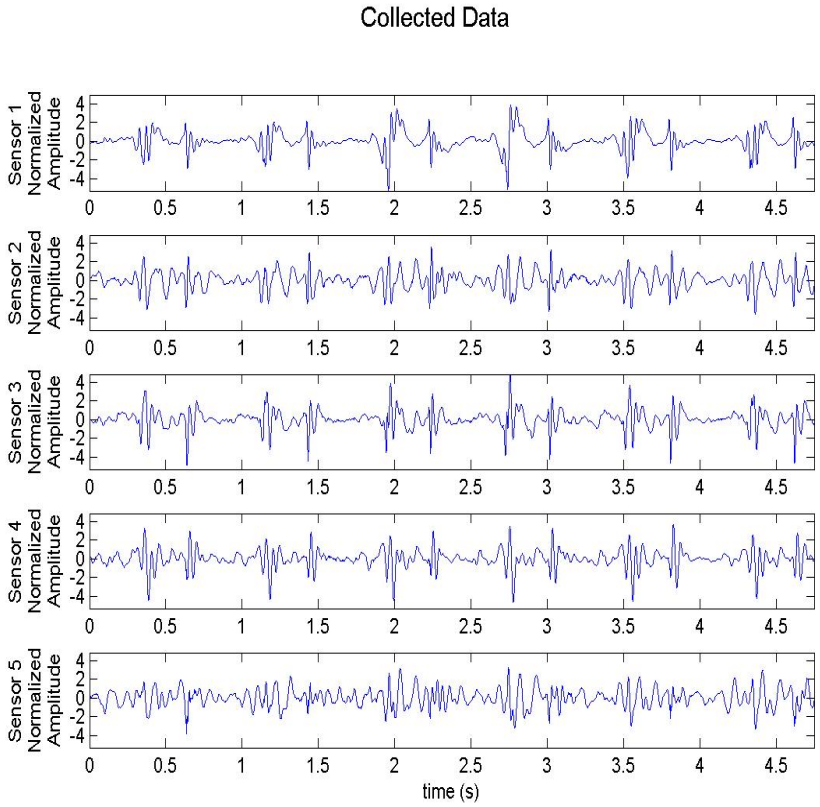


Figure 4.20 Normalized Data Collected

Importantly for this test, none of the input signals experience any clipping; indicating that the input channel electronics and A/D converter had sufficient measurement range. All five signals received resemble those expected from a healthy heart signal. Some of the signals appear to be noisier than the others, like sensor 5; while others resemble the expected heart sounds better, as in sensor 3. However, all five channels do contain recognizable waveform activity during the apparent S1 and S2 sound occurrences.

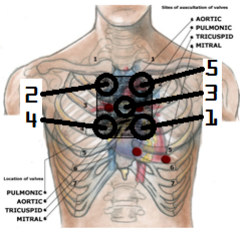
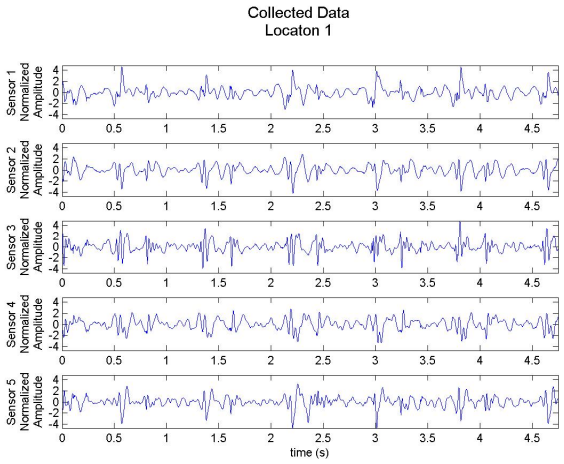
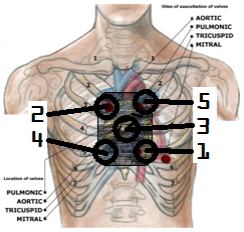
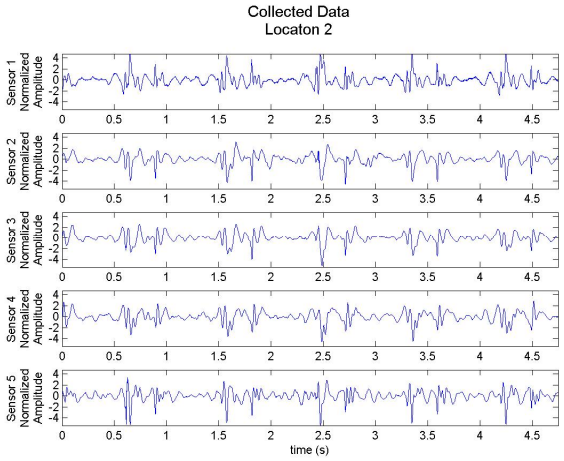
The noisier signal from sensor 5 may be attributed to the larger distance between the channel 5 stethoscope and the valves generating the sound. It may also indicate more susceptibility to external noise sources (ambient room noise), due to its location or difference in contact pressure. When comparing the signals in Figure 4.20, there is also a noticeable difference in delay between the heart

sound waveform peaks from the different sensors which will need to be corrected in the beamforming process before combining the signals together.

In conclusion this test verifies that the heart sound waveform data that can be reliably collected by the stethoscope apparatus.

4.4 Multiple Stethoscope Apparatus Body Placement Test

As mentioned in the *Introduction*, proper placement of the stethoscopes is important for receiving clear, strong heart sounds. The closer the stethoscopes are to the heart valves and turbulent blood flow producing the sounds, the stronger the sound waveforms picked up by the stethoscope diaphragm, as the sound waves are absorbed and attenuated by the tissue intervening between the heart and the skin. One problem with the multiple stethoscope apparatus is the uncertainty of where best to observe different heart sounds, since the different standard auscultation observation points cannot all be contacted at the same time, and it is desirable to have a single mounting location for the apparatus (rather than moving it around as a physician does with a standard single-diaphragm stethoscope). Therefore, a test was needed to understand the best location to place the stethoscope apparatus on the body to monitor the heart beat features of interest. The following experiment observes the waveform results from three different stethoscope apparatus locations on the body. The waveforms are compared and quantified to determine which of the three locations is best for observing the S1 and S2 features of heart beat as optimized beamforming to enhance these features will be explored later in this study. It is important to keep in mind that the data displayed in the figures that follow are again amplitude normalized to make it easier to compare the heart signal waveforms received at each different sensor.

	Placement	*Sound*
Location 1	 Figure 4.21 Location 1	 Figure 4.22 Collected Data from Location 1
Location 2	 Figure 4.23 Location 2	 Figure 4.24 Collected Data from Location 2

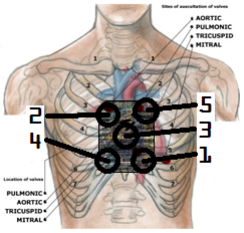
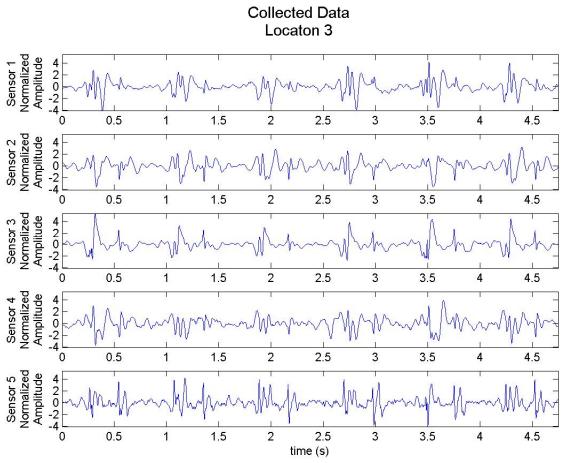
Location 3	 Figure 4.25 Location 3	 Figure 4.26 Collected Data from Location 3

Table 4.4 Location Experiment Results

The table above (Table 4.4) shows results from three different apparatus locations that were recorded to get a better understanding of the best location to observe the heart sounds. The second column in the Table (*Location*) shows the placement of the multiple stethoscope apparatus used in each test to observe the body sounds. Figure 4.21, Figure 4.23, and Figure 4.25 in this column show images of the chest with a black box representing where the multiple stethoscope apparatus was placed for each test. The third column (*Sound*), with Figure 4.22, Figure 4.24, and Figure 4.26, shows the signal waveforms that were observe by each sensor at that apparatus location. Note that each individual signal was divided by its root mean square value to normalize the waveform amplitudes and to make it easier to compare the shapes of the signals.

4.4.1 Observation

From the waveform results shown in Table 4.4 there are some differences in the characteristics of interest in the heart sounds observed in the three different locations. The first signal characteristics of interest is whether or not all the sensor signals basically resemble a heart signal, and are they all are fairly strong (have sufficient amplitude). Also observed and compared is how well the signals from different sensors correlate with each other.

4.4.2 Location 1

This test was observed on the most upper center region of the chest of the three test were done as shown in Figure 61. The signals observed in this test seem to have a pattern but it doesn't look like a normal heart signals. This signal also looks to be noisy. This data is probably not the most ideal data to work with.

4.4.3 Location 2

This test was done a little bit lower than location 1. The result seems to be similar to what was found in location 1, but less noisy. The data in collected in sensor 3 and 5 seem to be correlated, but the other sensors still seem pretty poor. After further investigation it was found that when the signals were aligned they were more correlated to each other.

4.4.4 Location 3

Location 3 was measured a bit lower than location 2. The signals here seem to be less noisy and more of the data collected seem to resemble a heart sound specifically the data collected from sensor 2, 3 and 5. Even the data from sensor 3 was clipped. From the results found in the quantitative analysis it was determine that after the signals were aligned they had the least correlated from the other location.

4.4.5 Quantitative Analysis

	Location 1	Location 2	Location 3
Average Correlation Coefficient to Sensor 3	0.72	0.83	0.69

Table 4.5 Average Correlation Coefficient to Sensor 3

above in the Table 4.5 is a quantitative analysis of the waveform quality at each of the three different apparatus placement locations. This table compares the average correlation coefficient of the S1 signal features in the waveforms from each sensor, when these S1 signal features are time aligned and correlated with those from the sensor 3 waveform. Thus, the highest average correlation coefficient indicates that the surrounding sensor signals (from sensors 1, 2, 4, and 5) most closely resemble the

heart sound waveforms captured in the center of the apparatus on sensor 3. The results found from this quantitative analysis show that Location 2 achieves the signals that are most correlated. Thus location 2 data will be used for further analysis on alternative beamforming techniques.

5 Optimized Beamforming Analysis

5.1 Purpose

The multiple-stethoscope apparatus and data acquisition system described previously were developed for this project to enable the investigation of several alternative signal processing methods that could potentially improve the quality of heart sounds provided to a diagnostician, and possibly provide new sound perceptions that might prove to be clinically beneficial.

Typically, a physician will move an acoustic stethoscope diaphragm to different locations on the chest in order to focus on or better hear sounds emanating from different heart valves and chambers. With the multiple-stethoscope apparatus, it is possible to provide a physician with access to many different auscultation listening points on the body at the same time. Beyond the obvious use of allowing a physician to quickly switch between listening positions, the multiple input architecture also enables us to employ the concepts of sensor array beamforming to combine signals from multiple simultaneous observation points to either improve the ability to focus on the primary signal source and remove peripheral noise signals, or to provide a more comprehensive acoustic signature that includes the primary source sound and some additional relevant information gleaned from the surrounding area (such as additional subtle sonic details, or echoes and reverberations that might provide insight into the condition of the surrounding structures).

In standard beamforming, the signals from multiple sensors are repositioned in time so that the signals emanating from a particular source location in space are temporally aligned to reinforce each other, producing a stronger aggregated signal, as shown in Figure 5.1. In so doing, noise signals emanating from other spatial locations will not be properly aligned in the different sensor signals after beamforming delays are applied. Therefore, the noise signals will be effectively diminished in the beamformed signal as they are not reinforced by the addition of adjacent realigned waveforms. Instead, the beamforming summation acts as a signal averaging operation that builds up the multiple copies of the

aligned signals from the desired source position while not reinforcing the misaligned noise source signals; thus increasing the signal-to-noise ratio of the beamformed sum.

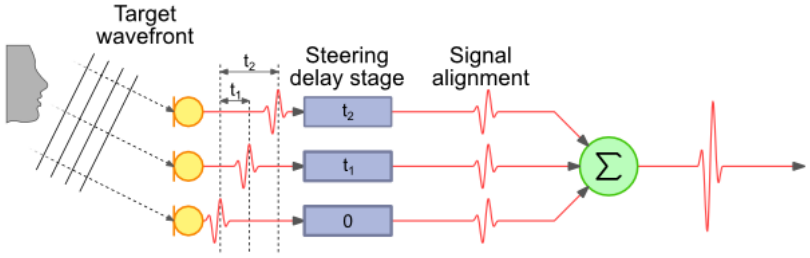


Figure 5.1 Beamforming Example

Typically the beamforming delays applied to align each sensor waveform to reinforce sounds from a particular signal source location are determined by the difference in propagation time for sound waves from the desired signal source position to each pickup sensor. These differences in propagation time are caused by the differences in physical distance from the source to each sensor, materials the sound travels through and the individual characteristics of each stethoscope. This assumes that the acoustic propagation velocity is the same for all paths from the source to each sensor, which is not strictly true, but is usually assumed to be so. This type of geometric beamforming requires exact knowledge of the spatial location of each pickup sensor (each stethoscope diaphragm and microphone) and a good estimate of the average sound speed (acoustic propagation velocity) in the tissue between the source and the sensors.

The need for a depth-compliant stethoscope diaphragm to ensure proper contact with the chest wall in the apparatus for this project means that we do not know the exact “z” position of each stethoscope sensor. This makes typical geometric beamforming difficult to implement accurately with this particular apparatus. Therefore, several alternative methods for optimized beamforming were explored in this project.

5.1.1 Beamforming Waveform Data Set

To test the alternative beamforming methods, a standard set of normal heart sound test waveforms collected through the multiple stethoscope apparatus were required. The details of the process used to collect the in-vivo data discussed in this experiment can be found in Appendix C. The body sounds were collected from the author, Spencer Wong, who does not have any known heart problems, based on a diagnostic determination made by his primary care physician. Below in Figure 5.2 is a picture of the waveform data collected for this analysis. This data is the same data shown in Multiple Stethoscope Apparatus Body Placement Test section for Location 2. The data shown in this figure is the unnormalized version that was received by the microcontroller, with corrections made to some of strongest signal peaks to undo the effects of some minor clipping observed in a few of the waveforms. MATLAB scripts were written to process and analysis the data. These scripts can be found in Appendix R and Appendix S.

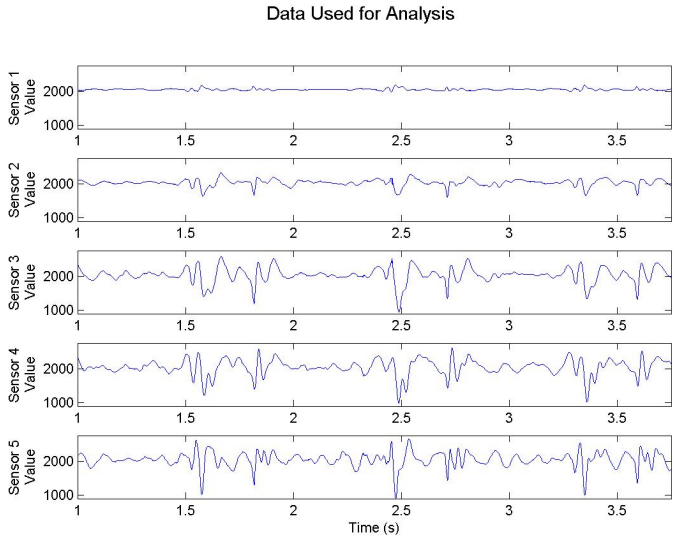


Figure 5.2 Original Sensor Signals (Unnormalized)

The next step in preparation for beamforming was to normalize the amplitudes of the data received in each channel to the maximum value received in each sensor waveform. This also helped to make the waveform analysis results easier to interpret visually.

5.1.2 Waveform Feature Segmentation

The next beamforming preparation step was to segment each waveform into sections that separate the S1, S2, systole and diastole periods. The sections were manually segmented by visual inspection, using fixed time window lengths for each feature. The length of the window used for the S1 feature is 175 ms, the S2 segment is 100 ms, systole is about 130 ms and diastole is about 435 ms. The same feature segmentation windows were used for all five waveforms, and so the windows had to be large enough to encompass the features in all waveforms despite different time delays in each channel. Figure 5.3 below shows one section of the normalized and segmented waveform data. Here, the blue highlight represents the S1 feature, green highlight shows the systole period, red highlight shows the S2 feature, and the yellow highlight shows the diastole.

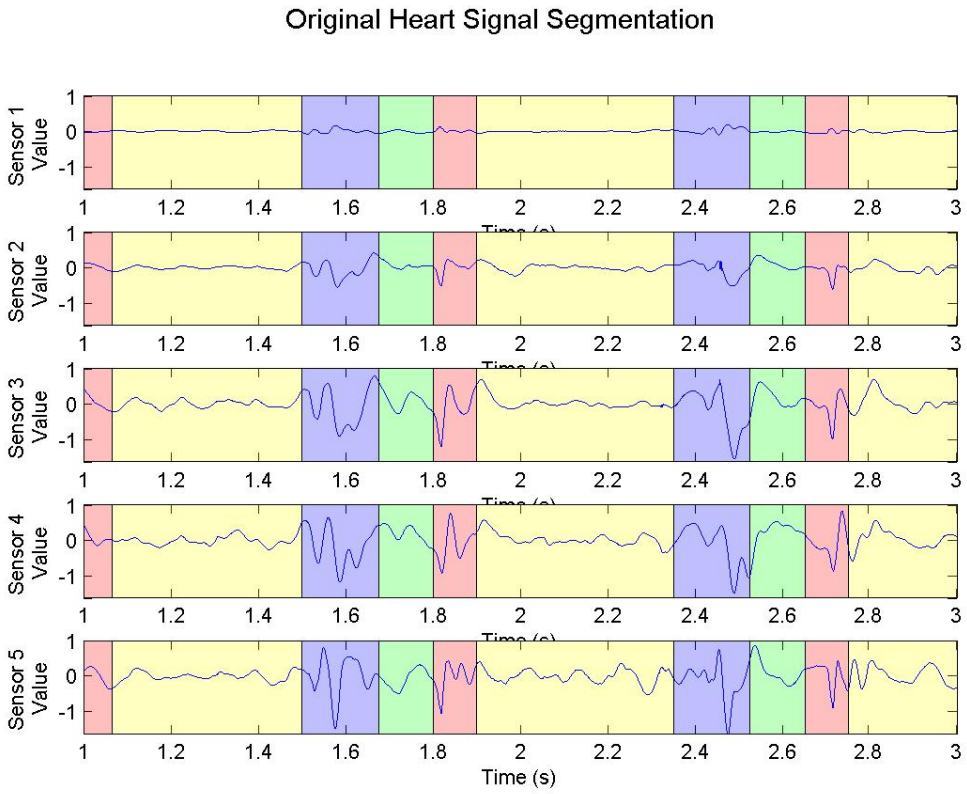


Figure 5.3 Original Signal with Section of the heart sounds identified.
(Blue = S1, Green = systole, Red = S2, Yellow = diastole)

5.1.3 Masking Signals to Extract Features of Interest

With the waveform data segmented, each individual feature in the heart sound sequence can be separately analyzed. The sections of particular interest for analysis here are the S1 and the S2 sounds. It should be noted that in a diagnostic setting, the systole and diastole periods might also be of interest, as murmurs can occur during these times. For the subject waveforms used in this experiment, where there were no heart murmur sounds present, only the S1 and S2 waveforms were optimized. For future clinical applications, the same processes developed here to optimize listening to the S1 and S2 features could similarly be applied to optimizing sounds occurring during the systole and diastole periods.

Shown below in Figure 5.4 is the masked and extracted S1 and S2 data from all five channels that will be used to determine the time delay needed for optimal beamforming. The top diagram shows the masked S1 features and the bottom diagram shows the S2 features.

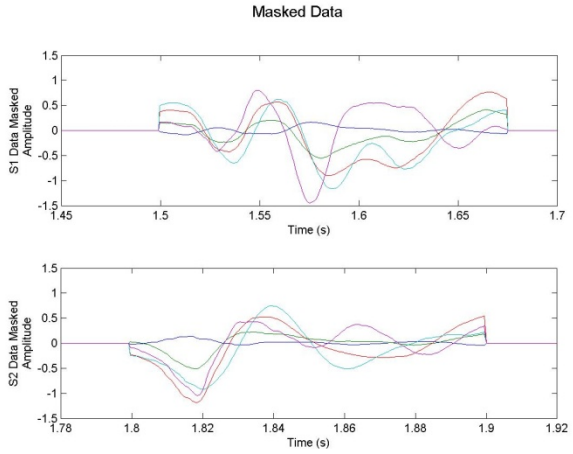


Figure 5.4 Masked S1 Data (top), Masked S2 Data (bottom)

5.1.4 Reference Signal

To align the five waveform signals for beamforming, one specific sensor needs to be selected as the time reference signal, to which the other signals will be time shifted to align their features of interest. The data received from sensor 2 was chosen for this analysis. This sensor was chosen because it had

the strongest average correlation coefficient with all the other sensors. Shown below is the sensor 2 data that was collected, and the average correlation coefficients computed using each sensor as the correlation comparison reference. Notice that the Sensor 2 data has similar attributes to a normal heart signal shown in Figure 1.5 Normal Heart Sound S1 and S2 Figure 1.5.

Sensor Average Correlation Coefficient Comparison					
	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5
Average Correlation Coefficient	0.58	0.85	0.83	0.80	0.78

Table 5.1 Sensor Average Correlation Coefficient Comparison

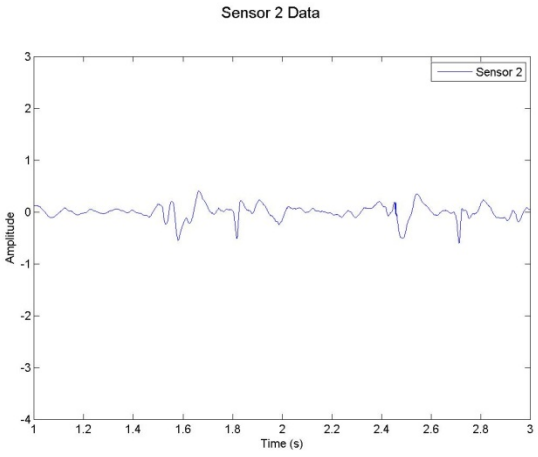


Figure 5.5 Sensor 3 Data

5.2 Heart Sound Feature Signal Alignment

With the strongest heart signal selected, the other signals now need to be aligned by removing the relative time (or phase) delay differences between them. There are a number of steps needed to determine the differential phase delay between two compared signals. The MATLAB Code used to determine the phase delay is shown below in Table 5.2.

The first step is to extract the samples in the vicinity of the feature of interest within the two signals (as described above). Then cross correlations between the two signal segments are computed, providing a measurement of similarity between the reference signal and successively time-shifted versions of one other sensor signal. The time shift resulting in the highest cross-correlation value

(greatest similarity) is assumed to be the shift needed to best align the features in the two signals. Thus, the cross correlation peak index relative to the zero-offset correlation position (the center index of the cross correlation result) is used to find the relative time shift needed to best align the desired feature in the two signals. This index difference will determine how many sample intervals one signal needs to be shifted to align with the feature of interest in the reference signal.

Thephasedelay.m

```
function [numshift, timedelay] = thephasedelay(data1,data2,samplingfreq)
        sampletime = 1/samplingfreq;
        c = xcorr(data1, data2);
        centerofxcorr = size(c,1)/2;
        maxc = max(c);
        numshift = find(c == maxc)-centerofxcorr;
        timedelay = numshift*sampletime;
end
```

Table 5.2 MATLAB Code used to determine the Phase Delay

Different beamforming optimizations can be performed by using different heart sound feature segments in the phase delay estimation process. For example, to optimize the beamforming to get the best S1 sound results, one could extract just the S1 sound segments and perform the phase delay estimation using only those segments (ignoring the alignment of the S2 and other waveform features). Theoretically, beamforming with these delay estimates should produce the best possible S1 sound, at the expense of possibly de-emphasizing other sound features like the S2 sound. Similarly, extracting and correlating only the S2 features should lead to beamforming results that provide the strongest, most detailed S2 sound (at the expense of other features). Performing phase delay estimation with signal segments that include both S1 and S2 features should result in a best compromise beam formation to hear both features reasonably well.

Using the peak correlation phase delay estimation algorithm, time delays were determined to align each of the sensor signals to the data collected by the reference sensor 2. First the signals were aligned and beamformed using phase delays estimated using only the S1 feature extracts from each waveform. Then, this was repeated but using only the S2 feature extracts. The time delays calculated for these two cases are shown below in Table 5.3. Notice that the time delays for best aligning the S1 and

S2 features are different. This verifies that the S1 and S2 signals are emanating from two different spatial locations in the heart.

Time Delay Comparison

	S1 (ms)	S2 (ms)
Sensor 1	79.8	19.8
Sensor 2	0	0
Sensor 3	-0.8	0.3
Sensor 4	-3.8	2.3
Sensor 5	53.3	0.3

Table 5.3 Time Delay Comparison

The results of these alternative signal alignments are shown in Figure 5.6 and Figure 5.7, where the top diagram shows the original signals and the bottom diagram shows the aligned signals. Notice in the top diagrams of Figure 5.6 how the signals are misaligned. It is almost impossible to tell if they are correlated. Also notice that by using the computed phase delays, the signals are fully aligned as shown in the bottom diagram in Figure 5.6. Similar results can be seen for the alignment based on S2 in Figure 5.7.

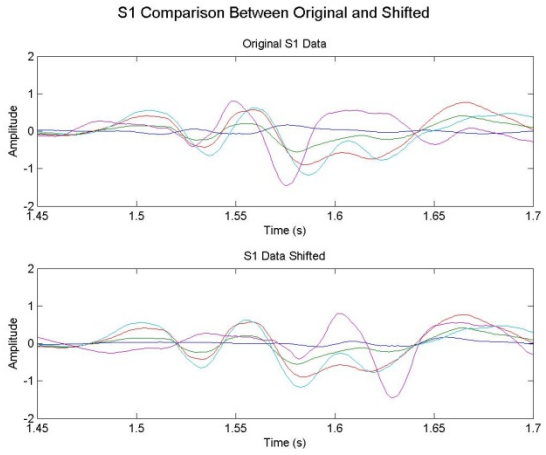


Figure 5.6 S1 Comparison Between Original and Shifted

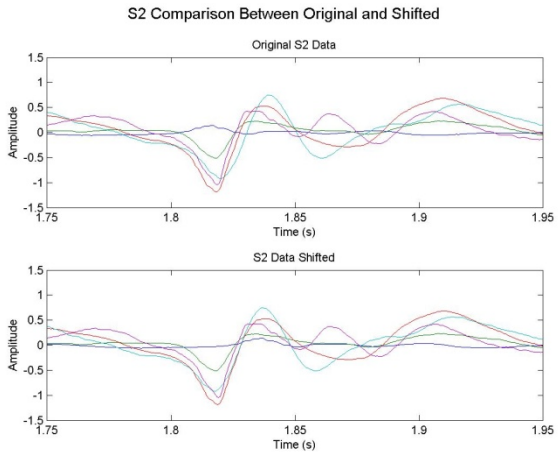


Figure 5.7 S2 Comparison between Original and Shifted

5.2.1 Correlation Comparison

The improvement in signal correlation between all the sensor signals and the sensor 2 data resulting from the time alignment of the waveforms is quantified in Table 5.4 and Table 5.5 below. This table was generated by computing the correlation coefficient values for both the original (unaligned) and the aligned masked heart signals for each sensor with the sensor 2 signal. These tables show a significant improvement after alignment for many signal correlations.

S1 Alignment Correlation Comparison Table

	Before Alignment	After Alignment
$\sigma_{21\ S1}$	-0.77	0.60
$\sigma_{22\ S1}$	1.00	1.00
$\sigma_{23\ S1}$	0.93	0.93
$\sigma_{24\ S1}$	0.83	0.88
$\sigma_{25\ S1}$	0.32	0.40

Table 5.4 S1 Alignment Correlation Comparison Table

S2 Alignment Correlation Comparison

	Before Alignment	After Alignment
$\sigma_{21\ S2}$	-0.78	0.44
$\sigma_{22\ S2}$	1.00	1.00
$\sigma_{23\ S2}$	0.91	0.91
$\sigma_{24\ S2}$	0.74	0.78
$\sigma_{25\ S2}$	0.92	0.91

Table 5.5 S2 Alignment Correlation Comparison

5.3 Beam Summation

With the signals phase delays determined, the final step to optimal beam forming is to sum the time-aligned signals together; either the entire waveforms (to hear the complete heart sound sequence) or masked extracts of the waveform (for focused listening to particular sound features without clutter or confusion from other features). Again, this summing operation will constructively reinforce the aligned feature sounds and de-emphasize by averaging the misaligned noise signals or sounds from other sources. A comparison of the optimized beamformed signal to the sensor 2 reference signals for both the S1 and S2 sound segments is shown in the figures below.

5.4 Processed Signal Analysis

Using the techniques described above, the next challenge is to explore different possibilities for methods of combining all the body sounds acquired from the different sensors in clinically helpful ways. The goal of combining the heart sounds is to create a strong, clear body sound presentation for a doctor to better observe and diagnose potential clinical problems. The results found with each of the methods below are experimental and need further research to verify any clinical efficacy, since we do not have professional training in auscultation or cardiac diagnosis. As shown below, four different methods for optimizing and combining the multiple stethoscope sensor data were explored in this project.

5.4.1 Method 1 – S1 and S2 Isolation (Masked Systole and Diastole)

Method 1 is a processed heart sound that combines the beamformed S1 signal excerpts optimized for S1 alignment, with the beamformed S2 excerpts optimized for S2 alignment, with the systole and diastole sounds masked out (silenced) from the heart sound sequences. This method is used to highlight the important S1 and S2 signal characteristics in a single sound file. The result is shown below in Figure 5.8 for one of the S1 and S2 time excerpts (not including the silenced diastole period).

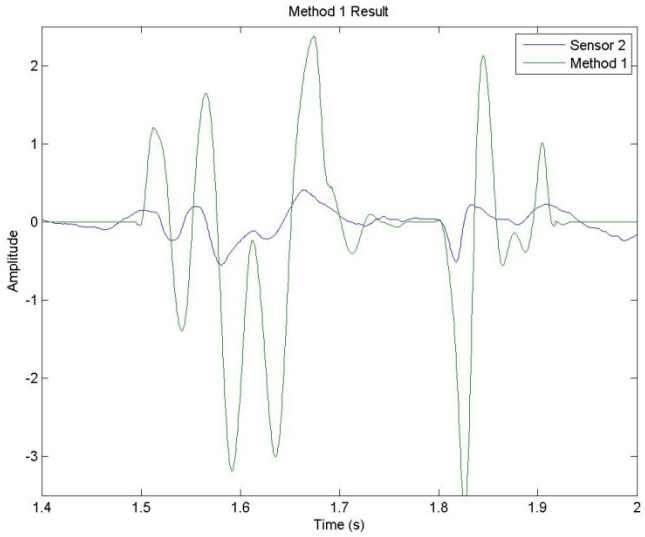


Figure 5.8 Analysis 1 Result

Note that the Method 1 signal has much higher amplitude than the original Sensor 2 data. Note also that the silence imposed between the S1 and S2 signals certainly achieves the desired results of not distracting the listener from the details of the S1 and S2 wave sections, as can be heard in the wave audio files produced from this test. However, for a clinician, this is only helpful if the sounds they are listening for are within the S1 or S2 segments (as in heart beat gallops and mitral valve prolapse). Heart murmurs and regurgitations would be mostly discarded in the silenced sections. Therefore, this processing method would not be helpful for heart murmur detection. Instead, a similar method that silenced the S1 and S2 waveforms and optimized the beamforming for masked systole and diastole periods might be applied for murmur detection.

Typically, an SNR calculation would be performed to demonstrate an improvement in the optimized waveforms produced here. However, since the SNR is computed using the diastole and systole waveform segments between S1 and S2 features as the “noise” amplitude measurement, it would not be valid to compute an SNR for this method 1, as the masked (silenced) “noise” times would produce an infinite SNR (zero noise variance). Even without a quantitative metric, however, the waveform picture shown in Figure 5.8, along with the audio file produced from this waveform, both clearly demonstrate that this method achieved the goal of improving and isolating the S1 and S2 sounds, providing a clearer,

stronger, unadulterated, and hopefully helpful heart sound record for a clinician to focus on the characteristics of the S1 and S2 features.

5.4.2 Method 2 – S1 and S2 optimized (Systole and Diastole retained)

For this method each heart beat sound sequence was divided into two different sections: extended S1 and extended S2, as shown below in Figure 5.9. One of each of these sections (one extended S1 segment plus one extended S2 segment) encompasses the complete sound from a single heartbeat. Two separate beamformed signals were then constructed: one starting with all of the extended S1 segments (with zeros between them during the extended S2 periods) and the other starting with all of the extended S2 segments (with zeros between them during the extended S1 samples). The extended S1 collection waveform was then aligned and summed based on the masked S1 excerpt optimization and the extended S2 sections were aligned and summed to optimize just the masked S2 segments. The aligned and beam summed extended S1 and S2 signals were then summed together, and processed through a low pass to filter to smooth any transitions. An excerpt of the resulting single waveform result is shown in Figure 5.10. Sensor 2 data is included in this diagram as a reference. Here, again, the diastole period is not shown, even though it was part of the extended S2 sections. In this case, the diastole would not be silent as it was in Method 1 but would instead be a beamsummed result aligned based on optimizing the S2 waveforms. Thus, a clinician could still hear heart murmurs and other artifacts occurring during diastole (and similarly also during systole) with this method.

Figure 5.9 Method 2 Heart sounds split up in S1 and S2 sections

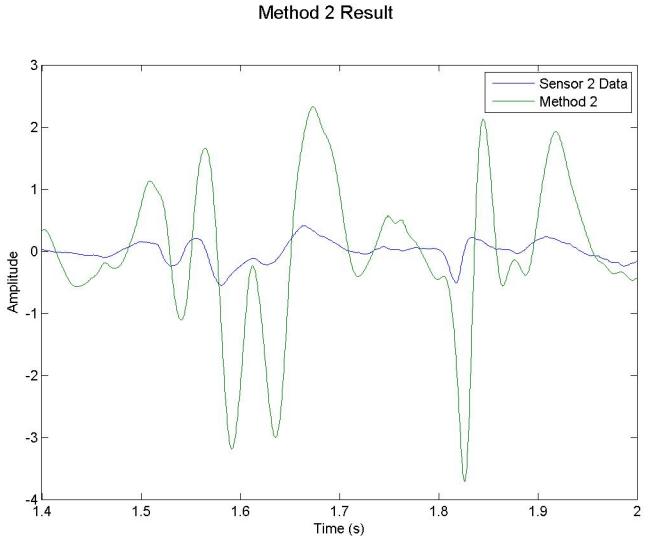


Figure 5.10 Method 2 results

Here, an SNR computation can be used to demonstrate the improvement in the heart signal. Again the SNR used here compares the variance of the signal in either the masked S1 or masked S2 segments to the variance in the "noise" level in the times between the S2 and the S1 (the systole and diastole times). The computed SNR results are shown in Table 9. This shows a dramatic improvement in SNR for both the S1 and S2 features. The question remains as to whether or not this would be helpful to

a clinician for diagnostic purposes. These results, however, suggest that this could be a very helpful method for improving the audibility of important features and artifacts in the heart sounds.

Method 2 SNR Table Comparison		
	S1	S2
Original (Sensor 3 signal)	80.77 dB	55.56 dB
Analysis 2 Processing	92.94 dB	82.10 dB

Table 5.6 Method 2 SNR Table

5.4.3 Method 3

Method 3 is the method that best simulates standard beamforming. It applies a single, unique time delay to each sensor signal to optimize one feature in the waveform, and sums the aligned waveforms. The issue for us is that we have two different sets of “optimum” time alignment delays for each signal – one set computed to maximize the cross correlation of the S1 signals and a different set found to optimize the S2 feature correlations. Therefore, this method was applied twice – once using the S1 optimized phase delays and again using the S2 optimized phase delays.

Method 3 S1 simulates beamforming optimized around the S1 features, and method 3 S2 simulates beamforming to favor the S2 features. For the Method 3 S1 results, the complete waveforms from each sensor were realigned using the S1 optimizing phase delays, and then beam summed to create a single aggregate signal. Likewise, the Method 3 S2 results were similar y generated by aligning the full sensor signal waveforms using the S2 optimizing delays, and summing these into a single signal.

An excerpt of the waveform results is shown below in Figure 5.11 and Figure 5.12. Sensor 2 data (Blue) is included in the diagram as a reference to the beamformed heart signals (Green). It is interesting to note that the S2 feature is still clearly apparent and was not degraded in the Method 3 S1 waveform. Similarly, the S1 feature is also enhanced and clearly improved in the Method 3 S2 results. It appears, therefore, that the phase alignment optimizations for each of these features are not so different as to cause degradation in the other key feature.

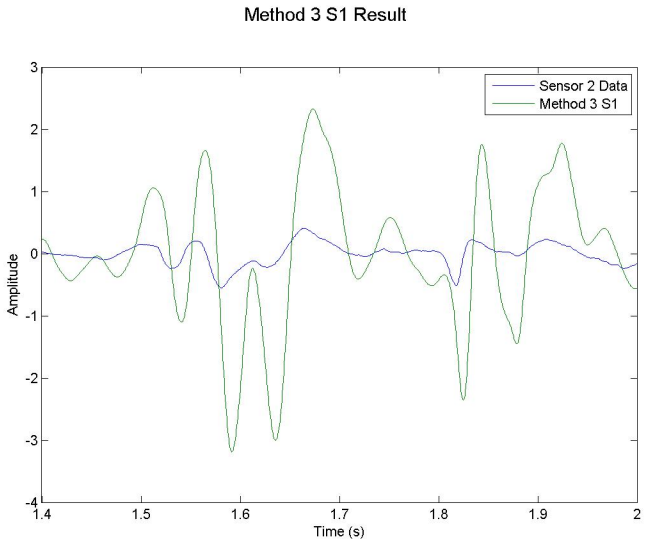


Figure 5.11 Method 3 S1

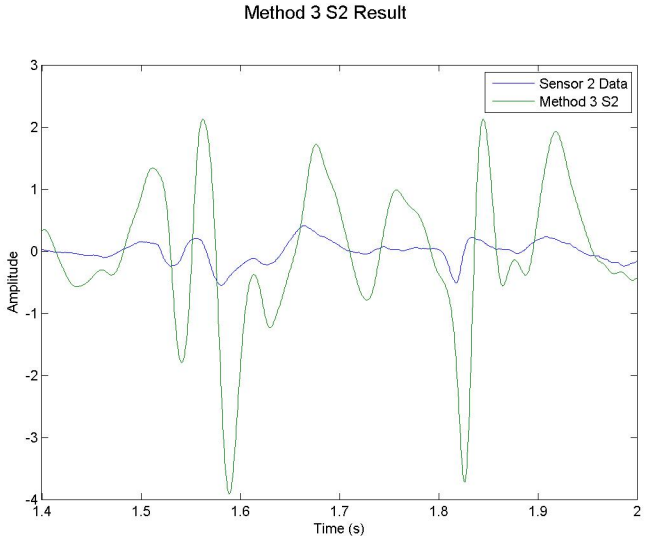


Figure 5.12 Method 3 S2 Result

To show quantitatively that there was an improvement in the heart signals with this beamforming method, the signal to noise ratio metric was again used. Below in Table 5.7 are these metrics. This table shows improvements in the signal to noise ratio for *both* the S1 and S2 signals with *both* phase delay optimizations. As expected, the Method 3 S1 results beamformed to optimize the S1 signal shows about a 4 times improvement (14 dB) in S1 SNR. Similarly, the Method 3 S2 results from beamforming

optimized for the S2 feature, shows a huge improvement (35 dB) in S2 feature SNR compared to the single Sensor 2 signal. Surprisingly, though, the Method 3 S1 optimization also improved S2 SNR by 6 dB; and the Method 3 S2 optimization exceeded the Method 3 S1 optimization for S1 SNR by 5 dB. This latter result is unexpected, and may be indicative that one or more of the "optimum" S1 phase delays may not be providing the best SNR improvement, even though it improves the signal S1 feature correlation coefficient.

Method 3 Signal to Noise Ratio Table

	S1	S2
Sensor 2 Data	80.77 dB	55.56 dB
Method 3 S1	89.94 dB	61.40 dB
Method 3 S2	94.87 dB	90.58 dB

Table 5.7 Method 3 Signal to Noise Ratio Table

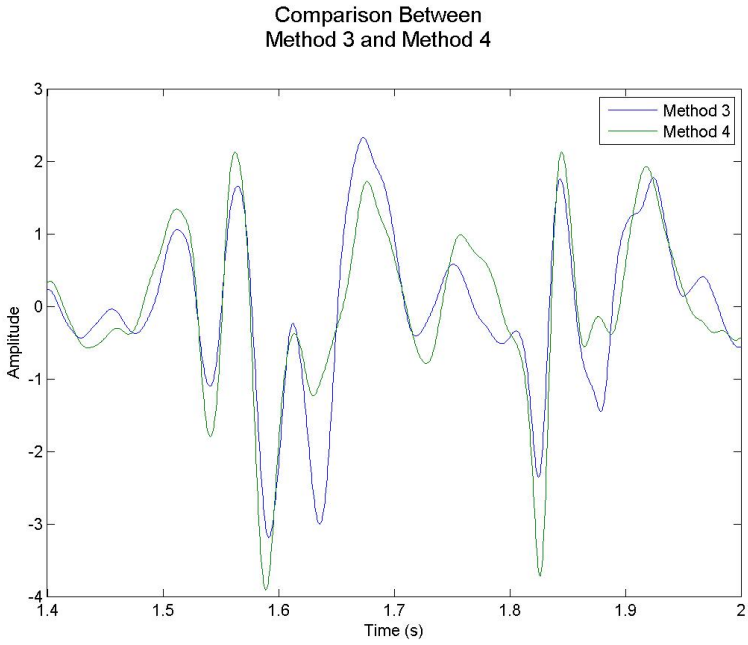


Figure 5.13 Comparison between Method 3 S1 and Method 3 S2 results

5.4.4 Method 4

Method 4 is a method that isolates only the signal feature of interest. It is similar in signal processing to Method 3, except that it does not use the full sensor waveforms. Instead, Method 4 S1 beamforms the masked S1 signals (masking out everything else including the systole, S2 features, and

diastole periods), using the S1-optimized phase delays. Similarly, in Method 4 S2 sums together the masked D2 signals (with the S1, systole, and diastole segments masked out) aligned using the S2-optimzed phase delays.

The beamformed signals from one heartbeat using each of these methods is shown below in Figure 5.14 and Figure 5.15. As with the masked signals in Method 1, where the "noise" periods are masked to silence, the effectiveness of this method again cannot be quantified using a signal-to-noise ratio. It does, however, demonstrate the potential for post-processing heart sound waveforms to enable a physician to completely isolate and optimize particular features that might enable them to distinguish between some of the more acoustically subtle and similar artifacts of different heart disease processes. Thus, this method was generated for a professional to listen to; and requires a subjective assessment by a clinician to determine if it is diagnostically relevant or helpful.

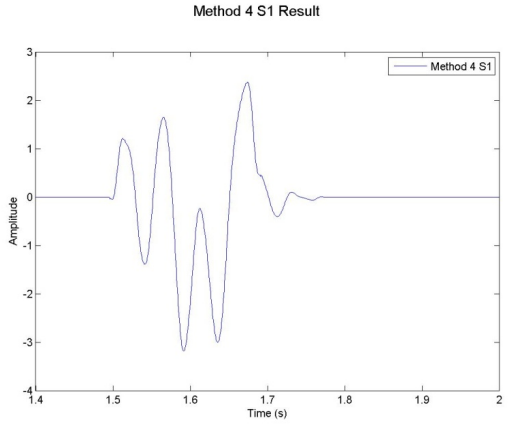


Figure 5.14 Method 4 S1 Results

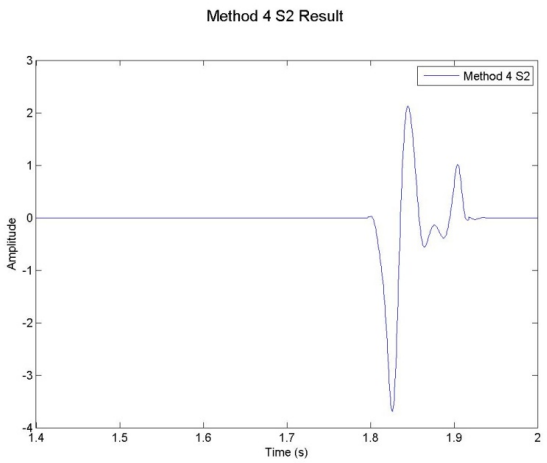


Figure 5.15 Method 4 S2 Result

Comparing the results of this Method 4 shown in Figure 5.14 and Figure 5.15, to the Method 3 results in Figure 5.11 and Figure 5.12, one would rightly conclude that the Method 4 results could have been generated from the Method 3 results, simply by masking the appropriate segments of the Method 3 S1 and Method 3 S2 waveforms. Except for the effects of the filtering used in Method 3 and possibly minor differences in the beamforming results at the edges of the masked S1 and S2 regions, the Method 4 S1 waveform should be nearly identical to the S1 segment of the Method 3 S1 results (with the systole, S2, and diastole sections masked out). Also, the Method 4 S2 waveform should match the S2 segment of the Method 3 S2 result (with the S1, systole, and diastole sections masked off). This is due to the similarity of processing steps used to generate the beamformed signals in these two methods.

5.5 Stereophonic Presentation Analysis

This project also evaluated another possible application for the multi-stethoscope apparatus - namely that the data gathered from multiple pickups could be used to create a stereophonic presentation of the heart sounds that might prove clinically helpful. Shown below are four different stereo signals generated from different pairs of sensors using the data already collected. The sound generated is intended for listening through headphones, so that the independent waveforms can be separately

presented to each ear of the listener. The blue signal in these figures represents the headphone signal for the left ear, and the green signal represents the headphone signal for the right ear.

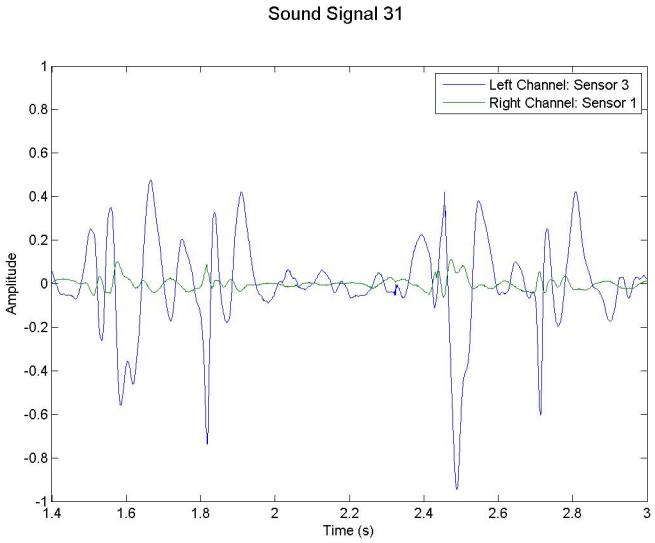


Figure 5.16 Sound Signal 31

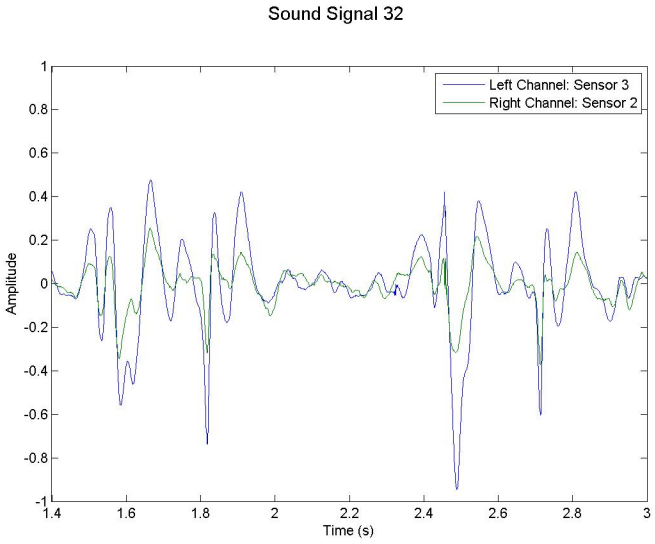


Figure 5.17 Sound Signal 32

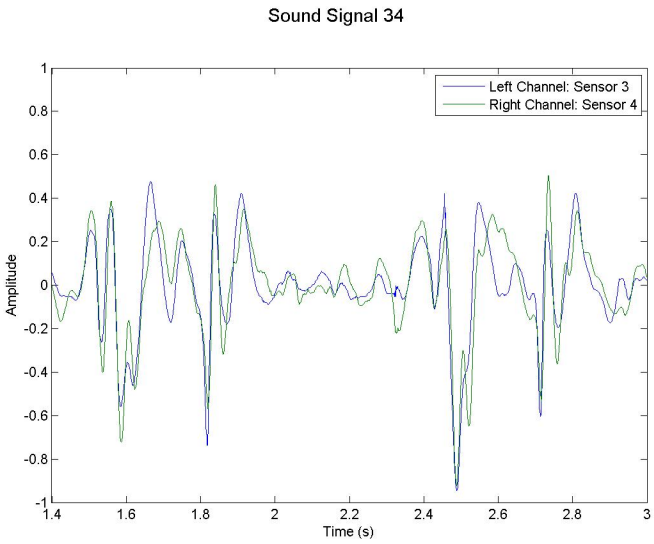


Figure 5.18 Sound Signal 34

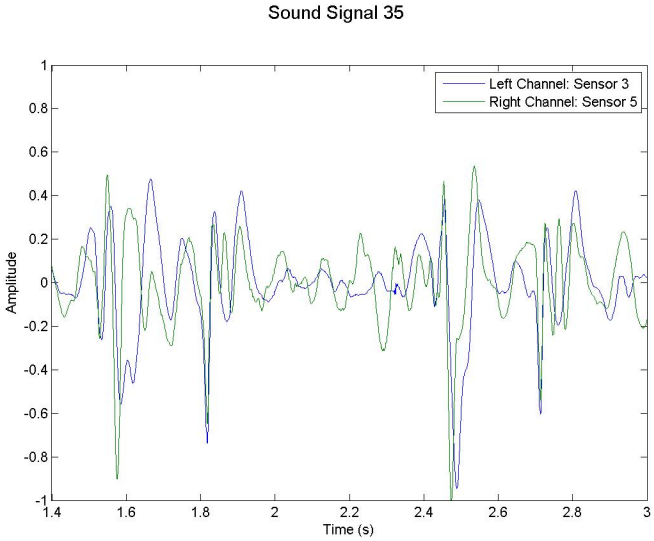


Figure 5.19 Sound Signal 35

5.6 Analysis

The graphical representation shown above does not represent this experiment well as this sound is meant for listening to and discerning if it helps determine where the sound is emanating from, or if it provides some other subtle acoustic cues that would be diagnostically beneficial. One

observation that can be made from the data shown above is that there does appear to be a different delay in the sound features presented to the left and right headphones, which should result in the perception of a particular lateral position of the sound source for the listener. Another observation that can be made is the sound produced for the left ear sometimes has greater amplitude than the sound produced for the right ear. This can give the illusion that the sound is closer to the left ear than the right ear.

To the untrained ear, the stereophonic presentations do provide a different experience for the listener, providing the perception of hearing sounds from a wider field and with greater “depth” (a subjective term) possibly due to perceiving the delayed waveform in one ear as an echo or reverberation of the original sound.

6 Future Work

In the process of finishing this project, I learned that there are many improvements that could be made to this project. These improvements are discussed below.

6.1 Electrical

There are many improvements that can be made in the electronic design of this project. The most significant work that needs to be done is to have a better wire management, meaning having shorter wire connections between the microphones and the preamplifiers. Long wires can result in noise susceptibility. Also, the system is more liable of breaking with long wires without strain relief.

Additional research should be done on using piezoelectric or electromagnetic diaphragm to improve the data recording. This could be a viable solution for removing the tube and the microphone.

6.2 Mechanical

In terms of the mechanical design, the biggest improvement that can be made on the stethoscope apparatus is to be able to actively determine the x, y and z location of the stethoscope relative to each other. This would be a necessary improvement to be able to do accurate source-directed beam forming for localizing listening inside the body. This could be done by putting additional sensors on the apparatus to determine the location of the stethoscope, particularly the depth (z axis) location of the compliant diaphragms. In addition to the placement of the stethoscope, a sensor to detect the pressure with which the stethoscope is placed on the body would be helpful. This would be useful because the frequency response of the stethoscope varies depending on the contact pressure between the stethoscope and the body. By knowing the pressure, the stethoscope response can be compensated if needed.

Another improvement that can be made is to use a shorter rubber tube between the diaphragm and the microphone; and to have tubes of exactly the same length in each stethoscope pickup. Having a shorter rubber tube can reduce the amount of noise that may be induced into the tube which will improve

the signal to noise ratio. Uniform tube lengths removes one more variable in the differences in signal arrival times at each sensor that needs to be calibrated out of the system for deterministic beamforming.

6.3 System

Several changes can be made to improve the overall system. The most fundamental improvement would be using more stethoscopes, possibly positioned closer together or in either a basic pattern (ring or grid), or placed directly over all standard auscultation sites on the body. Using more stethoscopes should improve the signal to noise ratio further after beamforming, and enables more precise focusing on signal source locations. The current data acquisition system can handle three more stethoscopes with the same sampling frequency.

Another improvement that can be made to the system is to implement a faster sampling rate and more data storage capability for waveform analysis. This improvement might be accomplished by optimizing the communication between the computer and microcontroller so that the data acquired can be stored in the computer in real time.

Additional work can be done to determine the optimal location to place the stethoscope apparatus so that all five stethoscopes receive stronger heart sounds. Also a study of different methods of observing the heart can compliment each other like an EKG and a stethoscope.

6.4 Potential Application

There are many specialized applications this project can be used for. Such applications include improved detection of heart murmurs and the providing better capabilities to locate the source of a heart murmur or better determination of the type of murmur occurring. Another auscultation application that this apparatus is also suited for would be monitoring lung sounds. Since some lung sounds aren't as loud as the heartbeat, they can be harder to detect. Some modification of the sensor mounting locations would be needed to better optimize the system for lung sound detection. This system could be used to listen to

specific locations within the lungs, using similar beamforming techniques. This system could be used to monitor stomach sounds as well – another application area for standard stethoscopes.

7 Conclusion

In conclusion a multiple stethoscope apparatus was designed, built and characterized. The design of the project included designing the electrical, mechanical and software elements for this system; in addition to designing test methods and analysis software to characterize the stethoscope apparatus. Once the design and characterization was achieved, explorations on collecting body sounds were carried out. Data was gathered from different locations on the body and compared to find the optimal sensor locations to collect data. With the data collected, various method of combining the body sounds for improved listening were explored using different approaches to optimized phase-delay beamformation. Ultimately, all of the beamforming results showed an improvement in signal to noise ratio on the signal features of interest.

This project demonstrated that there is potential for a multiple stethoscope apparatus as an effective and improved tool for listening to the heart. It has provided a “proof of concept” for the basic construction and signal processing required for such as system. While this project wasn’t able to demonstrate the potential benefits of this system for diagnosing clinical issues with a poorly functioning heart, it was successful at providing a starting point for further research on this topic.

BIBLIOGRAPY

(n.d.). Retrieved from http://en.wikipedia.org/wiki/Systole_(medicine)

(n.d.). Retrieved from http://www.webmd.com/heart-disease/heart-murmur-symptoms

(n.d.). Retrieved 4 2, 2014, from http://www.intechopen.com/source/html/19510/media/image2.jpg

3M Littman Stethoscope. (n.d.). *3M Littman Stethoscope.* Retrieved from Redefining Auscultation With Two Exceptional Choices: http://multimedia.3m.com/mws/mediawebserver?mwsId=SSSSSufSevTsZxtUOx2v58_vevUqevTSevTSevTSeSSSSSS--&fn=70-2010-7371-8.pdf

ANALOG DEVICES. (n.d.). *CMOS Quad Sample-and-Hold Amplfier (SMP04).* Retrieved from http://www.analog.com/static/imported-files/data_sheets/SMP04.pdf

Arduino. (n.d.). *Arduino Due*. Retrieved from http://arduino.cc/en/Main/ArduinoBoardDue

Arduino. (n.d.). *Arduino Due Reference Design.* Retrieved from http://arduino.cc/en/uploads/Main/arduino-Due-schematic.pdf

ATMEL. (n.d.). *AT91SAM ARM-based Flash MCU.* Retrieved from http://www.atmel.com/Images/doc11057.pdf

Cardiac Murmurs. (2006-2013). (wlammers) Retrieved 4 2, 2014, from http://www.fmhs.uaeu.ac.ae/wlammersteach/D.CVS/D.3.%20The%20contracting%20heart/D304CardiacMurmursFiles/murmurs.jpg

Greensted, A. (2010, November 29th). *The Lab Book Pages, An online collection of electronics information*. Retrieved Aprl 12, 2014, from http://www.labbookpages.co.uk/audio/beamforming.html

Heart Anatomy. (n.d.). (Prentice Hall's Heart and Lung Sounds) Retrieved 4 2, 2014, from http://www.prenhall.com/heartlungsounds/main/intro.htm

human heart picture with labels. (n.d.). Retrieved 4 2, 2014, from http://www.oxford174.com/human-heart-pictures-with-labels/

Jatupaiboon, N., Pan-ngum, S., & Israsena, P. (2010). *Electronic Stethoscope Prototype with Adaptive Noise Cancellation.* Chulalongkorn: IEEE.

MAXIM INTEGRATED. (n.d.). *Tiny, Low-Cost, Single/Dual-Input, Fixed-Gain Microphone Amplifiers with Integrated Bias.* Retrieved from http://datasheets.maximintegrated.com/en/ds/MAX9812-MAX9813L.pdf

McKee, A. M. (2004). *Beamforming for Multisensor Stethoscope.* Ottawa: Heritage Branch.

Mckee, A. M., & Goubran, R. A. (2004). Beam Shape, Focus Index, and Localization Error for Performance Evaluation of a Multisensor Stethoscope Beamformer. *IEEE EMBS.* San Francisco.

MediaCollege.com. (n.d.). Retrieved 3 26, 2014, from http://www.mediacollege.com/audio/microphones/how-microphones-work.html

Medical Department Store. (n.d.). Retrieved 3 18, 2014, from http://www.medicaldepartmentstore.com/Littmann-Classic-Ii-S-E-Stethoscope-p/2201.htm

MICROCHIP. (n.d.). *2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface.* Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/21298c.pdf

Murphy MD, D. R., Murphy, P. R., Brockington, M. G., & Vyshedskiy, P. A. (2006). *Prentice Hall's Heart and Lung Sounds Introduction*. (Prentice Hall) Retrieved 1 30, 2014, from http://www.prenhall.com/heartlungsounds/main/heart/s1s2.htm

Murphy, S. L., Xu, J. M., & Kochanek, K. D. (2013). Deaths: Final Data for 2010. *National Vital Statistics Reports, 61*(4).

National Heart, Lung, and Blood Institute. (2012, 9 20). *What is a Heart Murmur?* (National Institutes of Health) Retrieved 1 30, 2014, from http://www.nhlbi.nih.gov/health/health-topics/topics/heartmurmur/

National Telehealth TechnologyAssessment Resource Center. (n.d.). *TelehealthTechnology*. Retrieved from http://telehealthtechnology.org/: http://telehealthtechnology.org/toolkits/electronic-stethoscopes/about-electronic-stethoscopes/technology-overview

Steinhardt, A. O., & Van Veen, B. D. (1989). Adaptive Beamforming. *Internation Journal of Adaptive Control and Signal Processing*, 253-281.

Stethographics inc. . (n.d.). *Multi-Channel Stethograph.* Retrieved from http://www.stethographics.com/publications/Multinfo.pdf

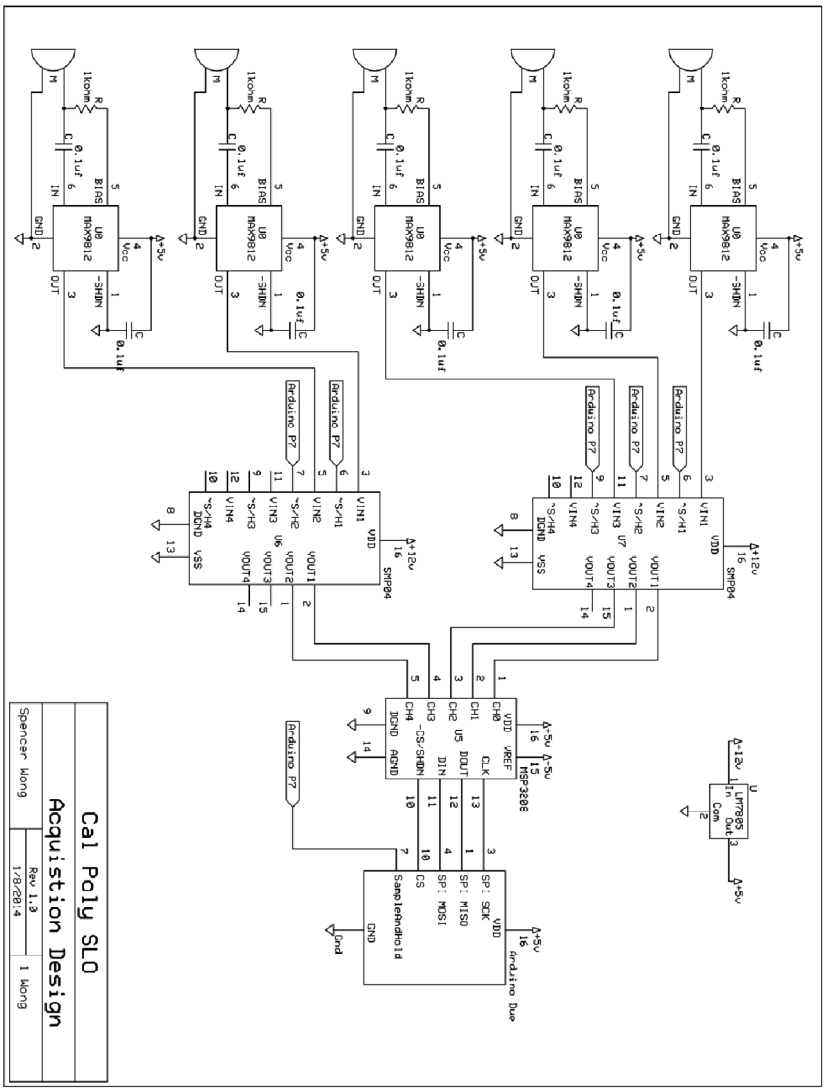
Synnevåg, J.-F., Austeng, A., & Holm, S. (2007). Adaptive Beamforming Applied to Medical Ultrasound Imagaing. *IEEE Transaction on Ultrassonics, Ferroelectrics, and Frequency Control*, 1606-1613.

Van Veen, B. D., & Buckley, K. M. (1988). Beamforming: A Versatile Approach to Spatial Filtering. *IEEE ASSP Magazine*, 4-24.

VAVRDA, M. (n.d.). Digital beamforming in wireless communication. 430 to 433.

www.forusdocs.doc. (n.d.). Retrieved from Forusdocs: http://www.forusdocs.com/reviews/images/Power_Spectra/Power_Spectra_Mike_Classic_II_SE.jpg

Appendix A Data Acquisition Circuit Design



Appendix B Bill Of Materials

Electronic Parts			
Qty	Description	Individual Price	
5	3M Littmann - Classic II S.E. Stethoscope	71.99	359.95
5	Microphone (Digikey PN: 668-1391-ND)	2.28	11.4
5	Microphone Amplfier MAX9812	0.91	4.55
5	Proto Board Adapter for SOT363 (For MAX9812)	1.61	8.05
2	IC OP AMP Sample and Hold (SMP04EPZ)	10.4	20.8
1	IC ADC (MCP3208)	4.18	4.18
1	Arduino Due	49.95	49.95
5	1 Kohm Resistor	0.1	0.5
15	0.1 uF	0.25	3.75
1	LM7805	0.95	0.95
5	1000 uF	0.35	1.75
2	Universal Component PC Board with 780 Holes	3.49	6.98
1	Grid-Style PC Board with 2200 Holes	4.49	4.49
			477.3

Mechanical Apparatus			
Qty	Description	Individual Price	
5	1-1/2 to 1-1/4 PVC Pipe Reducer Sch. 40	2	10
5	1-1/4 to 1 PVC Pipe Reducer Sch. 40	2	10
5	1 to 3/4 PVC Pipe Reducer Sch. 40	2	10
5	1 1/2 PVC 5in Pipe Sch. 40	2	10
1	10 - 3/4 PVC Pipe Holder	10	10
1	Scews	3	3
1	Bolts	3	3
1	Nuts	3	3
1	Washer	3	3
1	Wood	10	10
3	Stand Offs	2	6
		Total	78

Appendix C Procedures to Collecting Data from the Body

Procedures Collecting Data from the Body

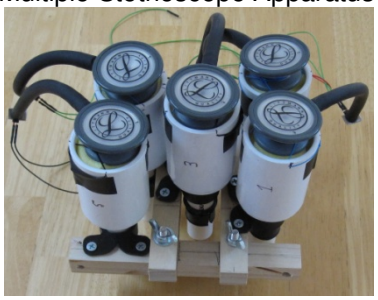
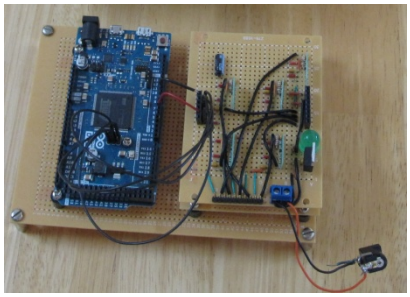
List of Equipment	
Multiple Stethoscope Apparatus  Figure C.1 Multiple Stethoscope Apparatus	Electronic Hardware  Figure C.2 Electronic Hardware
USB Cable	MATLAB R2013B
7 – Elastic Straps	Computer

Table C.1 List of Equipment

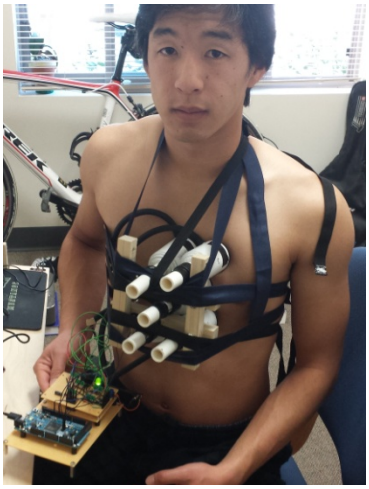


Figure C.3 Multiple Stethoscope Apparatus Setup

1. Setup Multiple Stethoscope Apparatus

   a. Wire the stethoscope apparatus to the electronic hardware. Table C.2 Important parts of the Multiple Stethoscope apparatus and the Electronic Hardware shows the parts that need to be connected between the electronic hardware and the multiple stethoscope apparatus.

      i. Figure C.6 shows the pin out of the electronic hardware.

      ii. Figure C.5 shows the pin out of the microphone attached to the tube which is attached to the stethoscope.

      iii. Figure C.4 shows the numbering of the stethoscope

Parts to connecting the Stethoscope apparatus to Electronic Hardware

Multiple Stethoscope Apparatus Number

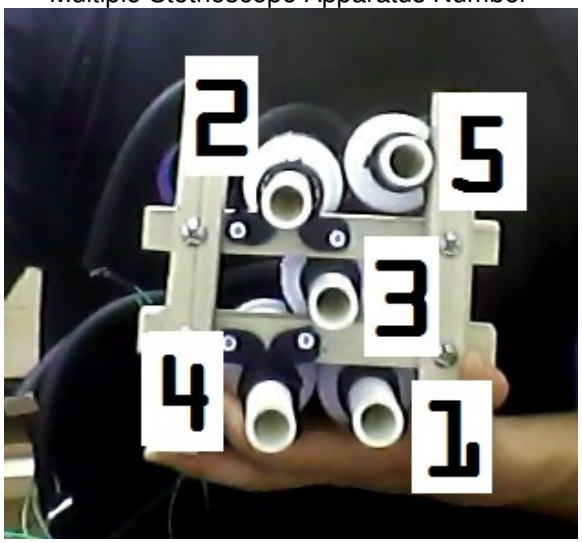


Figure C.4 Stethoscope Numbered

Microphone Output

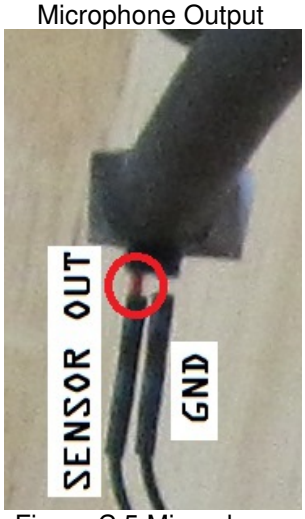


Figure C.5 Microphone attached to the Tube Pin Out

Electronic Hardware Pin Out

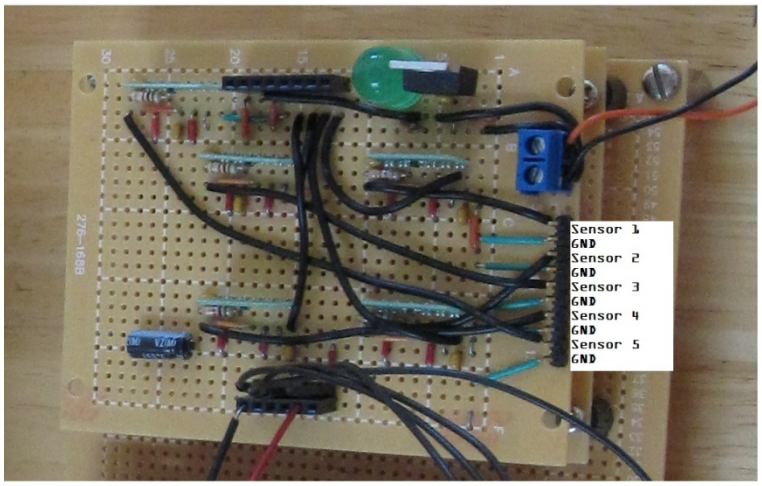


Figure C.6 Electronic Hardware Pin out to Stethoscope Apparatus

Table C.2 Important parts of the Multiple Stethoscope apparatus and the Electronic Hardware

2. Connect the Arduino to the computer by USB Cable.

    a. Check what port the Arduino is connected by going to Control Panel=>Hardware and Sound=>Devices and Printers

3. Open MATLAB

    a. Open the MATLAB sript CollectData.m (Shown in 0)

    b. Check that 'addpath' is reference to the right library (Appendix G).

```
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');
```

    c. Check if the Serial Command is correct

```
    arduinoDue = serial('COM11','BaudRate',57600,'DataBits',8
,'Parity','none');
```

4. Mount the Stethoscope Apparatus to the human specimen. (Review Table 4.4 for the place to put the stethoscope.)

    a. Use the 7 elastic straps to mount the Stethoscope Apparatus to the Chest. Refer to Appendix D on how to do this.

5. Run the MATLAB Code from Appendix F.

a. Use the command prompt for the rest of collecting data.

Appendix D Method of Mounting Stethoscope Apparatus

One of the issues that I had with this project was figuring out how to mount all 5 stethoscopes to the chest. Initially I thought that I could simply hold the stethoscopes apparatus to my chest while observing the heart sounds. But when I did that, I noticed that I got a very noisy signal. After some experimentation, I realized that the reason why the signals were so noisy was because I generated noise when holding the stethoscopes to the chest. The noise was generated either by my muscles straining to hold the stethoscope apparatus, or my arms shaking while holding the stethoscope. To solve this problem, I used elastic straps to hold the stethoscope apparatus to my chest while collecting data. Ultimately, I used seven elastic straps to hold the stethoscope apparatus to my chest as shown **Error! Reference source not found.Error! Reference source not found.** in Figure D.1. The straps were arranged to have all 5 stethoscopes make solid contact with the body.

Figure D.1 Multiple Stethoscope Apparatus Setup

Appendix E Microcontroller Collect Data Code

```
#include <SPI.h>

#define SAMPLEANDHOLD_PIN 7
#define SPISELECT_PIN 10
#define LED_PIN 13
#define SAMPLING_INTERVAL 500
#define NUM_SAMPLES  9500  /*          Number of Samples       */
#define  NUM_SENSORS  5

int NUM_DATA_RX = 3*NUM_SENSORS;
int TOTAL_DATA = NUM_SAMPLES*NUM_DATA_RX;
int DATA_RX =  NUM_SAMPLES*NUM_SENSORS;
int samplecount = 0;
int incomingByte = 0;   // for incoming serial data

byte val1[NUM_SAMPLES][NUM_SENSORS];
byte val2[NUM_SAMPLES][NUM_SENSORS];
unsigned long timeval;
unsigned long sampleflagvalue;
boolean sampledoneflag = false;
boolean computeflag = false;
byte sensorcommand[5][3]  = {{0x06, 0x00, 00},
                        {0x06, 0x40, 00},
                        {0x06, 0x80, 00},
                        {0x06, 0xC0, 00},
                        {0x07, 0x00, 00}};

void setup()
{
  Serial.begin(57600);
  pinMode (SAMPLEANDHOLD_PIN, OUTPUT);
  pinMode (SPISELECT_PIN, OUTPUT);
  pinMode (LED_PIN, OUTPUT);

  digitalWrite(LED_PIN, HIGH);

  //  Setup SPI
  SPI.begin(SPISELECT_PIN);
  SPI.setClockDivider(SPISELECT_PIN, 84);

}

void loop()
{

        // send data only when you receive data:
        if (Serial.available() > 0)
        {
                incomingByte = Serial.read();

                if(incomingByte == 'l')
```

```
                        {
                          //  Collect Data

                          digitalWrite(LED_PIN, LOW);
                          sampleflagvalue = micros()+SAMPLING_INTERVAL;
                          while(samplecount < NUM_SAMPLES)
                          {
                                  timeval = micros();
                                  if(sampleflagvalue <= timeval)
                                  {
                                          sampleflagvalue = timeval + SAMPLING_INTERVAL;
                                          digitalWrite(SAMPLEANDHOLD_PIN, HIGH);
                                          delayMicroseconds(15);
                                          for(int i = 0; i < NUM_SENSORS; i++)
                                          {
                                            SPI.transfer(SPISELECT_PIN,
sensorcommand[i][0],SPI_CONTINUE);
                                            val1[samplecount][i] = SPI.transfer(SPISELECT_PIN,
sensorcommand[i][1],SPI_CONTINUE);
                                            val2[samplecount][i] = SPI.transfer(SPISELECT_PIN,
sensorcommand[i][2]);
                                          }
                                          digitalWrite(SAMPLEANDHOLD_PIN, LOW);

                                           samplecount++;

                                  }
                          }
                           digitalWrite(LED_PIN, HIGH);
                          //  Done Collecting Data
                          samplecount = 0;

                          //  output data through UART at Baudrate of 115200
                          for(int i = 0; i < NUM_SAMPLES; i++)
                          {
                                  Serial.print(i);
                                  Serial.print("\t");
                                  for(int j = 0; j < NUM_SENSORS; j++)
                                  {
                                          Serial.print(((val1[i][j]&0x0F)<<8)+val2[i][j],DEC);
                                          Serial.print("\t");
                                  }
                                  Serial.print("\n");
                          }
                        }
                        else
                        {
                        }
                }
        }
```

Appendix F MATLAB Collect Data Script

```
%   Collect Data from the body
%   This program will ask to collect data then display the data collected.
%   Then it will ask if you would like to save it.

addpath('D:\Thesis\MATLAB Simulation\thesisfunction');

%figurenum = 1;
numsamples = 9500;
numcolumns = 6;     %   index and 5 sensors
data = zeros(numsamples,numcolumns);

arduinoDue = serial('COM11','BaudRate',57600,'DataBits',8 ,'Parity','none');
fopen(arduinoDue);
pause(2)    %  give the controller time to boot up.

display('Start Collecting in 4 seconds');
for i = 1:4
    pause(1)
    display(num2str(4-i));
end

display('Arduino Due is Sampling');
fprintf(arduinoDue,'l');
%   Gather Data
pause(5)
    display('Arduino Due is Done Sampling');
    display('Arduino Due is transmitting Data');
for i = 1:numsamples
    inputData = strsplit(fscanf(arduinoDue))
    cellsize = size(inputData);

    for j = 1:cellsize(2)-1
        data(i,j) = str2double(inputData{j});
    end
end
   display('Done Getting Data');

data1(:,1:5) = normalize_average5(data(:,2:6));
index = data(:,1);

figurenum = plot5time(index, data1,figurenum);
axis5(data1,0,9500)

%   Save Data Collected
str = input('Save Data? (y/n)\n','s');
if str == 'y'
    nameoffile = input('Name of Data File\n','s');
    nameoffile_txt = [nameoffile, '.txt'];
    fileID = fopen(nameoffile_txt,'w');
    fprintf(fileID, '%f\t%f\t%f\t%f\t%f\t%f\n', data');
```

```
    fclose(fileID);
else
end


fclose(arduinoDue);
```

Appendix G MATLAB Functions

MATLAB Function

```
%%DisplayDatafunc.m
function [Data2,Yss, f, figure_num] = DisplayDatafunc(datafile,samplingfrequency, figure_num)
%   Data 1
[Data(:,1),Data(:,2),Data(:,3),Data(:,4),Data(:,5),Data(:,6)] = textread(datafile, '%d %d %d %d %d %d');
Data2 = Data;
Data2(:,2:6) = normalize_average5(Data(:,2:6));
plot5time(Data2(:,1), Data2(:,2:6),figure_num);
figure_num = figure_num+1;

[Y,Yss, f] = fftfunc5(Data2(:,2:6),samplingfrequency);


%%GetThedata.m
function [index, time,  dataout] = GetThedata(nameofFile,samplingfrequency)

[index,data(:,1),data(:,2),data(:,3),data(:,4),data(:,5)] = textread(nameofFile, '%f %f %f %f %f %f');
dataout = normalize_average5(data);
time = index * (1/samplingfrequency);
end


%%HeartSoundSNR.m
function [HeartsoundSNRout] = HeartSoundSNR(HSrange, Data)
rangesize = size(HSrange);
HeartsoundSNRout = zeros(rangesize(1),2);
   for i = 1:rangesize(1)
   HeartsoundSNRout(i,1) =
var(Data(HSrange(i,1):HSrange(i,2),1))^2/var(Data(HSrange(i,7):HSrange(i,8),1))^2;
   HeartsoundSNRout(i,2) =
var(Data(HSrange(i,5):HSrange(i,6),1))^2/var(Data(HSrange(i,7):HSrange(i,8),1))^2;
   end
end


%%HeartSoundSection.m
%   Range
S1min1 = 1600;
S1max1 = S1min1+300;
othermin1 = S1max1;
othermax1 = 2300;
S2min1 = othermax1;
S2max1 = S2min1+300;
othermin2 = S2max1;
othermax2 = 3400;

S1min2 = othermax2;
S1max2 = S1min2+300;
othermin3 = S1max2;
othermax3 = 4000;
S2min2 = othermax3;
```

```
S2max2 = S2min2+300;
othermin4 = S2max2;
othermax4 = 5300;

S1min3 = othermax4;
S1max3 = S1min3+300;
othermin5 = S1max3;
othermax5 = 5900;
S2min3 = othermax5;
S2max3 = S2min3+300;
othermin6 = S2max3;
othermax6 = 7100;

S1min4 = othermax6;
S1max4 = S1min4+300;
othermin7 = S1max4;
othermax7 = 7750;
S2min4 = othermax7;
S2max4 = S2min4+300;
othermin8 = S2max4;
othermax8 = 8900;

%%  Array Definition
%   1st column: S1min
%   2nd column: S1max
%   3rd column: Noise1min
%   4th Column: Noise1max
%   5th column: S2min
%   6th column: S2max
%   7th column: Noise2min
%   8th column: Noise2max


HeartSoundRange = [S1min1 S1max1 othermin1 othermax1 S2min1 S2max1 othermin2 othermax2;
                   S1min2 S1max2 othermin3 othermax3 S2min2 S2max2 othermin4 othermax4;
                   S1min3 S1max3 othermin5 othermax5 S2min3 S2max3 othermin6 othermax6;
                   S1min4 S1max4 othermin7 othermax7 S2min4 S2max4 othermin8 othermax8];



%%HeartSoundSection1.m
S1range = 350;
S2range = 300;
%   Range
S1min1 = 1000;
S1max1 = S1min1+S1range;
othermin1 = S1max1;
othermax1 = 2300;
S2min1 = othermax1;
S2max1 = S2min1+S2range;
othermin2 = S2max1;
othermax2 = 3400;
```

```
S1min2 = othermax2;
S1max2 = S1min2+S1range;
othermin3 = S1max2;
othermax3 = 4000;
S2min2 = othermax3;
S2max2 = S2min2+S2range;
othermin4 = S2max2;
othermax4 = 5300;

S1min3 = othermax4;
S1max3 = S1min3+S1range;
othermin5 = S1max3;
othermax5 = 5900;
S2min3 = othermax5;
S2max3 = S2min3+S2range;
othermin6 = S2max3;
othermax6 = 7100;

S1min4 = othermax6;
S1max4 = S1min4+S1range;
othermin7 = S1max4;
othermax7 = 7750;
S2min4 = othermax7;
S2max4 = S2min4+S2range;
othermin8 = S2max4;
othermax8 = 8900;

%%  Array Definition
%   1st column: S1min
%   2nd column: S1max
%   3rd column: Noise1min
%   4th Column: Noise1max
%   5th column: S2min
%   6th column: S2max
%   7th column: Noise2min
%   8th column: Noise2max


HeartSoundRange = [S1min1 S1max1 othermin1 othermax1 S2min1 S2max1 othermin2 othermax2;
                   S1min2 S1max2 othermin3 othermax3 S2min2 S2max2 othermin4 othermax4;
                   S1min3 S1max3 othermin5 othermax5 S2min3 S2max3 othermin6 othermax6;
                   S1min4 S1max4 othermin7 othermax7 S2min4 S2max4 othermin8 othermax8];


%%HeartSoundSectionSNR.m
%   Range
S1min1 = 1650;
S1max1 = 1900;
othermin1 = 2000;
othermax1 = 2250;
S2min1 = 2300;
S2max1 = 2500;
othermin2 = 2800;
```

```
othermax2 = 3300;
%   HeartSoundSectionSNR
S1min2 = 3450;
S1max2 = 3700;
othermin3 = 3800;
othermax3 = 4000;
S2min2 = 4100;
S2max2 = 4300;
othermin4 = 4500;
othermax4 = 5200;

S1min3 = 5350;
S1max3 = 5550;
othermin5 = 5650;
othermax5 = 5800;
S2min3 = 5950;
S2max3 = 6150;
othermin6 = 6300;
othermax6 = 7050;

S1min4 = 7200;
S1max4 = 7400;
othermin7 = 7450;
othermax7 = 7650;
S2min4 = 7800;
S2max4 = 8000;
othermin8 = 8150;
othermax8 = 8900;

%%  Array Definition
%   1st column: S1min
%   2nd column: S1max
%   3rd column: Noise1min
%   4th Column: Noise1max
%   5th column: S2min
%   6th column: S2max
%   7th column: Noise2min
%   8th column: Noise2max


HeartSoundRange = [S1min1 S1max1 othermin1 othermax1 S2min1 S2max1 othermin2 othermax2;
                   S1min2 S1max2 othermin3 othermax3 S2min2 S2max2 othermin4 othermax4;
                   S1min3 S1max3 othermin5 othermax5 S2min3 S2max3 othermin6 othermax6;
                   S1min4 S1max4 othermin7 othermax7 S2min4 S2max4 othermin8 othermax8];




%%HighlightSectionsfunc.m
function [] = HighlightSectionsfunc(HSrange,Data)
rangesize = size(HSrange);
for i = 1:rangesize(1)
highlight(Data, HSrange(i,1),HSrange(i,2),'b');
```

```
highlight(Data, HSrange(i,5),HSrange(i,6),'r');
highlight(Data, HSrange(i,3),HSrange(i,4),'g');
highlight(Data, HSrange(i,7),HSrange(i,8),'g');
end
end


%%HighlightSectionsfunc5.m
function [] = HighlightSectionsfunc5(HSrange,Data)
rangesize = size(HSrange);

for i = 1:rangesize(1)
highlight5(Data, HSrange(i,1),HSrange(i,2),'b');
highlight5(Data, HSrange(i,5),HSrange(i,6),'r');
highlight5(Data, HSrange(i,3),HSrange(i,4),'g');
highlight5(Data, HSrange(i,7),HSrange(i,8),'g');
end
end


%%Plot5time_title.m
subplot(5,1,1);
title('Sensor 1');
subplot(5,1,2);
title('Sensor 2');
subplot(5,1,3);
title('Sensor 3');
subplot(5,1,4);
title('Sensor 4');
subplot(5,1,5);
title('Sensor 5');


%%axis5.m
function [] = axis5(data,xmin,xmax)
ymax = max(max(data));
ymin = min(min(data));
subplot(5,1,1)
axis([xmin,xmax,ymin,ymax])
subplot(5,1,2)
axis([xmin,xmax,ymin,ymax])
subplot(5,1,3)
axis([xmin,xmax,ymin,ymax])
subplot(5,1,4)
axis([xmin,xmax,ymin,ymax])
subplot(5,1,5)
axis([xmin,xmax,ymin,ymax])
end


%%axisfunc.m
function [] = axisfunc(data,xmin,xmax)
ymax = max(max(data));
```

```
ymin = min(min(data));
plotsize = size(data);
   for i = 1:plotsize(2)
      subplot(plotsize(2),1,i)
      axis([xmin,xmax,ymin,ymax])
   end
end


%%clippingfuncfix.m
function [dataout] = clippingfuncfix(datain)
%%  Find the start and end index of the clipping
clippingindex = find(datain == 4092);
startendclippingindex(1,1) = clippingindex(1);
j = 1;
nextvalue = clippingindex(1) + 1;
for i = 1:size(clippingindex,1)
   if clippingindex(i) > nextvalue
      startendclippingindex(j+1,1) = clippingindex(i);
      startendclippingindex(j,2) = clippingindex(i-1);
      j = j + 1;
   else
   end
    nextvalue = clippingindex(i) + 1;
end
startendclippingindex(j,2) = clippingindex(size(clippingindex,1));

%%  improve clipping
for i = 1:size(startendclippingindex,1)
%   Peak of the curve
i1 = startendclippingindex(i,1)-3;
i2 = startendclippingindex(i,1)-1;
p1 = datain(startendclippingindex(i,1)-3);
p2 = datain(startendclippingindex(i,1)-1);

leftsideslope = (p1-p2)/(i1-i2);

clip_range = startendclippingindex(i,2)-startendclippingindex(i,1)+4;
numriseindex = int16(clip_range/4);
%   Peak
peak = double((leftsideslope*numriseindex) + datain(startendclippingindex(i,1)-1));
halfclip_range = clip_range/2;

if rem(halfclip_range,1) == 0
   leftside = int16(linspace(datain(startendclippingindex(i,1)-1),peak,halfclip_range));
   rightside = int16(linspace(peak,datain(startendclippingindex(i,2)+1),halfclip_range));
   %   Input Values
   datain(startendclippingindex(i,1)-1:startendclippingindex(i,1)+halfclip_range-2) = leftside;
   datain(startendclippingindex(i,1)+halfclip_range-2:startendclippingindex(i,2)+1) = rightside;
else
   halfclip_range = floor(halfclip_range);
   leftside = int16(linspace(datain(startendclippingindex(i,1)-1),peak,halfclip_range));
   rightside = int16(linspace(peak,datain(startendclippingindex(i,2)),halfclip_range+1));
```

```
    %   Input Values
    datain(startendclippingindex(i,1)-1:startendclippingindex(i,1)+halfclip_range-2) = leftside;
    datain(startendclippingindex(i,1)+halfclip_range-2:startendclippingindex(i,2)+1) = rightside;
    end
end
%%  Apply LowPass Filter
Hd = LowpassFilter;
datain_1 = filter(Hd.numerator,1,datain);
dataout = datain_1;
end


%%ffttfunc.m
function [Y,Yss,f] = fttfunc(data,Fs)
y = data;
[L,n] = size(data);
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);
Yss = 2*abs(Y(1:NFFT/2+1));
end


%%ffttfunc5.m
function [Y,Yss, f] = fttfunc5(data,Fs)
[Y(:,1),Yss(:,1),f] = fttfunc(data(:,1),Fs);
[Y(:,2),Yss(:,2),f] = fttfunc(data(:,2),Fs);
[Y(:,3),Yss(:,3),f] = fttfunc(data(:,3),Fs);
[Y(:,4),Yss(:,4),f] = fttfunc(data(:,4),Fs);
[Y(:,5),Yss(:,5),f] = fttfunc(data(:,5),Fs);
end


%%highlight.m
function [] = highlight(data,xmin,xmax,color)
valmax = max(max(abs(data)));
valmin = -valmax;
hold on;
X = [xmin,xmax];
Y = [valmax,valmax];
Base = valmin;
harea = area(X,Y,Base);
set(harea, 'FaceColor', color);
alpha(0.25);
hold off;
end


%%highlight2.m
function [] = highlight2(ymin,ymax,xmin,xmax,color)
hold on;
X = [xmin,xmax];
Y = [ymax,ymax];
```

```
Base = ymin;
harea = area(X,Y,Base);
set(harea, 'FaceColor', color);
alpha(0.75);
hold off;
end


%%highlight5.m
function [] = highlight5(data,xmin,xmax,color)

subplot(5,1,1);
highlight(data,xmin,xmax,color);
subplot(5,1,2);
highlight(data,xmin,xmax,color);
subplot(5,1,3);
highlight(data,xmin,xmax,color);
subplot(5,1,4);
highlight(data,xmin,xmax,color);
subplot(5,1,5);
highlight(data,xmin,xmax,color);
end

%%normalize5.m
function [dataout] = normalize5(data)
dataout = data;
dataout(:,3:7) = (data(:,3:7)-min(min(data(:,3:7))))/(max(max(data(:,3:7)))-min(min(data(:,3:7))));
%dataout = (data-min(min((data))))/(max(max(data))-min(min(data)));


%%normalize_average.m
function [out] = normalize_average(datain)
%   This function gets rid of the DC offset by subtracting the average of
%   the data

out = datain(:) - mean(datain);
val1 = rms(out);
out = out/val1;

end


%%normalize_average5.m
function [out] = normalize_average5(data)

out(:,1) = normalize_average(data(:,1));
out(:,2) = normalize_average(data(:,2));
out(:,3) = normalize_average(data(:,3));
out(:,4) = normalize_average(data(:,4));
out(:,5) = normalize_average(data(:,5));

end
```

```
function [out] = normalize_average5_improvement(data)

out(:,1) = data(:,1) - mean(data(:,1));
out(:,2) = data(:,2) - mean(data(:,2));
out(:,3) = data(:,3) - mean(data(:,3));
out(:,4) = data(:,4) - mean(data(:,4));
out(:,5) = data(:,5) - mean(data(:,5));

end

%%plot4freq.m
function [figure_num] = plot4freq(x, data, sensor_num, figure_num)

figure(figure_num);
titleofdiagram=sprintf('Recieved White Noise From Sensor %d',sensor_num);
suptitle(titleofdiagram);
hold on;

for i = 1: 4
   subplot(4,1,i)
   plot(x,data(:,i))
   %title('Body Sounds Signal 1')
   xlabel('frequency')

   ylabelText=sprintf('Test %d',i);
   ylabel({ylabelText;'|Y(f)|'})
end


figure_num = figure_num+1;
end


%%plot5.m
function [] = plot5(x, data, figure_num)

figure(figure_num);

subplot(5,1,1)
plot(x,data(:,1))
%title('Body Sounds Signal 1')
xlabel('time')
ylabel({'Sensor 1';'Normalized Value'})

subplot(5,1,2)
plot(x,data(:,2))
%title('Body Sounds Signal 2')
xlabel('time')
ylabel({'Sensor 2';'Normalized Value'})

subplot(5,1,3)
plot(x,data(:,3))
%title('Body Sounds Signal 3')
```

```
xlabel('time')
ylabel({'Sensor 3';'Normalized Value'})


subplot(5,1,4)
plot(x,data(:,4))
%title('Body Sounds Signal 4')
xlabel('time')
ylabel({'Sensor 4';'Normalized Value'})


subplot(5,1,5)
plot(x,data(:,5))
%title('Body Sounds Signal 5')
xlabel('time')
ylabel({'Sensor 5';'Normalized Value'})


%%plot5__ver2.m
function [] = plot5_ver2(x, data, xlab,ylab,figure_num)

figure(figure_num);

subplot(5,1,1)
plot(x,data(:,1))
%title('Body Sounds Signal 1')
xlabel(xlab)
ylabel(ylab)

subplot(5,1,2)
plot(x,data(:,2))
%title('Body Sounds Signal 2')
xlabel(xlab)
ylabel(ylab)

subplot(5,1,3)
plot(x,data(:,3))
%title('Body Sounds Signal 3')
xlabel(xlab)
ylabel(ylab)


subplot(5,1,4)
plot(x,data(:,4))
%title('Body Sounds Signal 4')
xlabel(xlab)
ylabel(ylab)


subplot(5,1,5)
plot(x,data(:,5))
%title('Body Sounds Signal 5')
xlabel(xlab)
```

```
ylabel(ylab)


%%plot5freq.m
function [figure_num] = plot5freq(x, data, figure_num)

figure(figure_num);

subplot(5,1,1)
plot(x,data(:,1))
xlabel('frequency')
ylabel({'Sensor 1';'|Y(f)|'})

subplot(5,1,2)
plot(x,data(:,2))
xlabel('frequency')
ylabel({'Sensor 2';'|Y(f)|'})

subplot(5,1,3)
plot(x,data(:,3))
xlabel('frequency')
ylabel({'Sensor 3';'|Y(f)|'})


subplot(5,1,4)
plot(x,data(:,4))
xlabel('frequency')
ylabel({'Sensor 4';'|Y(f)|'})


subplot(5,1,5)
plot(x,data(:,5))
xlabel('frequency')
ylabel({'Sensor 5';'|Y(f)|'})

figure_num = figure_num + 1;
end


%%plot5time.m
function [figure_num] = plot5time(x, data, figure_num)

figure(figure_num);

subplot(5,1,1)
plot(x,data(:,1))
%title('Body Sounds Signal 1')
xlabel('Index')
ylabel({'Sensor 1';'Value'})

subplot(5,1,2)
plot(x,data(:,2))
%title('Body Sounds Signal 2')
```

```
xlabel('Index')
ylabel({'Sensor 2';'Value'})

subplot(5,1,3)
plot(x,data(:,3))
%title('Body Sounds Signal 3')
xlabel('Index')
ylabel({'Sensor 3';'Value'})


subplot(5,1,4)
plot(x,data(:,4))
%title('Body Sounds Signal 4')
xlabel('Index')
ylabel({'Sensor 4';'Value'})


subplot(5,1,5)
plot(x,data(:,5))
%title('Body Sounds Signal 5')
xlabel('Index')
ylabel({'Sensor 5';'Value'})

figure_num = figure_num + 1;
%Plot5time_title


%%plotfreq.m
function [figure_num] = plotfreq(x, data, figure_num)

figure(figure_num);

plot(x,data(:,1))
%title('Body Sounds Signal 1')
xlabel('frequency')
ylabel('|Y(f)|')
figure_num = figure_num + 1;


%%shiftdata.m
function [DataOut] = shiftdata(DataIn,Delay)

    DataOut = circshift(DataIn,int16(Delay));
    DataOut(1:abs(int16(Delay)),1) = 0;

end


%%shiftdata5.m
function [DataOut] = shiftdata5(DataIn,Delay)

    DataOut(:,1) = shiftdata(DataIn(:,1),Delay(1));
    DataOut(:,2) = shiftdata(DataIn(:,2),Delay(2));
```

```
    DataOut(:,3) = shiftdata(DataIn(:,3),Delay(3));
    DataOut(:,4) = shiftdata(DataIn(:,4),Delay(4));
    DataOut(:,5) = shiftdata(DataIn(:,5),Delay(5));
end


%%thephasedelay.m
function [numshift, timedelay] = thephasedelay(data1,data2,samplingfreq)
sampletime = 1/samplingfreq;
c = xcorr(data1, data2);
centerofxcorr = size(c,1)/2;
maxc = max(c);
numshift = find(c == maxc)-centerofxcorr;
timedelay = (find(c == maxc)-centerofxcorr)*sampletime;
end


%%thephasedelay5.m
function [numshift1, timedelay1, numshift2, timedelay2,numshift3, timedelay3, numshift4, timedelay4] =
thephasedelay5(data,col1,col2,col3,col4,col5,freq)

[numshift1, timedelay1] = thephasedelay(data(:,col1),data(:,col2),freq);
[numshift2, timedelay2] = thephasedelay(data(:,col1),data(:,col3),freq);
[numshift3, timedelay3] = thephasedelay(data(:,col1),data(:,col4),freq);
[numshift4, timedelay4] = thephasedelay(data(:,col1),data(:,col5),freq);



%%thephasedelay5_1.m
function [timedelay, numshift] = thephasedelay5_1(data,col1,col2,col3,col4,col5,freq)

timedelay(col1) = 0;
numshift(col1) = 0;
[numshift(col2), timedelay(col2), numshift(col3), timedelay(col3),numshift(col4),
timedelay(col4),numshift(col5), timedelay(col5)] = thephasedelay5(data,col1,col2,col3,col4,col5,freq);

End

function [SNRout] = VSNR(section, HSrange, Data)
%   Calculate the Signal to noise ratio using the variance equation
s1num = Data(HSrange(section,1):HSrange(section,2));
s2num = Data(HSrange(section,5):HSrange(section,6));
den = Data(HSrange(section,7):HSrange(section,8));

as1num = var(s1num)^2;
as2num = var(s2num)^2;
aden = var(den)^2;

s1SNR = as1num/aden;
s2SNR = as2num/aden;

SNRout = [s1SNR s2SNR];
```

```
end


function [SNRout] = ASNR(section, HSrange, Data)
%%  calculating the Signal to Noise Ratio using Peak Amplitude equation
s1num = Data(HSrange(section,1):HSrange(section,2));
s2num = Data(HSrange(section,5):HSrange(section,6));
den = Data(HSrange(section,7):HSrange(section,8));

as1num = abs(s1num);
as2num = abs(s2num);
aden = abs(den);

s1SNR = (max(as1num)/max(aden))^2;
s2SNR = (max(as2num)/max(aden))^2;

SNRout = [s1SNR s2SNR];
end

function y = envelope(x)
%Envelope function using Hilbert transform.
%   HILBERT(X) is the Hilbert transform of the real part
%   of vector X.  The real part of the result is the original
%   real data; the imaginary part is the actual Hilbert
%   transform.  See also FFT and IFFT.
%
%   If X is a signal matrix, HILBERT(X) transforms the columns
%   of X independently.

%   Author(s): C. Denham, 1-7-88
%          L. Shure, 11-19-88, 5-22-90 - revised
%          K. Creager, 3-19-92, modified to use power of 2 FFT
%          T. Krauss, 11-4-92, revised
%   Copyright (c) 1988-97 by The MathWorks, Inc.
%   $Revision: 1.1 $  $Date: 1999-07-15 15:35:42-04 $

%   Reference(s):
%     [1] Jon Claerbout, Introduction to Geophysical Data Analysis.

[r,c] = size(x);
if r == 1
    x = x.';   % make it a column
end;

% Remove DC offset
x = x-mean(x);

[n,cc] = size(x);
m = 2^nextpow2(n);
y = fft(real(x),m);
if m ~= 1
   h = [1; 2*ones(fix((m-1)/2),1); ones(1-rem(m,2),1); zeros(fix((m-
```

```
1)/2),1)];
   y(:) = y.*h(:, ones(1,cc) );
end
y = ifft(y,m);
y = y(1:n,:);
if r == 1
   y = y.';
end

% Find the envelop
y = abs(y);
```

Appendix H MATLAB Characterization Collect Data Impulse Script

Generate_Impulse

```
%   Generate Impulse Sound
t = (0:1/44100:4)';
impulsesignal = zeros(size(t));
impulsesignal(50000,1) = 1;

figure(1);
plot(t,impulsesignal);
ylabel('Amplitude');
xlabel('Time (s)');
suptitle('Speaker Output Signal');
saveas(1,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\Pictures\impulsetest\SpeakerOutputSignal.jpg');
```

Impulse_AirTest

```
%   Impulse Test
Generate_Impulse;  %   Generates a sound signal
numtest = 10;
numsamples = 9500;
numcolumns = 6;     %   index and 5 sensors
data = zeros(numsamples,numcolumns, numtest);
f = 0:1000/9499:1000;
arduinoDue = serial('COM11','BaudRate',57600,'DataBits',8 ,'Parity','none');
fopen(arduinoDue);
pause(2)    %  give the controller time to boot up.
%   Test
for l = 1:numtest
displaystring = ['Number Test ', num2str(l)];
display(displaystring);
soundsc(impulsesignal,44100);
pause(0.298);
fprintf(arduinoDue,'l');
%   Gather Data
pause(5)
   for i = 1:numsamples
      inputData = strsplit(fscanf(arduinoDue));
      cellsize = size(inputData);
      for j = 1:cellsize(2)-1
         data(i,j,l) = str2double(inputData{j});
      end
   end
end
% plot5time(data(:,1), data(:,2:6)    , figurenum);
% figurenum = figurenum+1;
fclose(arduinoDue);
```

Appendix I Procedure to Characterizing the Stethoscope Apparatus

Procedures

1. Open MATLAB

a. Generate Sound
    i. Impulse

MATLAB Command

t = (0:1/44100:4)';
impulsesignal = zeros(size(t));
impulsesignal(50000,1) = 1;

    ii. Chirp

MATLAB Command
t = (0:1/44100:4.75)'; chirpsignal=chirp(t,0,4.75,1000);

    iii. White Noise

MATLAB Command
whitenoise = wgn(1000000,1,1);

2. Set up Speakers
    a. Connect Auxiliary Cable from Computer to Speakers
    b. Test Speakers
        i. Impulse

MATLAB Code
soundsc(impulsesignal,44100)

        ii. Chirp

MATLAB Code
soundsc(chirpsignal,44100)

        iii. White Noise

MATLAB Code
soundsc(whitenoise,44100)

3. Set up Multiple Stethoscope Apparatus

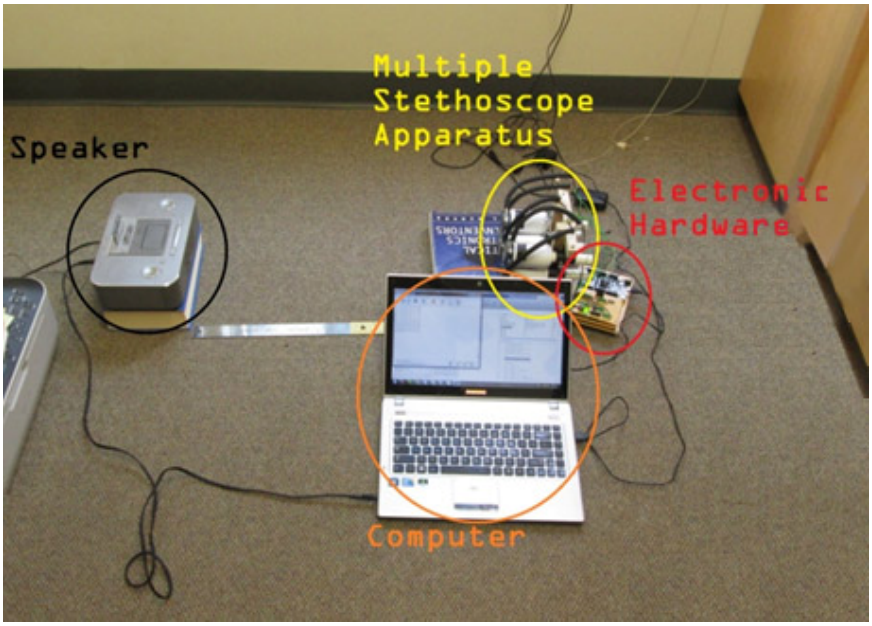


Figure I.1 Characterization Test Setup

a. Place the stethoscope facing the speaker at about a foot away.
b. Hook up the stethoscope apparatus to the Electronic Hardware that will be used to collect the data.
c. Connect the micro USB into the Arduino Due.
d. Connect a 9 to 12 V DC Power supply into the Electronic Hardware

4. Collect Data
    a. Run MATLAB Code to start collecting Data.
        i. Impulse Test
            1. Refer to Appendix H
        ii. Chirp Test
            1. Refer to Appendix K
        iii. White Noise Test
            1. Refer to Appendix M

Appendix J MATLAB Characterization Analysis Impulse Script

Impulseanalysis

```
load('ImpulseAirData.mat')
Fs = 2000;
onesdata = ones(size(data(:,2:6,1)));

for i = 1: 10

    datawork(:,:,i) = data(:,2:6,i);
    data_mean1 = mean(datawork(:,:,i),1);
    data_mean = onesdata(:,1)*data_mean1;
    norm_data(:,:,i) = datawork(:,:,i)-data_mean;

end
norm_data1 = norm_data(2200:2400,:,1);
figure;
plot(norm_data1);
xlabel('Time (s)');
ylabel('Amplitude');
title('Impulse Test Results');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
maxvalue = zeros(5,3);
for i = 1:5
    maxvalue(i,1) = max(norm_data1(:,i));

end

for i = 1:5
    maxvalue(i,2) = max(norm_data1(40:44,i));

end

maxvalue(:,3) = maxvalue(1,1)./maxvalue(:,1);
maxvalue(:,4) = maxvalue(1,2)./maxvalue(:,2);
```

PeakDifferenceInterpolated.m

```
clear
load('ImpulsePeakData.mat')
Fs = 2000;

figure(1);
plot(norm_data2);

norm_data2size = size(norm_data2);
endtime = norm_data2size(1) * 1/Fs;



upsampleval = 300;
Fsup = upsampleval * Fs;
```

```
interdata = zeros(upsampleval*norm_data2size(1), 5);


for i = 1:5
    interdata(:,i) =   interp(norm_data2(:,i),upsampleval);
end

interdata_size = size(interdata);
interdata_time = (0:1/Fsup:endtime-(1/Fsup))';
figure(2);
plot(interdata_time,interdata);
title('Upsampled Impulse Results');
xlabel('Index');
ylabel('Amplitude')

xmin1 = 0.015; xmax1 = 0.035; ymin1 = -1; ymax1 = 1;
axis([xmin1 xmax1 ymin1 ymax1]);

interdata_max = max(interdata)';
```

ShowImpulseResults.m

```
load('ImpulseAirData.mat')

Fs = 2000;
onesdata = ones(size(data(:,2:6,1)));


for i = 1: 10

    datawork(:,:,i) = data(:,2:6,i);
    data_mean1 = mean(datawork(:,:,i),1);
    data_mean = onesdata(:,1)*data_mean1;
    norm_data(:,:,i) = datawork(:,:,i)-data_mean;

end


    norm_data1 = norm_data(2200:2400,:,1);

%     figure;
%     plot(norm_data1);
%     xlabel('Time (s)');
%     ylabel('Normalized Amplitude Around Max Value');
%     title('Impulse Test Results');
%     legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor
5');
```

```
    max_norm_data1 = max(max(abs(norm_data1)));

    norm_data2 = (norm_data1)/(max_norm_data1);



    time =  (1:1:size(norm_data2,1)) * (1/Fs);

    figure(10);
    plot(time, norm_data2);

    xlabel('Time (s)');
    ylabel('Normalized Amplitude Around Max Value');
    title('Impulse Test Results');
    legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
    xmin1 = 0.02; xmax1 = 0.023; ymin1 = -.04; ymax1 = .1;
    axis([xmin1 xmax1 ymin1 ymax1]);
    %   Adjust the amplitude


    figure(11);
    plot(time, norm_data2);
    xlabel('Time (s)');
    ylabel('Normalized Amplitude Around Max Value');
    title('Impulse Test Results');
    legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
    xmin2 = 0.0252; xmax2 = 0.0266; ymin2 = 0; ymax2 = 1;
    axis([xmin2 xmax2 ymin2 ymax2]);


    figure(12);
    plot(time, norm_data2);
    xlabel('Time (s)');
    ylabel('Normalized Amplitude Around Max Value');
    title('Impulse Test Results');
    legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
    xmin3 = 0.015; xmax3 = 0.05; ymin3 = -0.4; ymax3 = 1;
    axis([xmin3 xmax3 ymin3 ymax3]);

    rectangle('Position',[xmin1,ymin1,xmax1-xmin1,ymax1-ymin1]);
    text(xmin1,ymin1-0.01,'Plot 1');

    rectangle('Position',[xmin2,ymin2,xmax2-xmin2,ymax2-ymin2]);
    text(xmin2,ymin2-.01,'Plot 2');


    for i = 10:12
        imagename =
['D:\Thesis\Pictures\impulsetest\ImpulseResponseResult', num2str(i-
9),'.jpg'];
        saveas(i,imagename);
```

```
    end
```

Spline.m

```
% Spline

clear
load('ImpulsePeakData.mat')
Fs = 2000;

figure(1);
plot(norm_data2);

norm_data2size = size(norm_data2);
endtime = norm_data2size(1) * 1/Fs;

%   Normalize
maxvalue = max(norm_data2);
norm_data2(:,1) = norm_data2(:,1)/maxvalue(1);
norm_data2(:,2) = norm_data2(:,2)/maxvalue(2);
norm_data2(:,3) = norm_data2(:,3)/maxvalue(3);
norm_data2(:,4) = norm_data2(:,4)/maxvalue(4);
norm_data2(:,5) = norm_data2(:,5)/maxvalue(5);

time = 0:1/2000:(endtime-(1/2000));

upsampleval = 300;
Fsup = upsampleval * Fs;


interdata = zeros(upsampleval*norm_data2size(1), 5);

interdata_time = (0:1/Fsup:endtime-(1/Fsup))';

for i = 1:5
    interdata(:,i) =
interp1(time,norm_data2(:,i),interdata_time,'spline');
end




intermax = max(interdata);

for i = 1:5
    interdata(:,i) = interdata(:,i)/intermax(i);
end
```

```
interdata_size = size(interdata);
interdata_time = (0:1/Fsup:endtime-(1/Fsup))';
figure(2);
plot(interdata_time,interdata);
title('Upsampled Impulse Results');
xlabel('Index');
ylabel('Amplitude')
xmin1 = 0.015; xmax1 = 0.035; ymin1 = -1; ymax1 = 1;
axis([xmin1 xmax1 ymin1 ymax1]);

legend('Sensor 1','Sensor 2','Sensor 3','Sensor 4','Sensor 5');

figure(3);
plot(interdata_time,interdata);
title('Upsampled Impulse Results');
xlabel('Index');
ylabel('Amplitude')
xmin1 = 0.0253; xmax1 = 0.0257; ymin1 = .9; ymax1 = 1.02;
axis([xmin1 xmax1 ymin1 ymax1]);

legend('Sensor 1','Sensor 2','Sensor 3','Sensor 4','Sensor 5');

[y,i] = max(interdata);

phasedelay = i*1/Fsup;
phasedelay = phasedelay-phasedelay(3);

saveas(2,'C:\Users\Spencer Wong\Documents\My Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\figures\impulseSplineResult.jpg');
saveas(3,'C:\Users\Spencer Wong\Documents\My Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\figures\impulseSplineResult1.jpg');
```

Appendix K MATLAB Characterization Collect Data Chirp Script

GenerateChirp

```
%   Generate chirp
t = (0:1/44100:4.75)';
chirpsignal=chirp(t,0,4.75,1000);

%
% %   Frequency Domain
% [Y,Yss,f] = fftfunc(chirpsignal,44100);
% plotfreq(f, Yss, 40);
% title('Input Signal Spectrum');
% textfilename = 'D:\Thesis\MATLAB Simulation\Chirp_AirTest\InputSignalFreq.jpg';
% saveas(40,textfilename)
% % ymin = 0;
% % ymax = max(Yss);
% % axis([0,20000, ymin,ymax])
%
%   Time Domain
% fignum = figure;
% plot(t,chirpsignal);
% ymin = min(chirpsignal);
% ymax = max(chirpsignal);
% % axis([0,.01, ymin,ymax])
% ylabel('Amplitude');
% xlabel('Time');
% suptitle('Input Signal');
% textfilename = 'D:\Thesis\MATLAB Simulation\Chirp_AirTest\InputSignal.jpg';
% saveas(fignum,textfilename)
%
% plot(t,chirpsignal);
% xlabel('time (s)');
% ylabel('Amplitu

%  Time Domain
fignum = figure;
plot(log(t),chirpsignal);
ymin = min(chirpsignal);
ymax = max(chirpsignal);
% axis([0,.01, ymin,ymax])
ylabel('Amplitude');
xlabel('Time');
suptitle('Input Signal');
textfilename = 'D:\Thesis\MATLAB Simulation\Chirp_AirTest\InputSignal.jpg';
saveas(fignum,textfilename)
```

Chirp_AirTest.m

```
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');
%   Chirp Test
GenerateChirp;  %   Generates a sound signal
numtest = 5;
numsamples = 9500;
numcolumns = 6;     %   index and 5 sensors
data = zeros(numsamples,numcolumns, numtest);

figurenum = 1;
arduinoDue = serial('COM11','BaudRate',57600,'DataBits',8 ,'Parity','none');
fopen(arduinoDue);
pause(2)    %  give the controller time to boot up.
%   Test
for l = 1:numtest   %   Loops for the number of test
    displaystring = ['Number Test ', num2str(l)];
    display(displaystring);
    soundsc(chirpsignal,44100);
    pause(0.298);
    fprintf(arduinoDue,'l');
    %   Gather Data
    pause(5)
        for i = 1:numsamples    %   Loops until all the data from the
                                %   arduino has been recieved
        inputData = strsplit(fscanf(arduinoDue));
        cellsize = size(inputData);

            for j = 1:cellsize(2)-1  %  Transfer data to array
                data(i,j,l) = str2double(inputData{j});
            end
        end
    plot5freq(f, data(:,2:6,l), figurenum); %   Generate a graph of the
                                        %   data collected
    figurenum = figurenum+1;
end

fclose(arduinoDue);
```

GenerateChirp3.m

```
%   Generate chirp
freqlim = 100;
t = (0:1/44100:4.75)';
chirpsignal=chirp(t,0,4.75,freqlim);

% %   Frequency Domain
% [Y,Yss,f] = fftfunc(chirpsignal,44100);
% plotfreq(f, Yss, 40);
% title('Input Signal Spectrum');
% textfilename = 'D:\Thesis\MATLAB Simulation\Chirp_AirTest\InputSignalFreq.jpg';
% saveas(40,textfilename)
% % ymin = 0;
% % ymax = max(Yss);
% % axis([0,20000, ymin,ymax])
%
%   Time Domain
% fignum = figure;
% plot(t,chirpsignal);
% ymin = min(chirpsignal);
% ymax = max(chirpsignal);
% % axis([0,.01, ymin,ymax])
% ylabel('Amplitude');
% xlabel('Time');
% suptitle('Input Signal');
% textfilename = 'D:\Thesis\MATLAB Simulation\Chirp_AirTest\InputSignal.jpg';
% saveas(fignum,textfilename)
```

Chirp_AirTest3.m

```
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');

%   Chirp Test
GenerateChirp3;  %   Generates a sound signal
numtest = 5;
numsamples = 9500;
numcolumns = 6;     %   index and 5 sensors
data = zeros(numsamples,numcolumns, numtest);
f = 0:freqlim/9499:freqlim;
arduinoDue = serial('COM11','BaudRate',57600,'DataBits',8 ,'Parity','none');
fopen(arduinoDue);
pause(2)    %  give the controller time to boot up.
%   Test
for l = 1:numtest   %   Loops for the number of test
    displaystring = ['Number Test ', num2str(l)];
    display(displaystring);
    soundsc(chirpsignal,44100);
    pause(0.298);
    fprintf(arduinoDue,'l');
    %   Gather Data
    pause(5)
        for i = 1:numsamples    %   Loops until all the data from the
                                %   arduino has been recieved
        inputData = strsplit(fscanf(arduinoDue))
        cellsize = size(inputData);

            for j = 1:cellsize(2)-1  %  Transfer data to array
                data(i,j,l) = str2double(inputData{j});
            end
        end
    plot5freq(f, data(:,2:6,l), figurenum); %   Generate a graph of the
                                    %   data collected
    figurenum = figurenum+1;
end

fclose(arduinoDue);
```

Appendix L MATLAB Characterization Analysis Chirp Script

ChirpAnalysis

```
load('ChirpData3_6_2014');

f = 0:1000/9499:1000;
onesdata = ones(size(data(:,2:6,1)));
shiftval = [179; 152; 154; 29;28];

fignum = 11;
for i = 1: 5
   datawork(:,:,i) = data(:,2:6,i);
   data_mean1 = mean(datawork(:,:,i),1);
   data_mean = onesdata(:,1)*data_mean1;
   norm_data(:,:,i) = abs(datawork(:,:,i)-data_mean);
   shiftdata(:,:,i) = circshift(norm_data(:,:,i),shiftval(i));
   shiftdata(1:shiftval(i),:,i) = 0;
end

meandata = mean(shiftdata,3);
movavefilter = [1/24;repmat(1/12,11,1);1/24];

for i = 1:5
meandata2(:,i) = conv(meandata(:,i),movavefilter,'same');
end

fignum = plot5freq(f,meandata2,fignum);
axis5(meandata2,0,1000);
suptitle('Frequency Response from Chirp Sound');
textfilename=strcat('D:\Thesis\Pictures\ChirpTest\ChirpTestFreqMeanResultSensor.jpg');
saveas(fignum-1,textfilename);
```

chirpAnalysis0to100

```
load('ChirpData0to100Hz_3_9_2014');
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');

freqlim = 100;
f = 0:freqlim/9499:freqlim;
onesdata = ones(size(data(:,2:6,1)));
shiftval = [172; 187; 188; 168;182];

fignum = 11;
for i = 1: 5
   datawork(:,:,i) = data(:,2:6,i);
   data_mean1 = mean(datawork(:,:,i),1);
   data_mean = onesdata(:,1)*data_mean1;
   norm_data(:,:,i) = abs(datawork(:,:,i)-data_mean);
   shiftdata(:,:,i) = circshift(norm_data(:,:,i),shiftval(i));
   shiftdata(1:shiftval(i),:,i) = 0;
end

meandata = mean(shiftdata,3);
 fignum = plot5time(data(:,1,1),meandata,fignum);
movavefilter = [1/320;repmat(1/160,159,1);1/320];

for i = 1:5
meandata2(:,i) = conv(meandata(:,i),movavefilter,'same');
end

fignum = plot5freq(f,meandata2,fignum);
axis5(meandata2,0,freqlim);
suptitle('Frequency Response from Chirp Sound');
textfilename=strcat('D:\Thesis\Pictures\ChirpTest\ChirpTestlowFreqMeanResultSensor.jpg');
saveas(fignum-1,textfilename);

fignum = fignum+1;

figure(fignum);
fignum = fignum+1;
plot(f,meandata2);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
suptitle('Frequency Response from Low Frequency Chirp Sound');
textfilename=strcat('D:\Thesis\Pictures\ChirpTest\ChirpTestlowFreqMeanResult2Sensor.jpg');
saveas(fignum-1,textfilename);
%  xmin1 = 0; xmax1 = 40; ymin1 = 0; ymax1 = 3;
%  axis([xmin1 xmax1 ymin1 ymax1]);
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5','Location','NorthWest');

textfilename=strcat('D:\Thesis\Pictures\ChirpTest\ChirpTestlowFreqMeanResult3Sensor.jpg');
saveas(fignum-1,textfilename);

%%  Get mean between frequency 10 to 30 Modification
Data10to30Hz= meandata2(950:2850,:);
```

```
freq10to30Hz = f(950:2850);
Data10to30HzMean = mean(Data10to30Hz);

gain = repmat((1./Data10to30HzMean),1901,1);
Data10to30HzGain = Data10to30Hz.*gain;

plot(freq10to30Hz, Data10to30HzGain);
axis([10 30 .5 1.5]);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
suptitle('Frequency Response with Gain');

legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5','Location','NorthWest');


gain = repmat((1./Data10to30HzMean),size(meandata2,1),1);
DataGainAdjusted = meandata2 .* gain;

figure(fignum);
fignum = fignum +1;
plot(f, DataGainAdjusted);
axis([0 50 0 10]);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
suptitle('Frequency Response with Adjusted Gain');

legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5','Location','NorthWest');


figure(fignum);
fignum = fignum +1;
plot(f, DataGainAdjusted);
axis([0 90 0 120]);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
suptitle('Frequency Response with Adjusted Gain');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5','Location','NorthWest');
```

HilbertScript.m

```
load('ChirpData3_6_2014');


f = 0:1000/9499:1000;
onesdata = ones(size(data(:,2:6,1)));
shiftval = [179; 152; 154; 29;28];

fignum = 11;
for i = 1: 5

    datawork(:,:,i) = data(:,2:6,i);
    data_mean1 = mean(datawork(:,:,i),1);
```

```
    data_mean = onesdata(:,1)*data_mean1;
    %norm_data(:,:,i) = abs(datawork(:,:,i)-data_mean);
    % Do not take the abs value yet.  It will throw off the envelop

    norm_data(:,:,i) = (datawork(:,:,i)-data_mean);
   shiftdata(:,:,i) = circshift(norm_data(:,:,i),shiftval(i));
    shiftdata(1:shiftval(i),:,i) = 0;

end

%meandata = mean(shiftdata,3);
% Take the average of the envelopes; not the average of the chirp data
%HIL = meandata(1426:2281,:);
HIL = shiftdata(1426:2281,:,:);
HIL = shiftdata;

%HIL2 = zeros(856,5);
HIL2 = zeros(856,5,5);% Accommodate a 3-D matrix.  Collapse to 2-D
later
HIL2 = zeros(size(HIL));

% ADDED outer loop to compute envelope of each of 5 trials
for ii = 1:5,  % envelope each trial before averaging
for i = 1:5
    %HIL2(:,i) = envelope(HIL(:,i));
    HIL2(:,i,ii) = envelope(HIL(:,i,ii));

end
end;

% NEW ADDITION: Average AFTER enveloping
HIL = mean(HIL,3);
HIL2 = mean(HIL2,3)
%

figure(30);
plot(HIL);
title('Averaged Data');
xlabel('Index');
ylabel('Amplitude');

figure(31);
plot(abs(HIL));
title('Abs Value of Averaged Data');
xlabel('Index');
ylabel('Amplitude');


figure(32);
plot(HIL2);
title('Averaged Hilbert Transform Result');
```

```
xlabel('Index');
ylabel('Amplitude');

% Smooth the Hilbert envelope
SmoothHIL2=zeros(size(HIL2,1)+ 9, size(HIL2,2));
for i = 1:5,
SmoothHIL2(:,i)=conv(HIL2(:,i),0.1.*ones(10,1));
end;
SmoothHIL2=SmoothHIL2(6:size(SmoothHIL2,1)-4,:);


figure(33);
plot(SmoothHIL2);
title('Smoothed Hilbert Transform Result');
xlabel('Index');
ylabel('Amplitude');
```

GenerateGraphswithSmoothhil.m

```
load('smoothil2');
foldername = 'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\Pictures\ChirpTest\';
fignum = 1;

f = 0:1000/9499:1000;
fignum = figure(fignum);



plot(f,SmoothHIL2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal Plot
1');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
xmin1 = 270; xmax1 = 295; ymin1 = 100; ymax1 = 1100;
axis([xmin1 xmax1 ymin1 ymax1]);

textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
fignum = fignum+1;

fignum = figure(fignum);
plot(f,SmoothHIL2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal Plot
2');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
xmin2 = 290; xmax2 = 350; ymin2 = 100; ymax2 = 1150;
axis([xmin2 xmax2 ymin2 ymax2])



textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
fignum = fignum+1;

fignum = figure(fignum);
plot(f,SmoothHIL2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal Plot
3');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
xmin3 = 150; xmax3 = 240; ymin3 = 100; ymax3 = 500;
axis([xmin3 xmax3 ymin3 ymax3])



textfilename=
```

```
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
fignum = fignum+1;

fignum = figure(fignum);
plot(f,SmoothHIL2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal
Overlapped');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
%   Draw Rectangle
rectangle('Position',[xmin1,ymin1,xmax1-xmin1,ymax1-ymin1]);
text(xmin1,ymax1+10,'Plot 1');

rectangle('Position',[xmin2,ymin2,xmax2-xmin2,ymax2-ymin2]);
text(xmin2,ymax2+10,'Plot 2');

rectangle('Position',[xmin3,ymin3,xmax3-xmin3,ymax3-ymin3]);
text(xmin3,ymax3+10,'Plot 3');



textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
fignum = fignum+1;
```

GenerateGraphs.m

```
load('ResultDatafromChirpAnalysis');
foldername = 'D:\Thesis\Pictures\ChirpTest\';
fignum = 1;

f = 0:1000/9499:1000;
fignum = figure(fignum);



plot(f,meandata2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal Plot
1');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
xmin1 = 270; xmax1 = 295; ymin1 = 100; ymax1 = 700;
axis([xmin1 xmax1 ymin1 ymax1]);

textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
```

```
saveas(fignum,textfilename);
fignum = fignum+1;

fignum = figure(fignum);
plot(f,meandata2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal Plot
2');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
xmin2 = 290; xmax2 = 350; ymin2 = 100; ymax2 = 750;
axis([xmin2 xmax2 ymin2 ymax2])


textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
fignum = fignum+1;

fignum = figure(fignum);
plot(f,meandata2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal Plot
3');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
xmin3 = 150; xmax3 = 240; ymin3 = 100; ymax3 = 350;
axis([xmin3 xmax3 ymin3 ymax3])


textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
fignum = fignum+1;

fignum = figure(fignum);
plot(f,meandata2);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Stethoscope Appartus Frequency Response from Chirp Signal
Overlapped');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
%   Draw Rectangle
rectangle('Position',[xmin1,ymin1,xmax1-xmin1,ymax1-ymin1]);
text(xmin1,ymax1+10,'Plot 1');

rectangle('Position',[xmin2,ymin2,xmax2-xmin2,ymax2-ymin2]);
text(xmin2,ymax2+10,'Plot 2');

rectangle('Position',[xmin3,ymin3,xmax3-xmin3,ymax3-ymin3]);
text(xmin3,ymax3+10,'Plot 3');
```

```
textfilename=
[foldername,'ChirpTestFreqResponse',num2str(fignum),'.jpg'];
saveas(fignum,textfilename);
fignum = fignum+1;
```

normailizedfrequencyresponseChirpData.m

```
load('smoothil2');
f = 0:1000/9499:1000;

%%  Normailized frequency response between 20 to 200;
%   The suggestion in the comment below is to show the frequency range
that
%   you had previously said was where most heart sound information
occurred
%   (20-200 Hz was what I recalled), and to display the difference
sensor
%   responses in a way that made clearer any differences in magnitude
%   response of the different channels without the added confusion of
%   differences in the overall magnitude level of each due to
sensitivity
%   or directivity effects.  To do this, you want to remove any overall
%   gain difference between each channel.  My suggestion was to
normalize
%   the magnitude response plots so that the power in each signal for
the
%   frequencies of interest was the same.  For this, you would sum up
the
%   magnitude squared values of each sensor signal's FFT for those FFT
%   samples that were in the range of 20-200 Hz (or whatever range was
%   appropriate), and divide the fft values by the square root of that
sum.
%   This would equalize the power in the signals in the frequencies of
%   interest.

data = SmoothHIL2(190:1899,:);
fval = f(190:1899)';

sqrtsumdata = repmat(sqrt(sum(data)),1710,1);
data1 = data./sqrtsumdata;
figure(2);
plot(data1);



%%
% Then choose a reference level, perhaps the peak value of the
equalized
% signal spectrum from channel 3 to divide all the magnitude values for
all
```

```
% the equalized signals (so that their magnitudes are referenced to
this
% level as a value of "1.0" or 0 db); and then take 20 log10 of these
% magnitude values to convert to dBs.  Then plot these all on one set
of
% frequency axes.

maxvalue = max(max(data1));
data2 = data1/maxvalue;
plot(data2);
data3 = 20 * log(data2);

figure(3);
plot(fval,data3);
axis([20 200 -120 0]);
legend('Sensor 1','Sensor 2','Sensor 3','Sensor 4','Sensor
5','Location','SouthEast');
xlabel('Frequency (Hz)');
ylabel('Sensitivity (dB)');
title('Sensitivity vs Frequency');

%%   Normalize around sensor 3
% Then  in a table, record the difference in sensitivity that you
equalized
% out for each, by converting the magnitude squared sum to dBs and
% normalize to the channel 3 value (take 10 log10 of the magnitude
squared
% sum for each channel to get dBs, and then subtract off the dB value
from
% Channel 3 so that it has 0 dB sensitivity and the others probably
have
% negative dB values relative to channel 3).

repsensor3 = repmat(data3(:,3),1,5);
data4 = data3-repsensor3;
figure(4)
plot(fval,data4);
axis([20 200 -10 20]);
legend('Sensor 1','Sensor 2','Sensor 3','Sensor 4','Sensor 5');
xlabel('Frequency (Hz)');
ylabel('Difference in Sensitivity from Sensor 3 (dB)');
title('Difference in Sensitivity to Sensor 3');
```

Appendix M MATLAB Characterization Collect Data White Noise Script

GenerateWhiteNoise

```
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');
%   Create white noise
whitenoise1 = wgn(1000000,1,1);
[Y,Yss,f] = fftfunc(whitenoise1,44100);

plotfreq(f, Yss, 40);
title('White Noise Spectrum');
n=get(gca,'Ytick');
set(gca,'yticklabel',sprintf('%1.0i |',n'));

n=get(gca,'Xtick');
set(gca,'xticklabel',sprintf('%1.1i |',n'));

ymin = 0;
ymax = max(Yss);
axis([0,22050, ymin,ymax])
```

Appendix N MATLAB Characterization Analysis White Noise Script

DisplayDataAirTest2_27_2014.m

```
addpath('D:\Thesis\CoolTerm\CoolTermWin');
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');

MagnitudeRange;
Fs = 2000;
figure_num = 1;

xmin = 0;
xmax = 9000;
[Data1, FFTData(:,:,1), freq(:,1), figure_num] =
DisplayDatafunc('Airtest2_27_2014_test1_whitenoise_fix.txt',Fs, figure_num);
[Data2, FFTData(:,:,2), freq(:,2), figure_num] =
DisplayDatafunc('Airtest2_27_2014_test2_whitenoise_fix.txt',Fs, figure_num);
[Data3, FFTData(:,:,3), freq(:,3), figure_num] =
DisplayDatafunc('Airtest2_27_2014_test3_whitenoise_fix.txt',Fs, figure_num);
[Data4, FFTData(:,:,4), freq(:,4), figure_num] =
DisplayDatafunc('Airtest2_27_2014_test4_whitenoise_fix.txt',Fs, figure_num);

onesdata = ones(size(FFTData(:,:,1)));
f = freq(:,3);
%%  Get rid of DC
for i = 1: 4

   WN_datawork(:,:,i) = FFTData(:,:,i);
   WN_data_mean1 = mean(WN_datawork(:,:,i),1);
   WN_data_mean = onesdata(:,1)*WN_data_mean1;
   WN_norm_data(:,:,i) = abs(WN_datawork(:,:,i)-WN_data_mean);
end

WN_meandata = mean(WN_norm_data,3);

figure_num = plot5freq(f,WN_meandata,figure_num);
axis5(WN_meandata,100,900);
suptitle('Original Frequency Response from the White Noise Test');
saveas(figure_num-1,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\Pictures\AirTest3_30_2014\whitenoise_freqresp.jpg');

movavefilter = [1/200;repmat(1/100,99,1);1/200];

for i = 1:5
WN_meandata2(:,i) = conv(WN_meandata(:,i),movavefilter,'same');

end

figure_num = plot5freq(f,WN_meandata2,figure_num);
axis5(WN_meandata2,100,900);
suptitle('Frequency Response from the White Noise Test');

saveas(figure_num-1,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\Pictures\AirTest3_30_2014\whitenoise_freqresp_movave.jpg');
```

```
figure(figure_num);
figure_num = figure_num + 1;
plot(f,WN_meandata2);
title('Frequency Response from the White Noise Test Overlapped');
xlabel('Frequency (Hz)');
ylabel('|Y(f)|');
legend('Sensor 1', 'Sensor 2', 'Sensor 3', 'Sensor 4', 'Sensor 5');
axis([100 900 .005 .03]);

saveas(figure_num-1,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\Pictures\AirTest3_30_2014\whitenoise_freqresp_movave_overlapped.jpg');
```

Appendix O MATLAB Script used for determine the location of the stethoscope

Determineposition2_8_2014.m

```
%   Ultrasound Beamforming Thesis Project
%   Spencer Wong
%   Winter 2014
%
%   Determine position of stethoscope using scanner.
%

%   Get Image
object2 = imread('position2_8_2014_0002.jpg');

object2 = imcrop(object2,[100 150 2500 1800]);
% figure(2);      %   Original image cropped
% imshow(object2);
H = figure(10);      %   Image with the analysis on it
s = size(object2);
blankimage = zeros(s)+255;
imshow(object2);

%   Find Circle in picture
[centers, radii, metric] = imfindcircles(object2,[155 170],'ObjectPolarity', 'dark', 'Sensitivity',.99);

centerobject = 3;
%   use the center as the reference
position_in(1,:) = centers(1,:) - centers(centerobject,:);
position_in(2,:) = centers(2,:) - centers(centerobject,:);
position_in(3,:) = centers(3,:) - centers(centerobject,:);
position_in(4,:) = centers(4,:) - centers(centerobject,:);
position_in(5,:) = centers(5,:) - centers(centerobject,:);

%   Convert from pixels to inches
position_in = position_in/300;

%   Draw line
blankimage(:,centers(centerobject,1)-5:centers(centerobject,1)+5) = 0;
blankimage(centers(centerobject,2)-5:centers(centerobject,2)+5,:) = 0;
object2(:,centers(centerobject,1)-5:centers(centerobject,1)+5) = 0;
object2(centers(centerobject,2)-5:centers(centerobject,2)+5,:) = 0;

%   Add circle to the image
radius = 10;
viscircles(centers(1,:), radius,'EdgeColor','b');
viscircles(centers(2,:), radius,'EdgeColor','b');
viscircles(centers(3,:), radius,'EdgeColor','b');
viscircles(centers(4,:), radius,'EdgeColor','b');
viscircles(centers(5,:), radius,'EdgeColor','b');

%   Identify the sensor number
sensordisplacex = -20;
sensordisplacey = 50;
fontsize = 15;
```

```
text(centers(1,1)-sensordisplacex,centers(1,2)-sensordisplacey,'Sensor 1','FontSize',fontsize);
text(centers(2,1)-sensordisplacex,centers(2,2)-sensordisplacey,'Sensor 2','FontSize',fontsize);
text(centers(3,1)-sensordisplacex,centers(3,2)-sensordisplacey,'Sensor 3','FontSize',fontsize);
text(centers(4,1)-sensordisplacex,centers(4,2)-sensordisplacey,'Sensor 4','FontSize',fontsize);
text(centers(5,1)-sensordisplacex,centers(5,2)-sensordisplacey,'Sensor 5','FontSize',fontsize);

%   Label position
displacevaluey = -50;
displacevaluex = -20;
text(centers(1,1)-displacevaluex,centers(1,2)-
displacevaluey,mat2str(position_in(1,:),3),'FontSize',fontsize);
text(centers(2,1)-displacevaluex,centers(2,2)-
displacevaluey,mat2str(position_in(2,:),3),'FontSize',fontsize);
text(centers(3,1)-displacevaluex,centers(3,2)-
displacevaluey,mat2str(position_in(3,:),3),'FontSize',fontsize);
text(centers(4,1)-displacevaluex,centers(4,2)-
displacevaluey,mat2str(position_in(4,:),3),'FontSize',fontsize);
text(centers(5,1)-displacevaluex,centers(5,2)-
displacevaluey,mat2str(position_in(5,:),3),'FontSize',fontsize);

text(centers(centerobject,1) + 10,30,'y axis','FontSize',fontsize);
text(s(2)-250 ,centers(centerobject,2)-40,'x axis','FontSize',fontsize);
title('Stethoscopes Position and Numbered');
```

Appendix P Mechanical Frame Dimensions

Quantity	Name	Dimension
4	Wood Blocks	¾ x ¾ x 6 ¾

Appendix Q MATLAB Code for collecting Body Sounds

CollectData.m

```
%   Collect Data from the body
%   This program will ask to collect data then display the data
collected.
%   Then it will ask if you would like to save it.
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');
%figurenum = 1;
numsamples = 9500;
numcolumns = 6;     %   index and 5 sensors
data = zeros(numsamples,numcolumns);

arduinoDue = serial('COM11','BaudRate',57600,'DataBits',8
,'Parity','none');
fopen(arduinoDue);
pause(2)    %  give the controller time to boot up.
display('Start Collecting in 4 seconds');
for i = 1:4
    pause(1)
    display(num2str(4-i));
end
display('Arduino Due is Sampling');
fprintf(arduinoDue,'1');
%   Gather Data
pause(5)
    display('Arduino Due is Done Sampling');
    display('Arduino Due is transmitting Data');
for i = 1:numsamples
    inputData = strsplit(fscanf(arduinoDue))
    cellsize = size(inputData);

    for j = 1:cellsize(2)-1
        data(i,j) = str2double(inputData{j});
    end
end
display('Done Getting Data');

data1(:,1:5) = normalize_average5(data(:,2:6));
index = data(:,1);
figurenum = plot5time(index, data1,figurenum);
axis5(data1,0,9500)
%   Save Data
str = input('Save Data? (y/n)\n','s');
if str == 'y'
    nameoffile = input('Name of Data File\n','s');
    nameoffile_txt = [nameoffile, '.txt'];
    fileID = fopen(nameoffile_txt,'w');
    fprintf(fileID, '%f\t%f\t%f\t%f\t%f\t%f\n', data');
    fclose(fileID);
else
end
fclose(arduinoDue);
```

Appendix R MATLAB Code Analysis Code for Beamforming Methods 1 to 6

Optimization_Script_3.m

```
       addpath('C:\Users\Spencer Wong\Documents\My Dropbox\Thesis\MATLAB
Simulation\thesisfunction');
      addpath('D:\Thesis\MATLAB Simulation\thesisfunction');
      %%  Optimization
      %   Cal Poly 2014 EE Thesis Project
      %   Optimization Test
      %   5/22/2014
      %   Spencer Wong

      %%  Initialize
      clear;
      close all;

      % Outputs Data into variable Data
      load('FixeData8_24_2014.mat')
      %%  Initialize the section of the sound
      FixeData8_24_2014_section;
      figurenum=1;

      %%  Time
      time = Data(:,1)*1/2000;

      %%  Figure Dimensions
      figdim = [10, 50, 1350,600];

      %%   Output the data
      plot5time(time, Data(:,2:6),figurenum);    %   Figure 1
      axis5(Data(:,2:6),1,3.75);
      suptitle('Data Used for Analysis');
      figurenum=figurenum+1;

      %%  Normalized the Data
      Data2 = Data;
      Data2(:,2:6) = normalize_average5(Data(:,2:6));
      Data2(:,2:6)=Data2(:,2:6)./max(max(Data2(:,2:6)));

      %%  Plot the Normalized Data with the sections
      plot5time(time, Data2(:,2:6),figurenum);    %figure 2
      HighlightSectionsfunc5((1/2000)*HeartSoundRange,Data2(:,2:6));
      figurenum=figurenum+1;
      axis5(Data2(:,2:6),1,3)
      suptitle('Original Heart Signal Segmentation');

      %%  Mask S1 Data
      DataS1_Masked =
HeartSoundMask5(Data2(:,2:6),HeartSoundRange(2,1:2));
      DataS1 = HeartSoundMask5(Data2(:,2:6),HeartSoundRange(:,1:2));
```

```
      %%  Mask S2 Data
      DataS2_Masked =
HeartSoundMask5(Data2(:,2:6),HeartSoundRange(2,5:6));
      DataS2 = HeartSoundMask5(Data2(:,2:6),HeartSoundRange(:,5:6));

      %%  Time Delay
      [timedelayS1, numshiftS1]  =
thephasedelay5_1(DataS1_Masked,2,1,3,4,5,2000);
      [timedelayS2, numshiftS2]  =
thephasedelay5_1(DataS2_Masked,2,1,3,4,5,2000);

      %%  Beamforming Test 1 Run
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
      [DataS1_shifted] = shiftdata5(DataS1,numshiftS1);
      [DataS1_shifted_masked] = shiftdata5(DataS1_Masked,numshiftS1);
      [DataS1_shifted_2] = shiftdata5(DataS1_Masked,numshiftS1);
      DataS1_shifted_added = DataS1_shifted(:,1) + DataS1_shifted(:,2) +
DataS1_shifted(:,3) + DataS1_shifted(:,4) + DataS1_shifted(:,5);

      [DataS2_shifted] = shiftdata5(DataS2,numshiftS2);
      [DataS2_shifted_masked] = shiftdata5(DataS2_Masked,numshiftS2);
      [DataS2_shifted_2] = shiftdata5(DataS2_Masked,numshiftS2);

      DataS2_shifted_added = DataS2_shifted(:,1) + DataS2_shifted(:,2) +
DataS2_shifted(:,3) + DataS2_shifted(:,4) + DataS2_shifted(:,5);

      %%   Combine S1 and S2 signal
      Data_combined = DataS2_shifted_added+DataS1_shifted_added;

      %   Taper edges
      Hd = FIR_LOWPASS_FILTER3;
      Data_combined2Test1= filter(Hd.numerator,1,Data_combined);

      %%  Display the Data
      figure(figurenum);      %   Figure 3
      figurenum = figurenum + 1;
      plot(time, Data2(:,3),time, Data_combined2Test1)
      axis([1.4 2 -3.5 2.5]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      legend('Sensor 2', 'Method 1');
      title('Method 1 Result');

      %   S1 Metrics
      CorrCo_S1_prealign = corrcoef(DataS1_Masked);
      CorrCo_S1 = corrcoef(DataS1_shifted);
      CorrCo_S1_masked = corrcoef(DataS1_shifted_masked);
      CorrCo_S1size = size(CorrCo_S1);

      %   S2 Metrics
```

```
      CorrCo_S2_prealign = corrcoef(DataS2_Masked);
      CorrCo_S2 = corrcoef(DataS2_shifted);
      CorrCo_S2_masked = corrcoef(DataS2_shifted_masked);



      %-------------------------------------------------------------------
--%
      %  Beamforming Test 2 Run
      %-------------------------------------------------------------------
--%
      Test2Mask;
      %%  Plot the Normalized Data with the sections
      plot5time(time, Data2(:,2:6),figurenum);    %   Figure 4
      HighlightSectionsfunc5_1((1/2000) * Test2MaskRange,Data2(:,2:6));
      figurenum=figurenum+1;
      axis5(Data2(:,2:6),1,3.75)
      suptitle('Section of the Data Used for Method 2');

      %%  Mask S1 Data
      DataS1_A2 = HeartSoundMask5(Data2(:,2:6),Test2MaskRange(:,1:2));

      %%  Mask S2 Data
      DataS2_A2 = HeartSoundMask5(Data2(:,2:6),Test2MaskRange(:,3:4));

      [DataS1_A2_shifted] = shiftdata5(DataS1_A2,numshiftS1);
      DataS1_A2_shifted_added = DataS1_A2_shifted(:,1) +
DataS1_A2_shifted(:,2) + DataS1_A2_shifted(:,3) + DataS1_A2_shifted(:,4)
+ DataS1_A2_shifted(:,5);

      [DataS2_A2_shifted] = shiftdata5(DataS2_A2,numshiftS2);
      DataS2_A2_shifted_added = DataS2_A2_shifted(:,1) +
DataS2_A2_shifted(:,2) + DataS2_A2_shifted(:,3) + DataS2_A2_shifted(:,4)
+ DataS2_A2_shifted(:,5);

      Method2combined = DataS2_A2_shifted_added +
DataS1_A2_shifted_added;
      %   Taper edges
      Hd = FIR_LOWPASS_FILTER3;
      Method2combined = filter(Hd.numerator,1,Method2combined);
      % Method2combined = Method2combined/max(abs(Method2combined));

      %%  Display the Data
      figure(figurenum);      %figure 5
      figurenum = figurenum + 1;
      plot(time, Data2(:,3), time, Method2combined)
      axis([1.4 2 -4 3]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      suptitle('Method 2 Result');
      legend('Sensor 2 Data', 'Method 2');
```

```
      % %%  Beamforming Test 3 Run
      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      [Method3S1data] = shiftdata5(Data2(:,2:6),numshiftS1);
      Method3S1dataResult = Method3S1data(:,1) + Method3S1data(:,2) +
Method3S1data(:,3) + Method3S1data(:,4) + Method3S1data(:,5);
      % Method3S1dataResult =
Method3S1dataResult/(max(Method3S1dataResult));
      Method3S1dataResult= filter(Hd.numerator,1,Method3S1dataResult);
      %%  Display the Data
      figure(figurenum);      %   Figure 6
      figurenum = figurenum + 1;
      plot(time, Data2(:,3),time, Method3S1dataResult)
      axis([1.4 2 -4 3]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      suptitle('Method 3 S1 Result');
      legend('Sensor 2 Data', 'Method 3 S1');



      %%  Beamforming Method 3 S2Run
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
      [Method3S2data] = shiftdata5(Data2(:,2:6),numshiftS2);
      Method3S2dataResult = Method3S2data(:,1) + Method3S2data(:,2) +
Method3S2data(:,3) + Method3S2data(:,4) + Method3S2data(:,5);
      Method3S2dataResulto = Method3S2dataResult ;
      % Method3S2dataResult =
Method3S2dataResult/max(Method3S2dataResult);

      Method3S2dataResult= filter(Hd.numerator,1,Method3S2dataResult);
      %%  Display the Data
      figure(figurenum);      %   figure 7
      figurenum = figurenum + 1;
      plot(time, Data2(:,3),time, Method3S2dataResult)
      axis([1.4 2 -4 3]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      suptitle('Method 3 S2 Result');
      legend('Sensor 2 Data', 'Method 3 S2');



      % %%  Beamforming Method 4 Run
      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      [Method4S1data] = shiftdata5(DataS1,numshiftS1);
      Method4S1dataResult = Method4S1data(:,1) + Method4S1data(:,2) +
Method4S1data(:,3) + Method4S1data(:,4) + Method4S1data(:,5);
      % Method4S1dataResult =
Method4S1dataResult/max(Method4S1dataResult);
```

```
    Method4S1dataResult= filter(Hd.numerator,1,Method4S1dataResult);
    %%  Display the Data
    figure(figurenum);      %      Figure 8
    figurenum = figurenum + 1;
    plot(time, Method4S1dataResult)
    axis([1.4 2 -4 3]);
    xlabel('Time (s)')
    ylabel('Amplitude')
    suptitle('Method 4 S1 Result');
    legend( 'Method 4 S1');


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
    %%  Beamforming Test 4 S2 Run
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
    [Method4S2data] = shiftdata5(DataS2,numshiftS2);
    Method4S2dataResult = Method4S2data(:,1) + Method4S2data(:,2) +
Method4S2data(:,3) + Method4S2data(:,4) + Method4S2data(:,5);
    % Method4S2dataResult =
Method4S2dataResult/max(Method4S2dataResult);

    Method4S2dataResult= filter(Hd.numerator,1,Method4S2dataResult);
    %%  Display the Data
    figure(figurenum);      %   figure 9
    figurenum = figurenum + 1;
    plot(time, Method4S2dataResult)
    axis([1.4 2 -4 3])
    xlabel('Time (s)')
    ylabel('Amplitude')
    suptitle('Method 4 S2 Result');
    legend('Method 4 S2');



    %%  Compare Method results 3 and 4
    figure(figurenum);      %   figure 10
    figurenum = figurenum + 1;
    plot(time, Method3S1dataResult,time, Method3S2dataResult)
    axis([1.4 2 -4 3]);
    xlabel('Time (s)');
    ylabel('Amplitude');
    suptitle({'Comparison Between'; 'Method 3 and Method 4'});
    legend('Method 3', 'Method 4')

    %%  Sensor 3 Data
    figure(figurenum);      %   figure 11
    figurenum = figurenum + 1;
    plot(time, Data2(:,4));
    axis([1 3 -4 3])
```

```
xlabel('Time (s)');
ylabel('Amplitude');
suptitle('Sensor 3 Data');
legend('Sensor 3')

%%  Sensor 2 Data
figure(figurenum);      %   figure 12
figurenum = figurenum + 1;
plot(time, Data2(:,3));
axis([1 3 -4 3])
xlabel('Time (s)');
ylabel('Amplitude');
suptitle('Sensor 2 Data');
legend('Sensor 2')



%%  Masked S1 Plot
figure(figurenum);
figurenum = figurenum + 1;
subplot(2,1,1)
plot(time, DataS1_Masked);
axis([1.45 1.7 -1.5 1.5])
xlabel('Time (s)');
ylabel({'S1 Data Masked';'Amplitude'});

subplot(2,1,2)
plot(time, DataS2_Masked);
axis([1.78 1.92 -1.5 1.5])
xlabel('Time (s)');
ylabel({'S2 Data Masked';'Amplitude'});
suptitle('Masked Data');


%%  S1 Data compared shifted
figure(figurenum);
figurenum = figurenum + 1;
subplot(2,1,1)
plot(time, Data2(:,2:6));
axis([1.45 1.70 -2 2]);
xlabel('Time (s)')
ylabel('Amplitude')
title('Original S1 Data');
subplot(2,1,2);
plot(time, Method3S1data);
axis([1.45 1.70 -2 2]);
xlabel('Time (s)')
ylabel('Amplitude')
title('S1 Data Shifted');
suptitle('S1 Comparison Between Original and Shifted');
```

```
      %%  S2 Data compared shifted
      figure(figurenum);
      figurenum = figurenum + 1;
      subplot(2,1,1)
      plot(time, Data2(:,2:6));
      axis([1.75 1.95 -1.5 1.5]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      title('Original S2 Data');
      subplot(2,1,2);
      plot(time, Method3S2data);
      axis([1.75 1.95 -1.5 1.5]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      title('S2 Data Shifted');
      suptitle('S2 Comparison Between Original and Shifted');

      figure(figurenum);
      figurenum = figurenum + 1;
      plot(time, Method3S1dataResult);
      axis([1.75 1.95 -4 4]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      suptitle('S1 Data Summed');

      figure(figurenum);
      figurenum = figurenum + 1;
      plot(time, Method3S2dataResult);
      axis([1.75 1.95 -4 4]);
      xlabel('Time (s)')
      ylabel('Amplitude')
      suptitle('S2 Data Summed');


      plot5time(time, Data2(:,2:6),figurenum);
      suptitle('Data used for Analysis');
      figurenum = figurenum + 1;

      %  Save the diagrams
      for i = 1:figurenum-1
         set(i,'Position',figdim);
          imagename = ['C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\figures\fig', num2str(i),'.jpg'];
          saveas(i,imagename);
      end


      %   SNR analysis on methods
      VSNRResult2(1,:) = VSNR(2, HeartSoundRange, Data2(:,3));
      VSNRResult2(2,:) = VSNR(2, HeartSoundRange, Method2combined);
      VSNR2compare = 20*log(VSNRResult2);
```

```
VSNRResult3(1,:) = VSNR(2, HeartSoundRange, Data2(:,3));
VSNRResult3(2,:) = VSNR(2, HeartSoundRange, Method3S1dataResult);
VSNRResult3(3,:) = VSNR(2, HeartSoundRange, Method3S2dataResulto);
VSNR3compare = 20*log(VSNRResult3);
```

Appendix S MATLAB Code for Additional Analysis

testTwoChannelSound5_29_2014.m

```
addpath('D:\Thesis\MATLAB Simulation\CollectData');
addpath('D:\Thesis\MATLAB Simulation\thesisfunction');
folderlocation = 'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\GenerateSoundSignal5_29_2014\';
fs= 2000;

load('FixeDATA5_14_2014.mat')
index = Data(:,1);
time = index *(1/2000);
Data2(:,2:6) = normalize_average5(Data(:,2:6));
Data2(:,2:6)=Data2(:,2:6)./max(max(Data2(:,2:6)));

dataout = Data2(:,2:6);
%[index, time,  dataout] = GetThedata('test3_5_2014_1.txt',fs);
ndata = zeros(8501,5);
for l = 1:5
    ndata = dataout./(max(max(abs(dataout))));
end
for i = 1:5
    for j = 1:5
            LeftPhone = ndata(:,i);
            RightPhone = ndata(:,j);
            %   Give File_Name
            File_Name =
[folderlocation,'WavFile_L',num2str(i),'_R',num2str(j)];

            %   Create Wave File
            wavwrite([LeftPhone,RightPhone],2000,16, File_Name);

            %   Create Plot of the signal generated
            fignum = figure(1);
            plot(time,LeftPhone,time,RightPhone);
            xlabel('Time (s)');
            ylabel('Amplitude');
            TitleOfDiagram = ['Sound Signal ', num2str(i),num2str(j)];
            suptitle(TitleOfDiagram);
            leftLegend = ['Left Channel: Sensor ', num2str(i)];
            rightLegend = ['Right Channel: Sensor ', num2str(j)];
            legend(leftLegend, rightLegend);

            image_File_Name_fig = [File_Name,'.fig'];
            saveas(fignum,image_File_Name_fig);
            axis([1.4 3 -1 1]);
            image_File_Name_jpg = [File_Name,'.jpg'];
            saveas(fignum,image_File_Name_jpg);
    end
end
```

CreateSound_For_SoundAnalysis.m

```
%%  Create Sounds
Data2(:,2) = Data2(:,2)/max(abs(Data2(:,2)));
Data2(:,3) = Data2(:,3)/max(abs(Data2(:,3)));
Data2(:,4) = Data2(:,4)/max(abs(Data2(:,4)));
Data2(:,5) = Data2(:,5)/max(abs(Data2(:,5)));
Data2(:,6) = Data2(:,6)/max(abs(Data2(:,6)));



%   Normalize Sound
Method1Results = Data_combined2Test1/max(abs(Data_combined2Test1));
Method2Results = Method2combined/max(abs(Method2combined));

Method3S1dataResult =
Method3S1dataResult/max(abs(Method3S1dataResult));
Method3S2dataResult =
Method3S2dataResult/max(abs(Method3S2dataResult));

Method4S1dataResult =
Method4S1dataResult/max(abs(Method4S1dataResult));
Method4S2dataResult =
Method4S2dataResult/max(abs(Method4S2dataResult));

%   Write Wave File
wavwrite(Data2(:,2),2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Original1');
wavwrite(Data2(:,3),2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Original2');
wavwrite(Data2(:,4),2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Original3');
wavwrite(Data2(:,5),2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Original4');
wavwrite(Data2(:,6),2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Original5');

wavwrite(Method1Results,2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Method1Sound');
wavwrite(Method2Results,2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Method2Sound');

wavwrite(Method3S1dataResult,2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Method3SoundS1');
wavwrite(Method3S2dataResult,2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
```

```
Simulation\DiagramsforResultsforReport\SoundAnalysis\Method3SoundS2');

wavwrite(Method4S1dataResult,2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Method4SoundS1');
wavwrite(Method4S2dataResult,2000,'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\SoundAnalysis\Method4SoundS2');
```

Appendix T Stereophonic Presentation Analysis

testTwoChannelSound8_24_2014.m

```
      addpath('D:\Thesis\MATLAB Simulation\CollectData');
      addpath('D:\Thesis\MATLAB Simulation\thesisfunction');
      folderlocation = 'C:\Users\Spencer Wong\Documents\My
Dropbox\Thesis\MATLAB
Simulation\DiagramsforResultsforReport\GenerateSoundSignal5_29_2014\';
      fs= 2000;

      load('FixeData8_24_2014.mat')
      index = Data(:,1);
      time = index *(1/2000);
      Data2(:,2:6) = normalize_average5(Data(:,2:6));
      Data2(:,2:6)=Data2(:,2:6)./max(max(Data2(:,2:6)));

      dataout = Data2(:,2:6);
      %[index, time,  dataout] = GetThedata('test3_5_2014_1.txt',fs);

      ndata = zeros(8501,5);
      for l = 1:5
          ndata = dataout./(max(max(abs(dataout))));
      end

      for i = 1:5
          for j = 1:5
                  LeftPhone = ndata(:,i);
                  RightPhone = ndata(:,j);

                  %   Give File_Name
                  File_Name =
[folderlocation,'WavFile_L',num2str(i),'_R',num2str(j)];

                  %   Create Wave File
                  wavwrite([LeftPhone,RightPhone],2000,16, File_Name);

                  %   Create Plot of the signal generated
                  fignum = figure(1);
                  plot(time,LeftPhone,time,RightPhone);
                  xlabel('Time (s)');
                  ylabel('Amplitude');
                  TitleOfDiagram = ['Sound Signal ',
num2str(i),num2str(j)];
                  suptitle(TitleOfDiagram);
                  leftLegend = ['Left Channel: Sensor ', num2str(i)];
                  rightLegend = ['Right Channel: Sensor ', num2str(j)];
                  legend(leftLegend, rightLegend);

                  image_File_Name_fig = [File_Name,'.fig'];
                  saveas(fignum,image_File_Name_fig);
                  axis([1.4 3 -1 1]);
                  image_File_Name_jpg = [File_Name,'.jpg'];
```

```
                    saveas(fignum,image_File_Name_jpg);
            end
        end
```