Desktop-Sized Magnetic Resonance Imaging System

Slavko N. Rebec
Adviser: Dr. Mark Griswold

Department of Physics
Case Western Reserve University
Cleveland, OH 44106-7079

May 3, 2014

ABSTRACT

The goal of this project was to develop an inexpensive, desktop-sized magnetic resonance imaging (MRI) system. In order to meet this goal, we built a series of hardware components necessary for MRI machines and tested the performance of each to ensure that they met the standards required to accurately execute a wide variety of MRI experiments. A microcontroller was programmed to control the timing and output of the hardware using simple experimental details provided by a user through a GUI. This device could be used in classrooms to provide students with hands-on MRI experience.

ACKNOWLEDGMENTS

I would first and foremost like to thank my advisor, Dr. Mark Griswold, for his guidance, support and willingness to answer my many questions throughout this project. I would also like to thank his graduate student, Michael Twieg, for teaching me more about circuitry and for his assistance in the design, building and testing of the hardware components. Finally, I would like to thank my family and friends for supporting me throughout not only this project, but also this busy senior year. Without all of these people, this work would have not been possible.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

Due to its clinical utility, magnetic resonance imaging (MRI) is a growing field and there are many graduate and undergraduate programs which offer courses about MRI. Due to the high hourly operational cost and the risk of damage to the MRI machine, hands-on experience in these MRI courses is impractical. The goal of this project is to produce a desktop sized MRI for classroom applications which is inexpensive and can be easily interfaced with through MATLAB. The proposed device could be used in laboratory and imaging courses at Case Western Reserve University, including but not limited to PHYS 317: Engineering Physics Laboratory "Junior Lab", PHYS/EBME 431: Physics of Imaging, and EBME 320/410: Medical Imaging Fundamentals. In addition to making the device, an extensive set of documentation will be created to allow other groups at different universities to follow the same steps and produce their own desktop MRI machines.

1.2 Background

Magnetic resonance imaging is a relatively new field with a lot of active research due to its useful clinical applications. MRI allows clinicians to noninvasively image inside of the body down to a nanometer scale and to generate contrast between a variety of soft tissues without the harmful radiation common to other medical imaging modalities [1]. To provide interested students/ future researchers an opportunity to get hands-on experience with MRI machines in their classrooms, we are proposing to design a portable, inexpensive MRI

machine. But before going further into my motivations and methods, it is necessary to give a brief background on MRI.

Before the advent of MRI, there was nuclear magnetic resonance (NMR). NMR was primarily concerned with determining properties of homogeneous chemical samples. The samples are composed of atoms, which all have an associated magnetic moment. Commonly samples with high proton concentrations are examined. When a sample is placed in a large magnetic field, the magnetic moments of the protons, and other particles, will preferentially align with the field. The magnetic moments are then perturbed by a RF field, commonly called an RF pulse, which is aligned perpendicularly to the main large field. A common NMR experiment is known as the spin echo and is visually depicted in Figure 1.2(A). In a spin echo experiment, spins are tipped by the pulse which causes them to de-phase while also precessing at a characteristic frequency, referred to as the Larmor frequency. The spins are then subjected to a secondary RF pulse which causes the spins to re-phase and produce a measureable signal at a time which is characteristic of the sample [1].

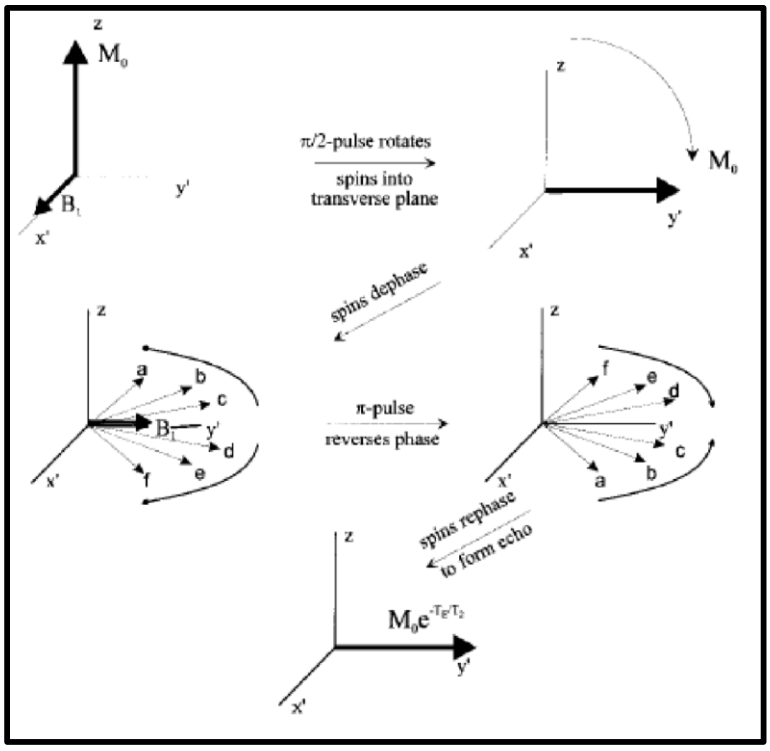

Figure 1.2(A): Magnetization Vector During Spin Echo Experiment ('Magnetic Resonance Imaging', M. Haacke, R. Brown, M. Thompson, R. Venkatesan, pg. 185).

In 1973, Paul Lauterbur, in a now famous experiment, added a set of linearly varying magnetic fields on top of

his NMR experiment and was able to make a 2D image of two cylinders filled with water. These linearly varying fields are now known as gradient fields and are generated by adding a set of coils inside of the main magnet. MRI utilizes these gradient field coils to add an additional magnetic field on top of the main magnet field to generate a net magnetic field dependent on spatial location inside of the magnet. This varying field causes spins to precess at different characteristic frequencies which allows us to localize where each signal originates in the sample. By collecting the signal output from a set of locations, we can generate an image of the sample. However, the output signal actually belongs to frequency-space, or k-space, and a 2D Fourier Transform must be done to create an image in the spatial domain. By changing the duration, timing, and amplitude of the gradient fields we can produce 2D or 3D images with a variety of contrasts. The details of a MRI experiment are normally visually depicted and are called pulse sequence diagrams (PSD). A PSD typically shows the output and timing of the RF pulse along with each of the gradient fields. Figure 1.2(B) displays the PSD for a common 3D imaging experiment, known as the 3D gradient echo [1].

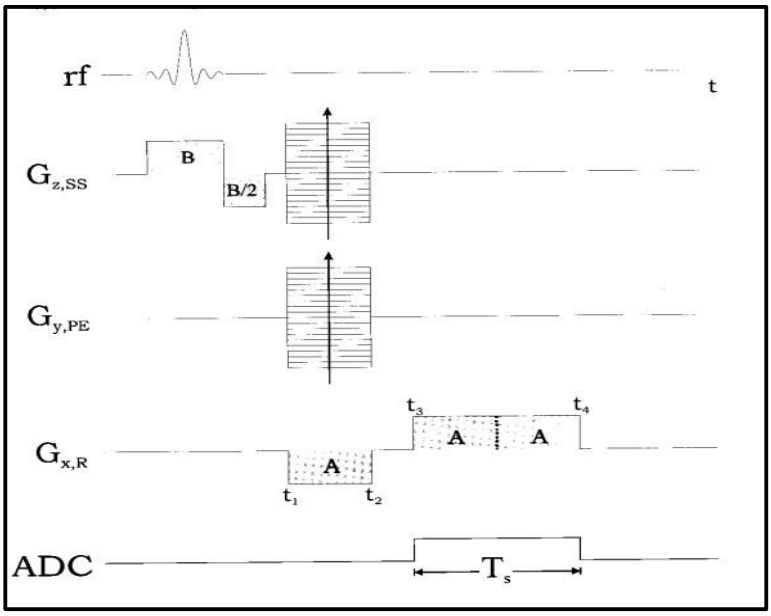

Figure 1.2(B): Example PSD ('Magnetic Resonance Imaging', M. Haacke, R. Brown, M. Thompson, R. Venkatesan, pg. 195).

Since its advent, MRI has grown into a major medical imaging modality because of its ability to generate high-resolution images with a variety of contrasts. These contrasts have been

useful in diagnosing cancer along with many other conditions such as strokes and stenoses. To improve contrast, reduce imaging times, decrease noise and limit field inhomogeneity, modern clinical MRI machines have become increasingly complicated and expensive. The primary cost is associated with generating and maintaining main magnet fields, typically with a field strength of 1.5 or 3 Tesla (T). To generate these large fields, first the main magnets are made by wrapping an average of six tons of metal wire around a bobbin-like structure which is then super-cooled down to a few kelvin to make the wire super-conducting. Once the wire is super-conducting, its electrical resistance goes to zero, allowing high currents to flow indefinitely without power loss. The cooling is done with a bath of liquid helium, which in recent years has become very expensive. Other costly components in clinical MRI systems are the variety of custom-made electronics, RF/gradient coils and computers, which are needed to process and run the experiments. These expenses lead to a purchase cost of a clinical MRI system upwards of $1,000,000 and yearly operational cost upwards of $200,000. Because of all of these costs, a single hour on a clinical MRI costs on average $1,000. [2 -3]

1.3 Review of Previous Work

While there has been a lot of successful research done on designing small MRI machines for imaging small animals, they utilize high-strength magnetic fields which are both non-portable and expensive [3-5]. Little work has been done to create small MRI machines which are inexpensive enough for classroom applications. However, Wright et al. did create a desktop-sized MRI machine in 2001, which could be used in classrooms [6]. Their design utilized an open, .21 T, permanent magnet and cost $13,500. Wright et al.'s design was

difficult to practically implement into a course budget and also utilized electronics which offered little flexibility.

1.4 Objectives

Our original goal was to design and build a simple, desktop-sized, inexpensive MRI machine. More specifically we proposed to,

1. Build a MRI machine which is portable, inexpensive, and easy to use.
2. Design a program which will allow students to easily interface with the MRI machine and carry out their own imaging experiments.
3. Write an extensive set of documentation describing every aspect of the MRI system to allow students to obtain a deeper understand the device they are using and also allow other groups to follow the same steps to make their own MRI machine.

This MRI system will allow students to apply their theoretical knowledge of MRI to design experiments and teach them how to use an actual MRI system and to see the results of their imaging experiments. This in turn will lead to a greater understanding and appreciation of MRI and hopefully continue to increase student interest in this growing field.

2 MICROCONTROLLER PROGRAMMING

2.1 Design Requirements

The microcontroller is the heart of the MRI system, and thus the majority of the time spent on this project was devoted to its programming. In general, the microcontroller must receive arbitrary experimental details in the form of a PSD matrix and convert that into a series of precisely timed controls for the gradient coils, the RF coil, the data sampling, and computer communication systems.

2.2 Microcontroller Selection

While there are many microcontroller options available, the clear choice for our project was the Arduino. First of all, to write the complicated program needed to control the MRI in the course of a semester, required a modern, high-level coding interface. This requirement severely restricted our options and effectively left us with the choice between Arduino and Raspberry Pi. Arduinos are a true microcontroller while Raspberry Pis are actually microprocessors, which functions as a simple computer running an operating system. While both systems are commonly used in electronic projects, the Raspberry Pi is better equipped to handle projects which deal with complex math, video processing, or graphical interfaces. The Raspberry Pi does not feature many I/O pins, since the majority of its control and communication is done through USB. Arduinos are more commonly used to control servos, motors, and sensors. Since we do not need to work with any video processing or complex math, and we require a number of I/O pins to control all of the different MRI components, the Arduino was the clear choice.

There were a variety of Arduino microcontrollers available for purchase. Each had its own perks and unique interface opportunities. The Arduino Due (AD) was the microcontroller that best matched our criteria. It features an 84 MHz clock, which means it should have the temporal resolution required to accurately control the MRI system. The AD features two 12-bit DACs which are required to accurately drive the RF pulse. The AD can be purchased for just $50.00.

2.3 Pulse Sequence Diagram Input

At the beginning of an experiment, the AD will read in a PSD. The PSD must contain information on the timing and output of the RF pulse, each of the gradient coils, along with the timing of data collection and communication with the computer. The PSD must be formatted according to the details of Table 2.3(A).

Column #	Corresponding Contents	Formatting
1	Time of Current command	Time in microseconds of each command
2	X-Gradient Output	Integer commands between 1 and 5, corresponding to: 1: Stay 0, 2: Stay Min Output, 3: Stay Max Output, 4: Increment from 0 to Max, 5: Increment from 0 to Min, 6: Increment from Min to Max, 7: Increment from Max to Min.
3	Y-Gradient Output	
4	Z-Gradient Output	
5	RF Output	Binary Command: 1 = On, 2 = Off
6	Data Read	Binary Command: 1 = On, 2 = Off
7	Data Send	Binary Command: 1 = On, 2 = Off
8	Running Total of Time	Current Sum of Column 1

9	All of the other experimental variables: # of increments of G_x, G_y and G_z, # of Repeat Experiments	A total of 4 integer values

Table 2.3(A): Description of the formatting and contents of the PSD matrix.

Interpreting Table 2.3(A), we can see the PSD must have 9 columns and has an arbitrary number of rows.

2.4 Program Overview

In lieu of providing the final AD program, which would be completely incomprehensible to most readers, I will provide a simplified verbal description of each critical aspect of the program. In the later sections, I will explain the specific difficulties of implementing particular controls. For instructions on accessing the program, see section 3.3 of this document.

The program begins by initializing a number of variables which are used throughout the rest of the program. The program waits for a connection to be established with Matlab before reading in the provided PSD matrix from the serial port. Once completed, the program will enter into the main program loop which will be continually cycled through until the end of the experiment. The first time the main program loop is entered, a PSD timer is started at zero. This PSD timer is updated at the beginning of each loop through the main program. By comparing the PSD timer to the times provided in the PSD matrix, the program can determine what row of PSD instructions should be given to the coil, data sampling, and data sending routines. Once the PSD timer is greater than the largest time in the PSD matrix, the PSD timer is reset and the output of the gradients will be changed according to the number of gradient increments and repeat

experiments specified in the 9th column of the PSD matrix. The PSD timer reset step will continue, until the experiment concludes. The number of PSD timer resets that indicates the end of the experiment is determined by multiplying the number of gradient increments by the number of repeat experiments as set by the 9th column of the PSD matrix.

2.5 Output Timing

While writing my AD program, I ran into many timing problems. Frequently these timing issues were easily fixable. However, one timing error, in particular, required more than one and a half months to adequately understand and address. This timing error was caused by the nature of the AD analog output pins that control the gradient coils. When writing a particular output value to one of these pins, instead of varying the voltage level, the duty cycle of the output square wave is changed. This technique, commonly used on microcontrollers, is called pulse width modulation (PWM). When writing a value of zero to a PWM pin, the square wave produced will have a 0% duty cycle, or a constant magnitude of 0 V. When writing the max value to a PWM pin, a constant, max voltage will be produced. However, when writing a value some

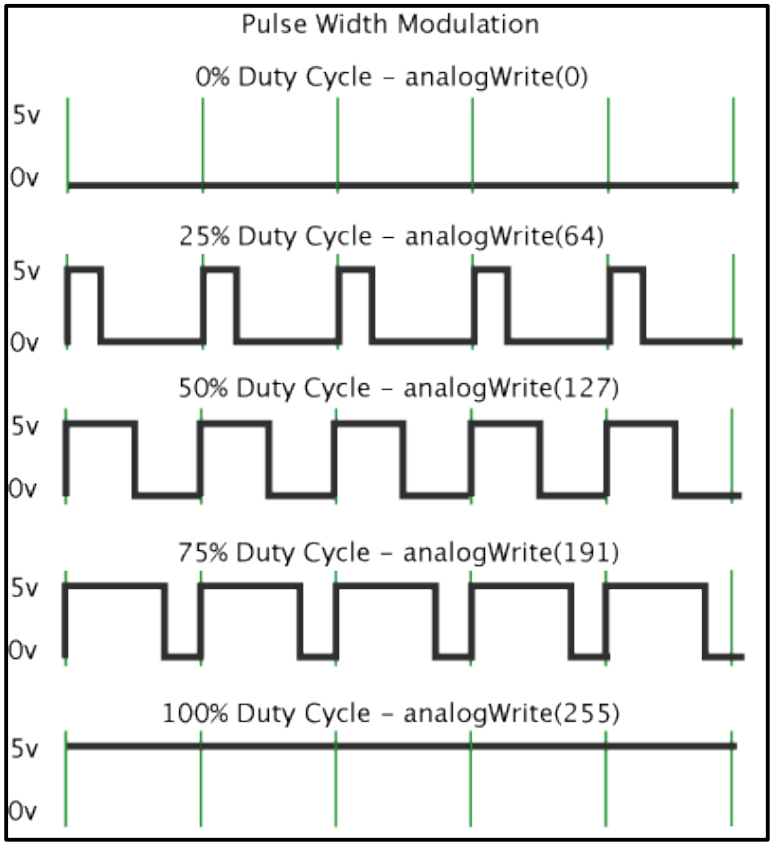

Figure 2.5(A): Simple diagram depicting possible PWM outputs.

fractional percent, p, of the max value, a square wave with a duty cycle p will be produced. A visual depiction of PWM outputs can be seen in Figure 2.5(A).

The default frequency for the AD PWM outputs is 1 kHz, which is not fast enough to accurately reproduce the required gradient outputs, which only last a couple hundred microseconds. Unfortunately, the Arduino software library offered no way to change the frequency of the PWM outputs. This lack of functionality is most likely due to the Arduino' s frequent use in low frequency applications, where 1 kHz is suitable. After reading through online forums, I found other users had run into this same issue. Some workarounds had been developed, but they always came at the cost of ruining all of the built-in Arduino timing functions, which are critical to the functionality of the AD MRI control program.

After further investigation, I found an individual who had begun to write a PWM timing library for the AD that attempted to fix the low frequency PWM problem through issuing direct commands to the Atmel SAM3X8E ARM Cortex-M3 CPU featured on the AD. After reading the datasheet on the CPU and studying other programs which also directly worked with the CPU, I was able to complete the timing library. This process involved assigning the CPU clocks to different pins and increasing the speed of the clock assigned to control the gradient coils. Through the application of this library, I successfully increased the PWM frequency to 100 kHz without compromising the AD built-in timing functions. With the gradient outputs at this higher frequency, the AD program was able to accurately produce the gradient wave forms.

With the PWM timing issues corrected, I then expanded the program to control the RF pulse. While the majority of the analog output pins on the AD function with PWM, there are two

12-bit DAC channels that produce an output of truly varying voltage. The output of the AD must be a high frequency sine wave in order to be used for the RF pulse. This output is only possible through the use of the DAC outputs. Unfortunately, the default maximum frequency output was once again too low. However, this issue had been thoroughly explored by other users and was, in fact, partly caused by an improperly referenced value in the source AD programming. With simple changes, I was able to generate a 16 kHz sine wave, which could be used to generate the RF pulse. The AD source programming has since been updated, remedying the problem.

2.6 Data Sampling and Serial Communication

With the coil controls working, data sampling functionality was added into the program. In MRI, an accurate sampling rate is required to convert the phase space data into any meaningful spatial images. The sampling time needed to be fast to maximize the experimental efficiency. A specious programing implementation would be to store the time that each sample is collected, then taking the difference between consecutive time records to find the sampling time. However, calculating and storing this data made the sampling times too high to be usable. To find the optimal programming solution for this data sampling time problem, a number of different loop architectures were tested. The ideal solution was a while loop containing an if-statement that checks the time since the previous measurement. If the time is equal to a specified sampling time, then a data point is stored. A sampling time of 40µs was selected because it was both fast and consistent. Simple testing shows that a 1 µs jitter appeared in the total sampling time after 64 consecutive measurements are made.

Due to the memory constraints of the AD, only small amounts of data can be stored at a time. This limit forces us to periodically send data back to the computer during the experiment, in order to prevent data loss or a memory overload. This cannot be done as soon as the data is read, because communicating through the serial port requires too much time to send the first bit, and it would therefore throw off the timing of the rest of the program. Instead, data was sent back during a portion of the experiment in which no other controls are necessary.

Serial communication libraries for Arduino have been well developed and are easy to use, especially with the Serial Monitor built into the Arduino IDE. However, the built-in Matlab serial libraries proved to be difficult to work with. Sample communication between the AD and Matlab would always result in timeout errors and missing data. A quick and typical work around for direct Arduino-Matlab serial communication was made using a program called Processing. Processing is a programming environment that utilizes a lot of the same serial functions that the Arduino does. Processing is commonly used to read, display and store data coming from the AD.

During a gap in the work schedule for the hardware portion of the project, I reexamined the Matlab-Arduino communication problems. I was able to resolve these issues and therefore eliminate the need for the intermediary Processing program through a better understanding of Matlab serial objects and a more sophisticated serial reading structure.

3 COMPUTER INTERFACE

3.1 Design Requirements

The goal of the computer interface is to allow students to easily interact with the microcontroller through a simple graphical interface instead of directly through the more complicated Arduino programming interface. The program must be able to read in a PSD matrix designated by the student. Then it must connect to the AD and send it the provided PSD matrix. Then it must constantly monitor for, collect, and save data coming through the serial port from the AD until the end of the experiment. The program must also be able to communicate any errors to the user and provide documentation detailing how to use the program.

3.2 Graphical User Interface

The graphical user interface (GUI) was programmed in Matlab. Matlab was selected over other popular languages such as R, Python, and Java primarily due to its prevalent usage for science and engineering undergraduates across the country, especially at Case Western Reserve University. Matlab was also selected due to the availability of many libraries focused on MRI data analysis and due to my familiarity with the language.

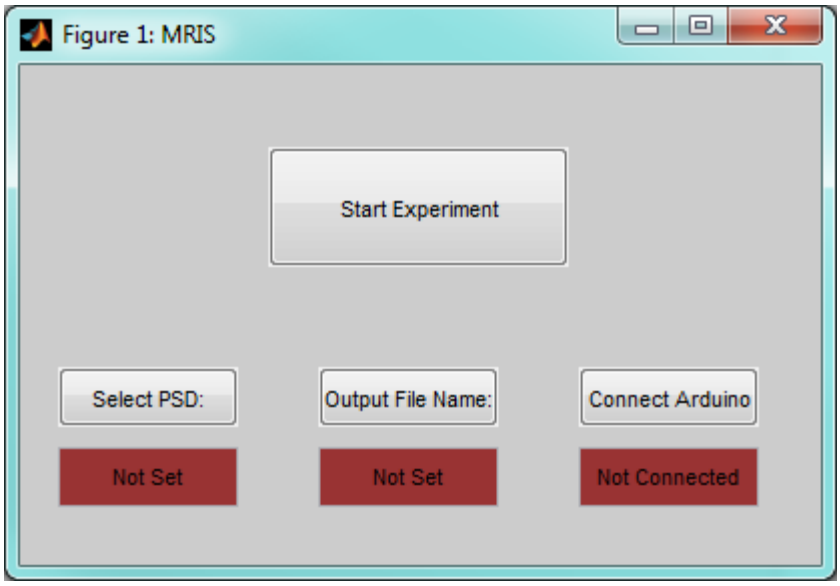

Figure 3.2(A): Starting Matlab GUI panel generated by MRIS.

The Matlab program, titled *MRIS*, is started with the Matlab command line. It begins by creating the starting GUI panel as shown in Figure 3.2(A). This panel and all of the following were made through hard programming of each button, textbox, and color as opposed to the use of the Matlab program GUIDE. GUIDE is a program that allows users to make GUIs with a simple visual interface. This interface allows students to drag and drop any options they want to add. While GUIDE is sufficient for making quick and simple GUIs, it ultimately lacked the options which were required to make our GUI successful.

The user will start by clicking the "Select PSD" button. This action will launch a file explorer window that the user can use to locate a .mat file containing a PSD, either of their own creation or one provided with the program. If the selection is not the proper format, the program will display an error message and restart the file explorer prompt. Next the user will click the "Output File Name:" button. This will launch a text dialog box in which students can enter the name they wish to save the output file data as. If this file already exists in the

current directory, the user will be warned. The user will then click the “Connect Arduino” button. This will launch a dialog box that will ask students to enter the comport that the AD is connected to, such as COM3. The program will first check to make sure that comport is open and not being currently used by Matlab or another program. If the comport is being used by Matlab, then the program will close the current connection and establish a new one. The program and the AD will communicate through an 115,200 baud serial channel connection, which is the fastest available option for the AD. If the connection is successful, MRIS will send an introductory handshake character to the AD and wait for the response. If no response is received or something else goes wrong when connecting, the user will be given an error message and asked to try again.

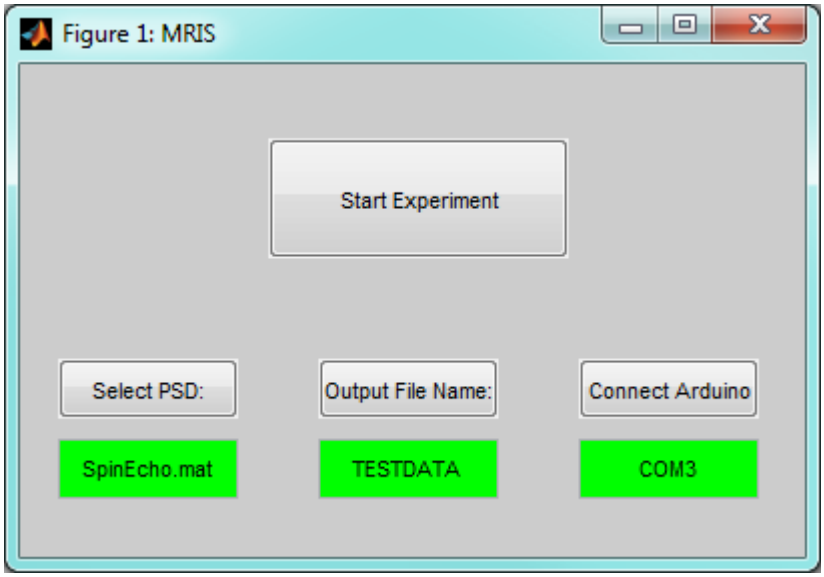

Figure 3.2(B): MRIS GUI panel when “Start Experiment” becomes active.

If all of the preceding steps were successful, the GUI will look something like Figure 3.2(B). It is at this point the “Start Experiment” button will become active. If the user selects this button, all of the visual elements will be removed and replaced with text saying “Starting Experiment.” The program will send a character to the AD indicating that it is going to begin

sending the PSD. It will first send the dimensions of the user provided PSD matrix, allowing the AD to create a matrix to store the PSD. If successful, MRIS will send the PSD one entry at a time. When the full PSD is read and stored, the AD will begin to run the experiment, and the GUI will display “Running Experiment.” At the same time, the program will be waiting to read and store each line of data coming from the AD. It will continue this process until receiving the finished command from the AD. If there is an issue that occurs sometime during the experiment, an error message will display the last known event and point the user towards a help file provided with the program.

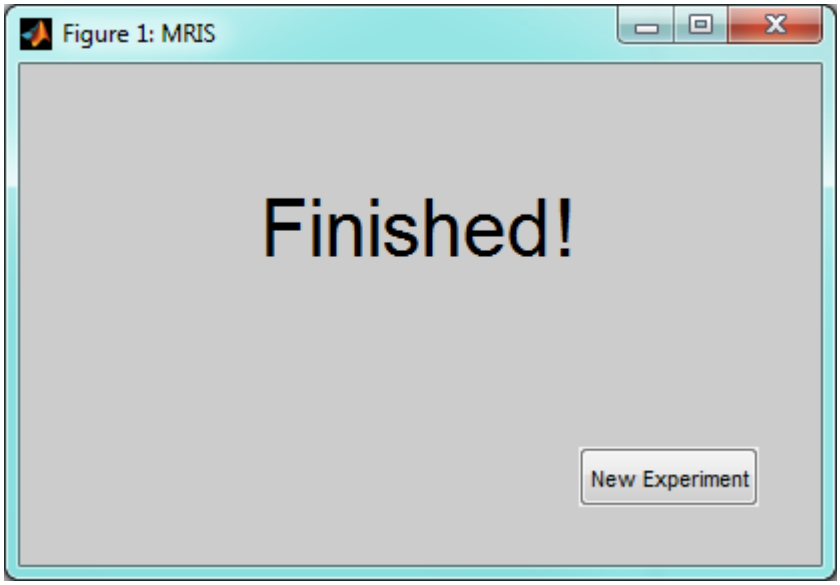

Figure 3.2(C): MRIS GUI panel after experiment is finished.

After the experiment is completed, the program will save the data with the specified file name and will display the screen shown Figure 3.2(C). If the user selects the “New Experiment” button, the GUI will return to the starting screen.

3.3 Associated Documents and Availability

The Matlab and AD program, along with any other completed files associated with this project are hosted on the popular code sharing site, GitHub, and are publically available for download. The repository containing the most up to date programs and files can be accessed at https://github.com/DKSMRI/Desktop-Sized-MRI-Project. Apart from the core programs, the hosted files will in the future contain a help document detailing how to use the Matlab program and some simple PSDs in a .mat format. Along with these programing files and this report, a more detailed design and construction file of the MRI system will be hosted so that groups seeking to recreate this design can find all of the documentation they need in one location.

4 HARDWARE

4.1 Original Design

The original design goal was to build a desktop-sized MRI with a .2 Tesla, permanent magnet. The electronics were to be either purchased inexpensively or designed and hand-built using inexpensive circuit elements. The final cost was to be under $3,000.00 with the majority of the cost coming from the anticipated cost of the magnet at approximately $2,000.00.

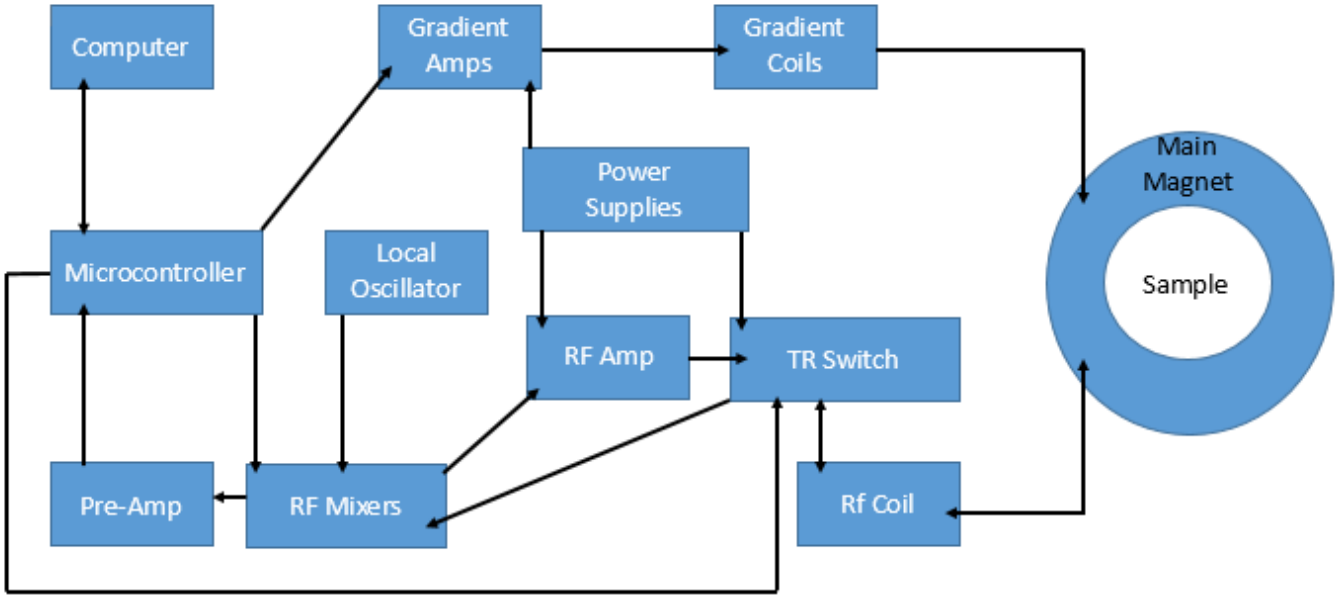

Figure 4.1(A): Simple block diagram of the purposed desktop MRI system.

A simple block diagram of the primary components of the MRI system is shown in Figure 4.1(A). At the beginning of an experiment, the user will send a PSD to the Arduino microcontroller (MC), from Matlab. The MC will use the PSD to generate three outputs to control the each of the three gradient coils, but only after going through three gradient amplifiers to provide the power to generate the fields necessary for imaging. The three gradient coils would be placed inside of the main magnet bore, and would have to be made according to the dimensions of the magnet.

According to the PSD, the MC will also create an output that will be used to generate the RF pulse. The AD output must be first sent through a mixer in order to increase the frequency into the MHz regime. The output will go through a RF power amplifier before traveling through a transmit receive switch (TRSW or TR Switch) that is set to transmit to the RF coil. The TRSW allows us to use the RF coil not only for transmission of a high-power RF pulse, but also for receiving signals from the sample. Like the gradient coils, the RF coil must be built around the specific inner dimensions of the main magnet. The TRSW would then be set to receive signal from the coil. The incoming signal is then sent through a mixer to a preamplifier connected to the MC. The output of the preamplifier would then be sampled by the MC and then sent back to the computer for analysis.

4.2 Design Simplifications

After plans to purchase the main magnet fell through, we were forced to simplify and change the design in order to build and test the remainder of the hardware. Instead of a commercially available permanent magnet, we decided to use the field generated by a clinical MRI system for testing. We relied on prebuilt components such as: local oscillator, Broadband RF amplifier, and a specialty SIEMENS preamplifier. These components are unlikely to be used in any final design due to their specialty and high cost.

Without the specific dimensions of the magnet to work with, any coils we build will be unusable in the final system. For this reason, we decided to focus on building a single gradient coil and RF coil of arbitrary dimensions.

4.3 Gradient Amplifier

The gradient amplifier (GA) connects the output of the AD PWM gradient control to the gradient coil. The GA is a necessary intermediary because the AD is not designed to output high power. The GA is composed of two main components, the MOSFET driver and the H-bridge circuit. Both of these components rely heavily on transistor circuitry, which I was not familiar with when I started. Before continuing, I learned basic transistor theory and common transistor circuits from a book entitled *Electronics,* written by Allan Hambley [7]. I then used a circuit design and simulation software called, Multisim, to build and test simple transistor circuits.

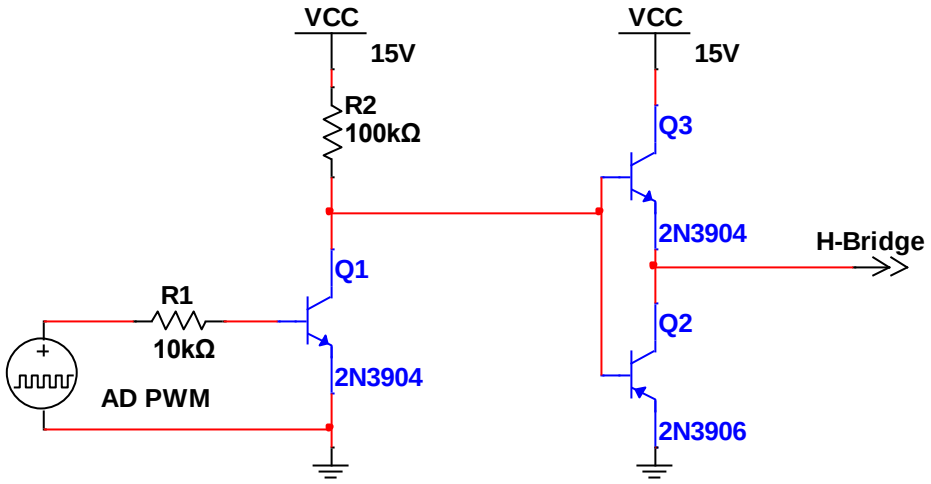

Figure 4.3(A): Simple MOSFET driver Multisim circuit diagram.

Figure 4.3(A) shows a simple MOSFET driver circuit of which we will need two. They will be controlled with a single AD PWM output branched into two signals, with one going directly to the driver, while the other is first inverted before entering the second driver. These outputs then enter into an inverting, voltage amplifying stage. The increased voltage signals then enter an inverting, BJT current amplification stage. The net result of these two stages are two outputs with amplified voltage and current that are reciprocals of each other. Instead of having to build all of

these stages with individual components, we were able to find a pin package, the TC428CPA, which contains all of these circuit elements in a small convenient package. The MOSFET driver outputs are then connected to the H-bridge.

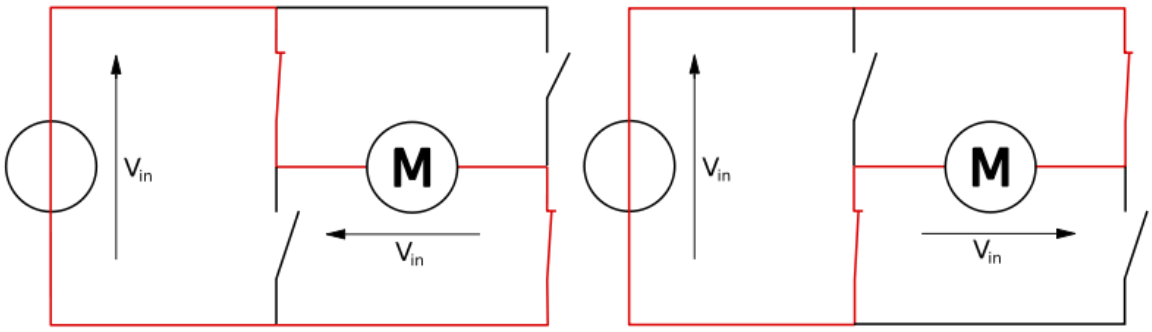

Figure 4.3(B): Simple diagram of a H-Bridge circuit showing the two possible current paths.

An H-bridge is a common circuit that utilizes four switches that, when activated in a certain order, can drive current through a load forwards or backwards. A simple H-bridge circuit operating in its two primary states is shown in Figure 4.3(B). In our H-bridge, two p type MOSFETs (pmos) and two n type MOSFETs (nmos) are used for switches where the load is the gradient coil. The state of each of the switches are controlled with the outputs of the MOSFET driver. The combined MOSFET driver, H-bridge GA is shown the Multisim Circuit diagram in Figure 4.3(C).

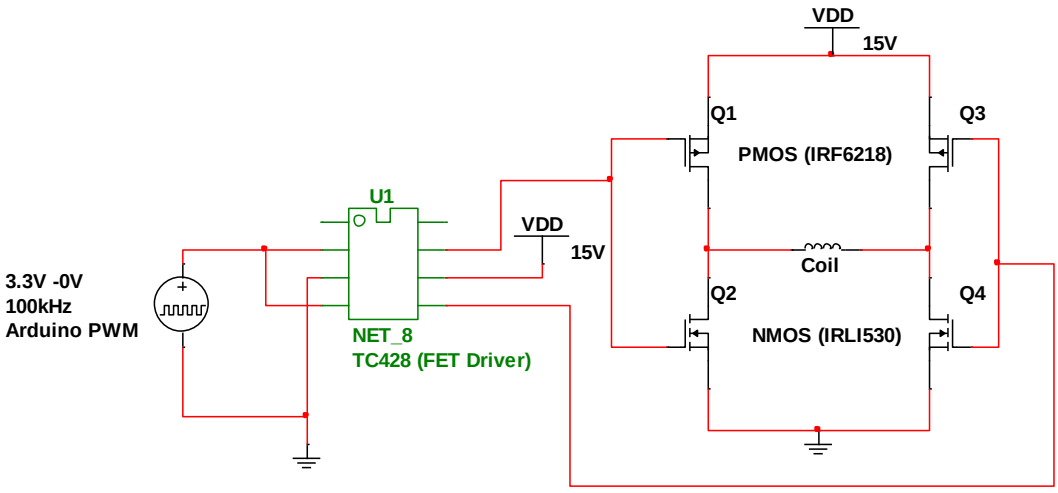

Figure 4.3(C): Gradient Amplifier Multisim circuit diagram.

The two sides of the H-bridge are driven with inverted logic inputs. This means that at any given time there is only one path for current to flow through the bridge. If the AD PWM output is at 3.3 V, then the current will flow from the 15 V power supply to Q3, through the coil, to Q2, and then to ground. If the PWM output is at 0 V, then the current will flow from the 15V power supply to Q1, through the coil, to Q4, and then to ground. The pmos (IRF6218) and nmos (IRLI1530) were selected for their ability to handle high power and quickly switch between states.

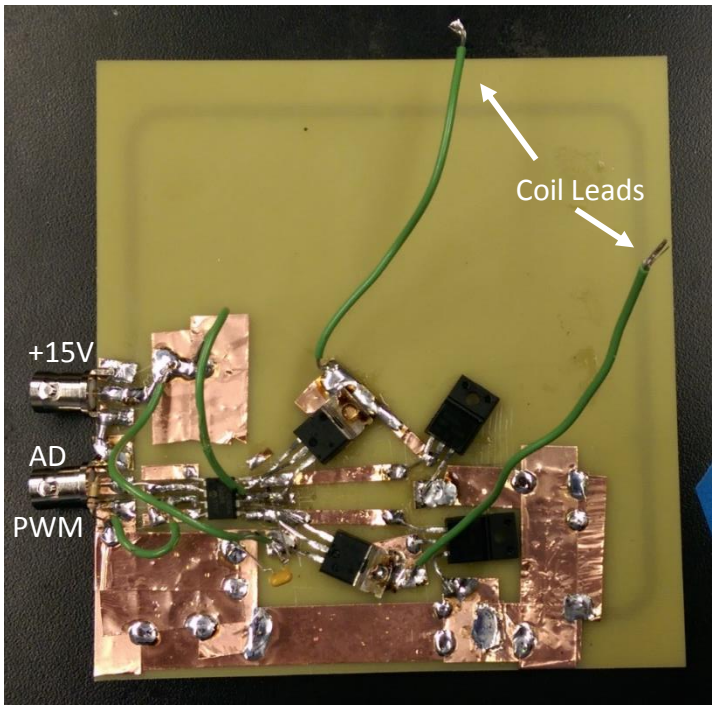

Figure 4.3(D): Picture of the final Gradient Amplifier Circuit.

Due to the high power flowing through the GA, it cannot be laid out on a simple bread board. It must be built using a printed circuit board. Due to time constraints, I constructed this using "Dead Bug" circuit design. "Dead Bug" circuits are made by laying down copper tape and creating a large grounding plane. Then additional pads, which are isolated from ground, are added to make connections between different circuit components. The name "Dead Bug"

comes from the visual likeness of a pin package placed on its back to a dead bug. The final GA circuit is pictured in Figure 4.3(D).

4.4 Gradient Coil

The gradient coil takes in a high-power input across its leads and uses it to generate a linear field inside of its bore. Much research has been done to find coil designs with high linearity and that are easily constructed. In general, increased power and linearity leads to better signal to noise ratio, typically with the tradeoff of simplicity. Since our system focuses on simplicity, the original system design utilized two Golay pair coils for x and y gradient generation and a Maxwell pair coil for the z gradient. Due to the simplification of the original design, only the z-gradient coil was made.

Our gradient coil was made using two 20 turn bundles of tightly bound, counter-wound 16 AWG copper wire wrapped around a 5 cm diameter, plastic tube. Due to the practical difficulties of keeping the bundles bound, they were wrapped with plastic cable ties. A picture of the gradient coil can be seen in Figure 4.4(A).

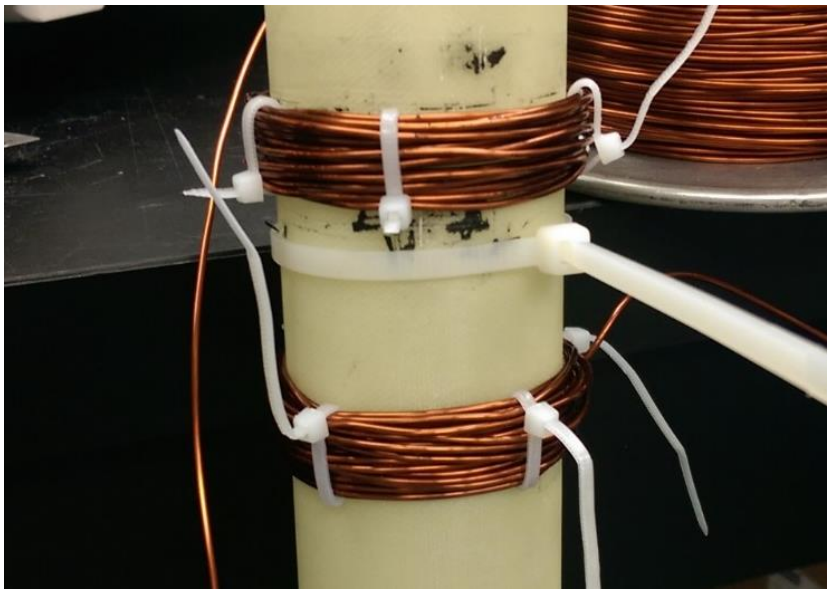

Figure 4.4(A): Picture of completed Maxwell Pair Gradient Coil.

By estimating the length of the wire, we can use the well-known material properties of copper wire to estimate the resistance of the wire.

$$L \approx 14.5m, R(Theory) = L * \frac{\Omega}{L} = 14.5m * 13.17 \frac{\Omega}{km} = .191\,\Omega$$

$$R(Measured) = .2\,\Omega$$

We can see that the measured resistance matches well with the theoretical estimation. With both the resistance of the bundle and the max allowed current of 3.7A, we can use ohm's law to find the approximate maximum voltage across the coil of .74 V.

To test the quality of the gradient coil, the coil leads were connected to the output of the GA and driven with 1 amp controlled by the AD. The field produced by the coil was measured along its central axis with a Gaussmeter. Due to the simplicity of the coil design, the theoretical field produced could be found using the Biot-Savart law, Eq. 4.4(A).

$$B = \frac{\mu_0}{4\pi} \int_C \frac{Idl\ x\ \hat{r}}{|r|^2}\ Eq.4.4(A)$$

Using Eq 4.4(A), we can find the field produced on axis for a single coil on origin Eq. 4.4(B).

$$B(z) = \frac{\mu_0}{2} * \frac{R^2 I}{(z^2 + R^2)^{\frac{3}{2}}}\ Eq.4.4(B)$$

Where μ_0 is the magnetic permeability of free space, R is the radius of the loop, I is the current, and z is the distance along its central axis.

To find the magnetic field produced by our coil, we have to sum the contributions from the 40 loops of current, half positive and half negative. A plot of the measured magnetic field and theoretically derived field is shown in Figure 4.4(B).

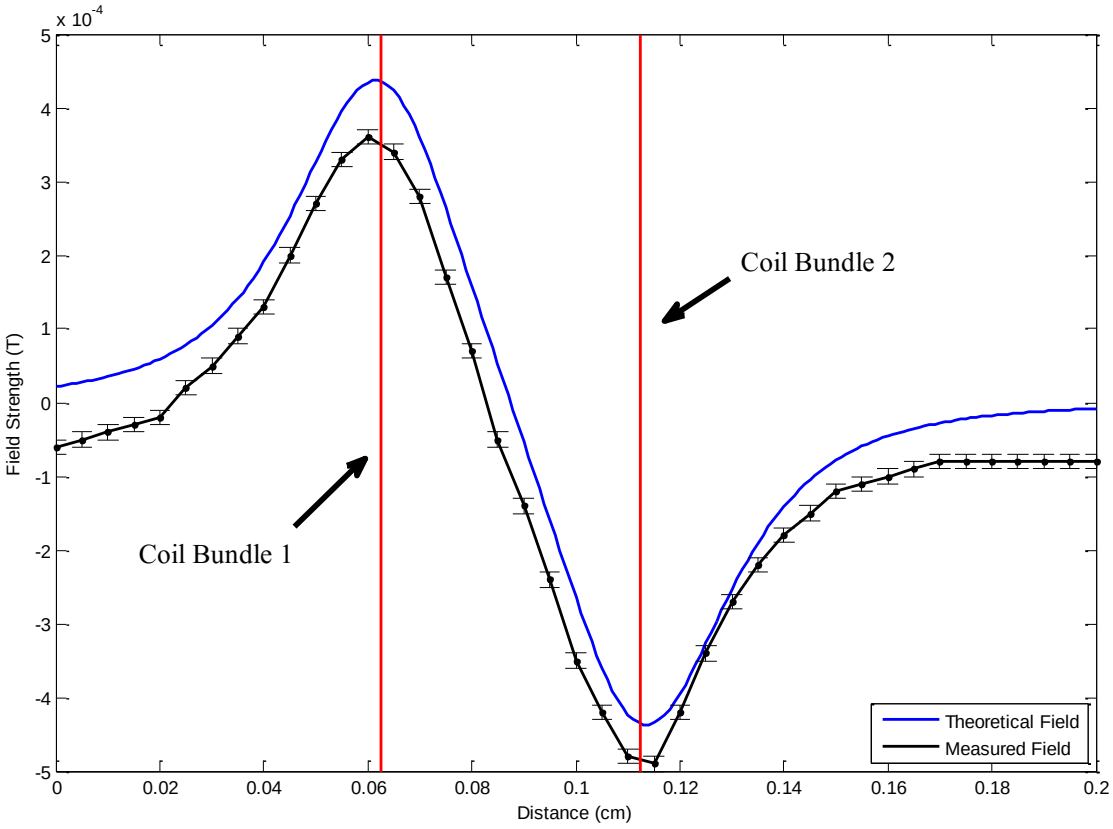

Figure 4.4(B): Field strength of the gradient coil versus distance.

While the offsets are different between the measured and theoretical curves, they both share the same shape. These offsets are most likely due to the contribution of both the earth magnetic field and the magnetic field created from current following through the wire leads heading to the bundles. The shape of the curves between the two bundles is the most important aspect of the plot. For accuracy in our images, we need a linearly changing magnetic field between the bundles. The slope of the inner regions were found to be

$$\frac{dB}{dz}(Measured) = -19 \pm 1 \frac{mT}{m}, \qquad \frac{dB}{dz}(Theory) = -20 \frac{mT}{m}$$

The measured value agrees with the theoretical value. Typical gradient coil field strengths are between 1 mT/m and 100 mT/m, which means that this coil can be used as our gradient coil.

4.5 RF Mixers, Amplifier and Preamplifier

Mixers, in general, are devices that take in two inputs and nonlinearly combines them to form a new output. Frequently the two inputs are of different frequencies, f_1 and f_2, and the mixer will multiply them together. This process produces a new output signal with frequencies of $n*f_1 + m*f_2$, where m and n are positive or negative integer values not including zero. Typically, one of the input signals is of a well-defined frequency generated from either a variable-frequency oscillator or a function generator. This input of a constant, well-known frequency is typically called the LO input. The remaining input is typically called the RF channel and the output is called IF channel. Mixer circuits are typically used to shift the frequency band of an input signal up or down to a new frequency band. Our MRI system utilizes two mixer circuits.

Before we could utilize any mixer circuits, we needed to have the LO input. In the original system design, we were going to build one from scratch. We used instead the available Wavetek 1062 HF Sweep Generator as our LO for testing purposes. The LO will be connected to the two different double balanced mixers. The “double balanced” term distinguishes it as a mixer which heavily suppresses the original frequencies of both the RF and LO inputs. A simple circuit diagram of the two mixers is shown in Figure 4.5(A).

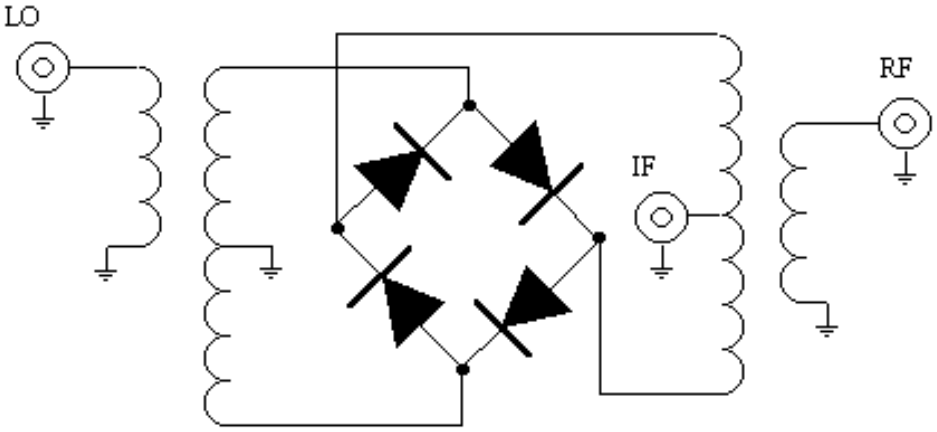

Figure 4.5(A): Simple double balanced mixer circuit diagram.

The diodes featured in Figure 4.5(A) are 1N4148 high-speed diodes that are used because of their fast switch times. The loops which appear to be inductors, are actually transformers. The transformers used were handmade using two, thin, colored, magnet wires and an m6 plastic washer. To begin, the two magnet wires, with colored epoxy coatings, were twisted around each other. The twisting was done by sticking one end of each of the wires in an electric drill and the other ends in a clamp. The drill was then spun until there were about 10 twists/inch in the wire. The result is called bifilar wire and is pictured in Figure 4.5(B).

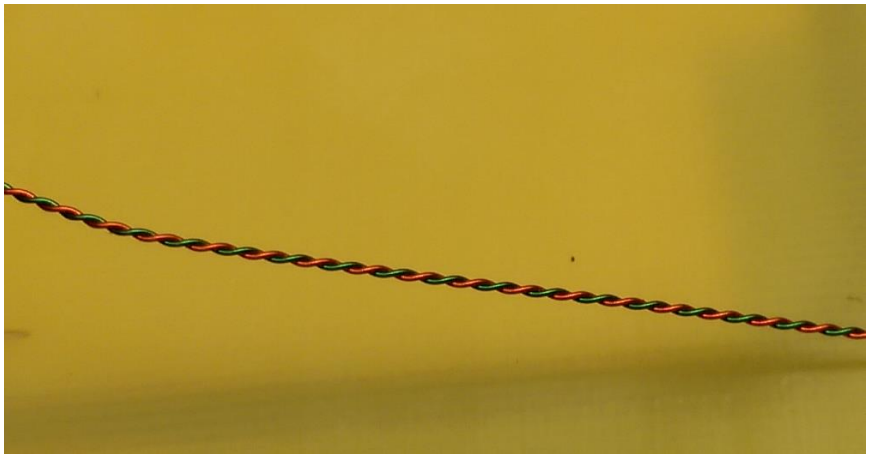

Figure 4.5(B): Picture of bifilar wire with approximately 10 turns/inch.

The bifilar wire was then cut into eight identical lengths of about 8cm. For each transformer, two of these lengths were required. These two lengths of bifilar wire were wrapped tightly around the washer a total of 5 times, creating a simple transformer. A sample transformer is pictured in Figure 4.5(C).

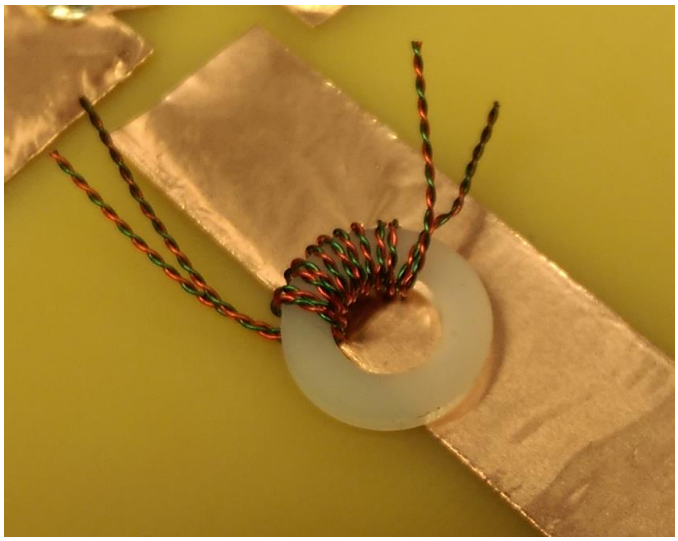

Figure 4.5(C): Picture of transformer made with two lengths of bifilar wire.

With all of the necessary parts available, the mixer circuit was then made using “Dead Bug” circuit design, with a large grounding plane on the underside of the circuit. A picture of the final mixer circuit is shown in Figure 4.5(D).

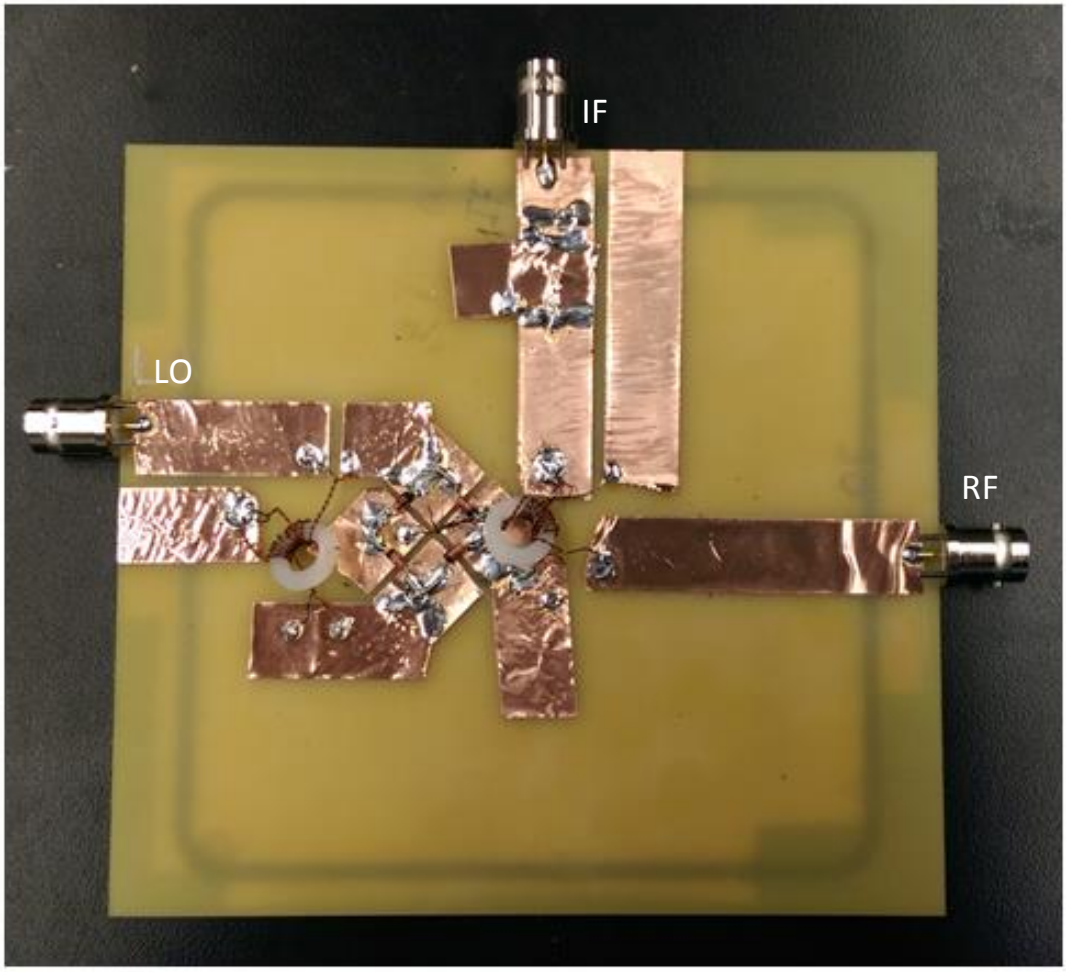

Figure 4.5(D): Photograph of the final mixer circuit.

To test the quality of our mixer circuit, we utilized an Agilent N9000A Signal Analyzer. The LO pin was connected to a 64MHz 1.3Vpp output from the Sweep Generator. The RF port was connected to a DC power supply which was slowly varied between .14V and 1.18V. The IF output was connected directly to the spectrum analyzer. At a set of DC power supply voltages, the attenuation of the LO signal was recorded. The results are displayed in Figure 4.5(E). The test clearly shows that the RF waveform can be controlled using a DC voltage all the way up to GHz levels, which is the regime we need to work in. Moreover, we can see that there is a region of linear response from .53V to .89V.

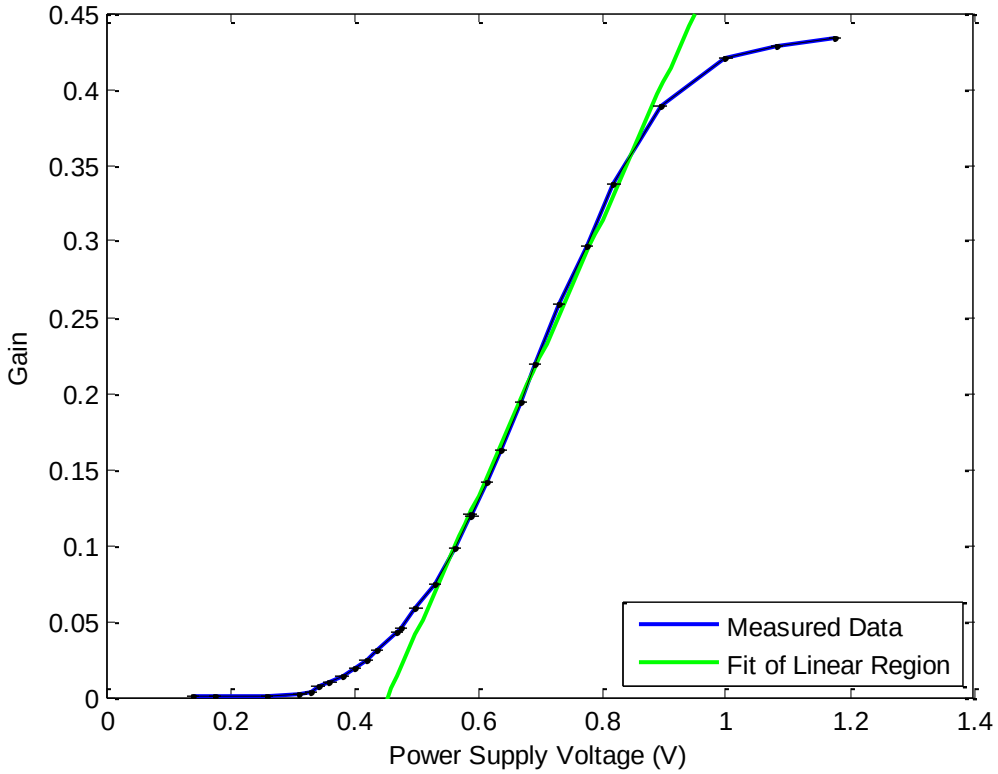

Figure 4.5(E): Graph of gain through the mixer as measured by the network analyzer versus the power supply voltage. The linear region was fit and yielded y = .91x - .41 with an R^2 = .9971.

With the mixers confirmed working, they can now be integrated into the system. One mixer will have the AD RF output signal connected to the RF port, the Sweep Generator as the LO, and the IF output going to the RF amplifier. The remaining mixer will have the RX port of the TR switch connected to the mixer RF port, the Sweep Generator as the LO, and the IF output going to the preamplifier.

4.6 Transmit Receive Switch

The purpose of a Transmit Receive Switch (TRSW) is to allow a single RF coil to be used for both transmission (TX) of a RF pulse and for reception (RX) of a signal coming from the sample. When in the TX on state, the TX port is directly coupled to the RF coil and isolated from the RX

port. When in the RX on state, the RX port is directly coupled to the RF coil and isolated from the TX port. In both states, the RX must always be isolated from the TX. If the two ports were coupled, the high-power input from the TX would overwhelm and destroy the sensitive preamplifier connected to the TX port.

Due to the our high-power, high-isolation and fast switch time requirements, our TRSW relies on two single-pole single-throw (SPST) switches, made with two pin diodes (MA4P740F-1072T) that are controlled independently from two AD logic signals. The overall design was based on a TRSW created by Michael Twieg [8]. A Multisim circuit diagram is displayed in Figure 4.6(A).

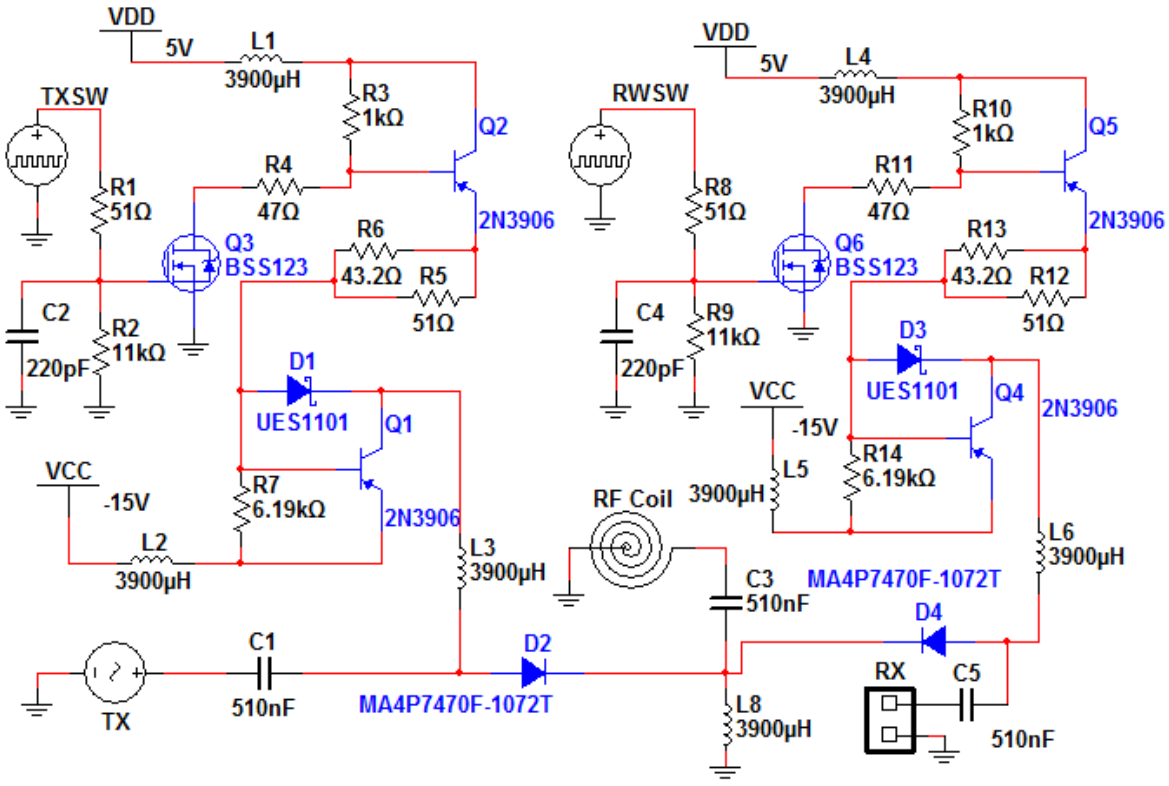

Figure 4.6(A): Multisim circuit diagram of the Pin Diode TRSW.

While, the design may look complicated, it can be broken down into simple groupings. The circuit is essentially symmetric around the RF Coil. The upper left branch is designed to map

the transmit switch (TXSW) logic signal from the AD to -15/4V from 0/3.3V and apply it to the node between C1 and D2. The upper right branch is designed to map the receive switch (RXSW) logic signal from the AD to -15/4V from 0/3.3V and apply it to the node between C5 and D4. All of the inductors in the circuit are added to prevent any RF power from leaving the lower RF rail and going to the power supply or the AD and damaging them. The capacitors on the RX, TX, and RF Coil ports were added to prevent any DC from entering these devices and damaging them.

The TRSW was built using "Dead Bug" circuit design. Since we are now dealing with RF, the current paths must be as short and symmetric as possible, and we must also have a larger grounding plane to prevent attenuation. To ensure a large grounding plane, the circuit was built on a large copper covered board. All connections between other components were created using isolated pads made by cutting through the copper surface to the board underneath. The built circuit is shown in Figure 4.6(B).

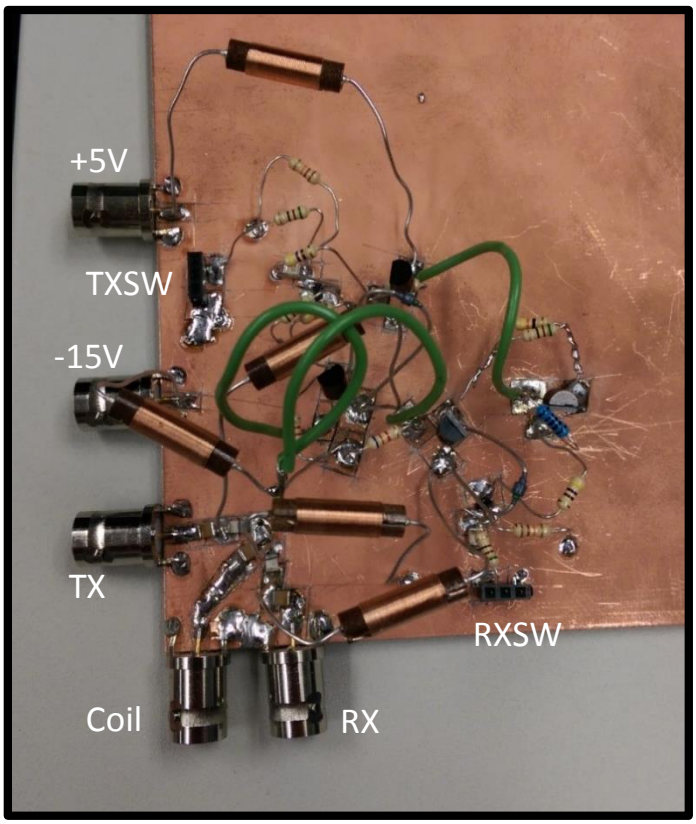

Figure 4.6(B): Photograph of the final TR Switch.

To test the quality of the TRSW, we used an Agilent E5061A network analyzer. After connecting the power supply inputs and AD logic inputs, we connected two of the three ports (TX, RX, Coil) to the network analyzer and a 50 ohm terminator to the remaining port. Then we activated either the TXSW or the RXSW and measured the attenuation across the all of the possible port combinations. The ideal and measured values for the RX on state are shown in Table 4.6(A) and the TX on state in Table 4.6(B).

Measured (RX ON)	TX	RX	Coil
TX	N/A	-36 dB	-36 dB
RX	-36 dB	N/A	-.29 dB
Coil	-36 dB	-.29 dB	N/A
Ideal	TX	RX	Coil
TX	N/A	-∞ dB	-∞ dB
RX	-∞ dB	N/A	0 dB
Coil	-∞ dB	0 dB	N/A

Table 4.6(A): Summary of measured and theoretical attenuation across TRSW ports with RX on.

Measured (TX ON)	TX	RX	Coil
TX	N/A	-36 dB	-.28 dB
RX	-36 dB	N/A	-36 dB
Cable	-.28 dB	-36 dB	N/A
Ideal	TX	RX	Coil
TX	N/A	-∞ dB	0 dB
RX	-∞ dB	N/A	-∞ dB
Coil	0 dB	-∞ dB	N/A

Table 4.6(B): Summary of measured and theoretical attenuation across TRSW ports with TX on.

As we can see from the two tables above, the TRSW behaved as the theory of our design dictated. Instead of a complete isolation between two ports that should be totally isolated, we found about a -36 dB attenuation, which was high enough to prevent damage to any of our sensitive electronics. Instead of a unity gain between ports which should have been coupled, we found an approximately -.28 dB attenuation, which was suitable for our purposes.

4.7 Final Steps Required to Test System

Due to time constraints, we were unable to take the few final steps required to test our simplified system as a whole. Before testing is possible, a RF coil would have to be built on the inside of the gradient coil. The RF coil will be made using a simple birdcage coil design, and would require about eight hours to build and test. With the RF coil working, all of the individual hardware components would have to be connected, collected onto a cart, moved to the clinical MRI system, and setup. Then a set of simple 1D MRI experiments could be performed on a simple, readily-available phantom to test the performance of the system.

5 DISCUSSIONS

5.1 Discussions of Objectives

To summarize, my original objectives were:

1. Build a MRI machine which is portable, inexpensive and easy to use.
2. Design a program which will allow students to easily interface with the MRI machine and carry out their own imaging experiments.
3. Write an extensive set of documentation describing every aspect of the MRI machine to allow students to understand the device they are using and also allow other groups to follow the same steps to make their own MRI machine.

I have conclusively finished only one of my original objectives. I was successful in both programming the microcontroller to run the system and also in programming the Matlab GUI to allow students to easily interact with the microcontroller. While I have not completely finished neither the first nor the third objective, I did make significant progress on both. This report will ultimately serve as the primary documentation of the project and process behind the work which has been completed.

In retrospect, the project was likely too ambitious to be completed over the course to two semesters.

5.2 Learning Points

While I had hoped to be able to finish more of my project, I still learned important lessons about doing research in general and important skills which I will be able to utilize for the rest of my life.

On future projects, I will begin by learning more about the project as a whole and how each aspect is interconnected. This information will not only give me a better appreciation of the project, but also allow me work on different parts of a project concurrently. Working on multiple aspects at once would minimize the effects of an unexpended delay on the overall progression of the project as a whole.

Through this project I learned a variety of useful skills including:

- Arduino/C++ Programming
- Matlab Serial Communication
- “Dead Bug” circuit building
- Surface mount soldering
- Transistor, RF, and mixer circuitry

5.3 Future Work

There is both short-term and long-term work left to be done. The short-term work will involve making the RF coil and testing the system by acquiring images of sample phantoms. In total, this should only take about 10 more hours of work.

The long-term future work will involve redesigning the current system to meet the original design specifications. Primarily this will include buying a main magnet and designing and building three gradient coils and one RF coil based off of the magnet dimensions. We will also have to design and build a LO, preamplifier, and RF amplifier all of which must be inexpensive.

5.4 Conclusion

While the entire desktop MRI system was unable to be completed, substantial steps were made towards the final product. The AD programming required a significant portion of the project time to work through all of the timing and communication issues that arose. However, the AD program was successfully coupled with a Matlab interface, which will allow students, once the hardware is complete, to easily execute a variety of MRI experiments. All of the remaining work involves designing, building, and testing new hardware. A new main magnet will have to be found, and the prebuilt, expensive hardware will have to be replaced with inexpensive, hand-built circuits. If this project is continued, it is likely that this work could be finished and the system implemented by next year.

REFERENCES

[1] Haacke, M.; Brown, R.; Thompson, T.; Venkatesan, R.; *Magnetic Resonance Imaging*. WILEY-LISS. 1999.

[2] Beshberg, J.; Seibert, A; Leidholdt, E.; Boone, J. *The Essential Physics of Medical Imaging*. LWW. 2011.

[3] Macovski A. Conolly S. *Novel approaches to low-cost MRI*. Magn Reson Med 1993; 30(2):221-30.

[4] Shirai, T; Haishi, T.; Ursuzawa S.; Matsuda, Y.; Kose, K. *Development of a compact mouse MRI using a yokless permanent magnet*. Magn Reson Med 2005; 4(3): 137-143.

[5] Magin RL, Webb AG, Peck TL. *Miniature magnetic resonance machines*. IEEE Spectrum 1997; 34(10):51-61.

[6] Wright, S; Brown, G.; Porter, J.; Spence, D; Esparza, E.; Cole, D.; Huson, F. *A desktop magnetic resonance imaging system*. MAGMA 2002; 13:177-185.

[7] Hambley, A; *Electronics*. PRENTICE HALL. 1999.

[8] Twieg, M; M.S. thesis. Case Western Reserve University. 2013.