Software Defined Radio Based Frequency Modulated Continuous Wave Ground Penetrating Radar

Patrick Fiske
University of Vermont

Software Defined Radio Based Frequency Modulated Continuous Wave Ground Penetrating Radar

A Thesis Presented

by

Patrick Fiske

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Master of Science
Specializing in Electrical Engineering

October, 2020

Defense Date: August 7th, 2020
Dissertation Examination Committee:

Tian Xia, Ph.D., Advisor
Dryver Huston, Ph.D., Chairperson
Jeff Frolik, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of Graduate College

Abstract

Frequency modulated continuous wave (FMCW) radar allows for a wide range of research applications. One primary use of this technology and what is explored in this thesis, is imaging in the form of ground penetrating radar. To generate proper results, spectral wide-band reconstruction has been developed to overcome hardware limitations allowing for high resolution radar. Requiring complex reconstruction algorithms, the proposed method benefits greatly in terms of performance and implementation compared to other radar systems.

This thesis develops a wideband linearly frequency modulated radar leveraging a software-defined radio (SDR). The modular system is capable of a tunable wideband bandwidth up to the maximum SDR ratings. This high-resolution system is further improved through implementation of grating side-lobe suppression filters that correct for the spectral discontinuities imposed by the reconstruction. These grating lobes are managed through multiple techniques to alleviate any ghost imaging or false positives associated with object detection. The solution provided allows for generally non-coherent devices to operate with synchronous phase giving accurate sample-level measurements. Various corrections are in place as mitigation of hardware transfer functions and system level noise. First the system was theorized and simulated, illuminating the performance of the radar. Following development of the radar, measurements were conducted to confirm proper and accurate object detection. Further experiments were performed ensuring Ground Penetrating Radar (GPR) performance as designed. Applications of this work include Synthetic Aperture Radar (SAR) imaging, innovative GPR, and unmanned aerial vehicle (UAV) systems.

Acknowledgements

Throughout the journey of this thesis, there have been several individuals who have motivated, encouraged, and mentored me. Without these people and their support, I would not have been able to complete this thesis and have acquired the skills to develop as an engineer. Although I struggled to find any sort of motivation through my earlier education, my mother has not only raised, but inspired me to have the work ethic essential to complete this project along with many others. Without her influence and determination in parenting, I would have never been able to not only graduate with good standing, but continued on to achieve what I have. My brother, Brad has always been a role model and someone who I have emulated all throughout my academic career. In addition to all of the immense support my family has provided, they also have pushed above and beyond what was neccessary amidst the COVID-19 pandemic. I was provided a comfortable home to stay in and was also given materials to continue my testing for research. Shortly after beginning testing I broke my foot and was unable to continue with my work. My family provided an inside solution for testing and I am grateful.

I would additionally like to acknowledge the constant support of my advisor, Dr. Tian Xia. Your constant mentoring has taught me not just the necessary theory, but the skills integral for implementation and problem solving. Behind Dr. Xia stands a research team of myself and three others, Wenzhe, Yan, and HaiLong. I would like to thank all of them as they have each helped me throughout my time as a graduate student. Thank you to Dr. Frolik and Dr. Huston for being a part of my committee. Dr. Frolik has always been an exceptional role model and has taught several courses that I have been fortunate enough to participate in throughout my undergraduate education. I am grateful for all of the lessons and knowledge I have received from everyone.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Ground Penetrating Radar's (GPR) existience begin in the early 1900. The ideas proposed were limited by the technology of the time, but there were some significant radar advancements especially after military applications arose. During the World Wars, the demand for radar systems skyrocketed into what it is today which superseded past technologies. Military defense companies consider radar systems to be some of the most important technological advancements in their line of work. In more recent times, the use of radars has increased in academia and in civil applications. The resulting improvements have allowed for more affordable systems creating a cycle of compounding innovations.

The research team led by Dr. Tian Xia has been undergoing the development of a functioning GPR system with plans of various applications. A current system is in place, but is limited by the techniques and implementation. This project overcomes such limitations and has several benefits that will expand the use of the system. With the proposed method, several techniques to design GPR will be viable, along with Synthetic Aperature Radar (SAR) applications. Not all of these will be explored in depth for this thesis, but the development of the radar system with these capabilities will be conducted. Through the use of Software Defined Radio, (SDR), the system will be fully modular within the limitations of the equipment specifications.

As a GPR, the system will be capable of detecting subsurface objects along with general target ranging. Through the proposed method, airborne techniques will be possible and currently are undergoing separate development. An airborne system has the potential of landscape SAR imaging and also ground penetrating applications, providing endless possibilities for expansion and innovation.

1.2 Problem Statement

Developing a GPR system on a printed circuit board (PCB) has an associated cost including the bill of materials, board development, and soldering. All of this accrues and results in a rather expensive system. With the aforementioned implementation, the system can only have slight modifications, but generally is limited to performing the same task. If a related system or technology is desired, the process must be repeated and a new board must be developed, meaning the overall capital required to perform varying tasks can quickly escalate. Through the implementation of SDR, one base price is allocated for the system. Now the device can be coded to perform whatever task is desired, given that it is within the capabilities of the radar. Designing a system with an SDR provides a completely modular solution allowing the user to adapt to their application of choice without purchasing additional hardware.

This thesis presents the design of a radar system used for GPR. The design will be implemented on an Ettus Research B210 SDR and will leverage frequency domain reconstruction with a linearly frequency modulated (LFM) signal to achieve a synthesized wideband signal. In addition, post processing will be completed for the system and GPR plots will be constructed. This radar system will be capable of achieving a tuneable bandwidth only limited by the hardware in use. Grating side lobes resulting from the reconstruction will be mitigated. The work presented is unique to the research team at the University of Vermont, but not to the industry as a whole.

1.3 Contributions

There are several academic papers providing similar solutions to the problem statement, but a few of them had a significant influence of the design of the system. Listed below are the sources that provided the greatest contribution to the thesis. The culmination of this literature facilitated the success of this thesis. Detailed derivations and algorithms allowed for the replication of the proposed systems.

1. *High Range Resolution Radar using Narrowband Linear Chirps offset in Frequency.*

   This paper lays the groundwork of LFM reconstruction using narrowband chirps. The most important details of the reconstruction are presented including methods of combining sub-pulses, phase correction, frequency shifting, and many more. There were minimal results presented, but the theory illustrated proved significant.

2. *Bandwidth Synthesis for Stepped Chirp Signal" A Multichannel Sampling Prospec-*

tive

Modeling the stepped frequency technique is a difficult task and is presented in this paper. The analysis of complex demodulated tranceived signals is outlined in detail and was necessary for the implemented of the algorithm presented later. Both time and frequency domain representations were outlined for use.

3. *Ultra-Wideband Synthesis for High-Range-Resolution Software Defined Radar*

   This conference paper was emulated throughout the project because of its detail and matching application. A similar Ettus Research SDR was used and the algorithms were derived from the previously mentioned contributions. This paper holds the innovations of the frequency stacking method and grating lobe suppression that was directly used in this thesis. Results from this report include loopback verification, field range testing, and finally GPR scans using the matched filtering method.

1.4 Thesis Outline

This thesis abides by the following template : Chapter 2 provides an introduction to GPR and radar with the theory behind it. Chapter 3 will discuss the available methods to implement the proposed system. The varying degree of effectiveness will be analysed for each technique. The fourth chapter provides an in depth analysis of the hardware leveraged in the thesis and the necessary corrections made. Chapter 5 outlines the signal processing methods and the development of the phase coherent LFM signal. The sixth and final chapter presents the simulations and results. Future work and final words will be presented.

Chapter 2

Ground Penetrating Radar

2.1 Introduction

This chapter will delve into the background of ground penetrating radar. Following a breif history of the topic, the methods of GPR scans will be discussed, along with the considerations of practical implementation. The baseline processing methods for post-collection scans will be detailed with example simulations.

2.2 History

GPR has a relatively long history in regards to current technology. As early as the 1920's there have been scans recorded from GPR surveys. These used simple radar systems, with basic processing techniques. With the continuing advancements of radar systems, GPR is still a valid and frequenly used subsurface imaging technique. Commercial use of GPR began in the 1970s' [8] and today is used for military, commercial, and academic applications. Currently there is a push to develop new ways of conducting not only GPR images, but SAR images. These both share the same radar technology and often are implemented by the same system.

2.3 GPR and Radar Background

2.3.1 GPR

As previously stated, this chapter will discuss the inner-workings of GPR systems and how scans are conducted. General theory and processing techniques will be discussed. There will be slight detailing of some radar fundamentals and the importance of knowing the scanned environment.

For all types of GPR,the radar transmits and then receives a signal and leverages the response to generate an image. Based on the magnitude and phase response of the transmitted signal, the two way travel time can be determined, providing the information of the object that caused the reflection. If known, permeability of the medians the wave has traveled through allow the distance of the object to be calculated and the target can characterized. This process is the most basic representation of radar imaging.

Given that a system can calculate how far away an object is, iterating this process at a known interval of distance will form what is known as a "radargram". These types of figures are the generic GPR imaging plots. The process for obtaining these images is depicted in figure 2.1. The GPR unit will travel laterally and take a scan every distance interval and store the data for future processing.

A typical radar response from example data is shown as in figure 2.2 with time on

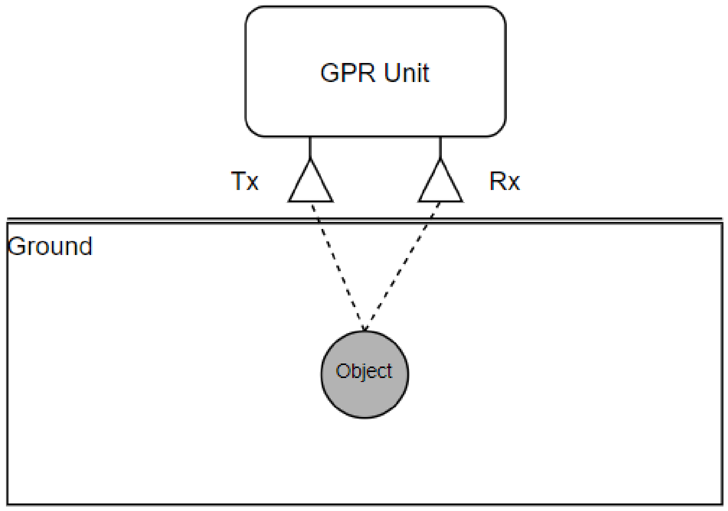

Figure 2.1: General GPR Configuration

the x-axis and amplitude on the y-axis. This response is then inverted and stored in a two dimensional matrix containing the individual receive signals stored in columns. These individual signals are known as "A Scans". Once placed in the matrix where each column is an A scan,the data can be plotted with MATLAB's function "imagesc" or similar variants. These functions associate the magnitude of the signal to a color scale, which outline the contrast between a zero value and the signal peaks. Plotting the matrix of collected data will produce the B scan image. Following the generation of these figures, there are several techniques for image enhancement and filtering.

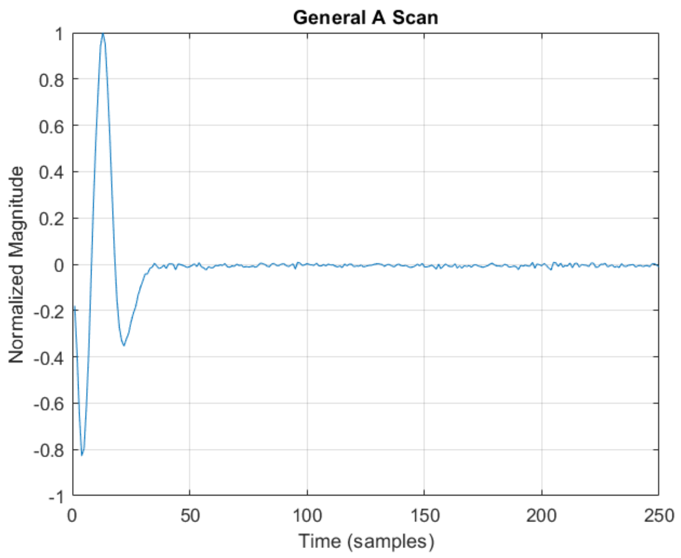

Figure 2.2: General A Scan

Frequently GPR is used to detect objects such as pipes, cables, or any other buried object of importance. Figure 2.3 shows the typical hyperbolic shape associated with a rounded object. The work presented in this thesis will attempt to replicate this shape. Creating this classic hyperbolic shape requires evenly spaced scans and a pipe-shaped or round object. As a GPR unit moves laterally over an object, at the farthest point of detection, the peak in the reflection signal takes the longest time to travel to the receiver. As the GPR device continues, this peak returns to the receiver in a relatively decreasing time until the system is directly above the object. Similarly, when the unit moves away from the object the reflection time increases and the target appears farther away. The combination of these acts produce the hyperbolic shape.

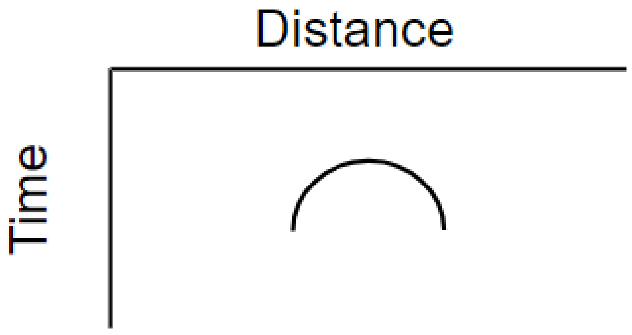

Figure 2.3: General Hyperbolic Response

2.3.2 Implementation

Theory

Conducting a practical GPR scan has several variables impacting the image quality. There are multiple processing methods that can be performed to not only enhance the performance, but correct for general errors. These methods range from basic smoothing filters to more advanced background removal.

To begin, it must be realized that in almost all textbooks and sources for radar theory, the term for speed of light, c, is used in the equations for range calculation. In ideal scenarios this is acceptable, but when calibrating recorded GPR data, there are a few factors that augment this value. Permeability is defined as the characteristic of a material or membrane that allows liquids or gases to pass through it. With radar systems, permeability refers to the ability for radar waves to travel through a medium and this directly impacts the speed of the wave. Depending if the GPR is operating through sand, gravel, soil, wet soil, or anything else, the permeability will alter the signal speed. If not accounted for, this will result in distance miscalculations. This error holds the potential to cause serious issues is some circumstances such as an excavation of a construction site. Table 2.1 contains permeability values for common materials along with the associated speed of light [8]. In practice this is often taken into account by multiplying the speed of light with a velocity factor that is derived from the permeability of the medium. It is also always important to perform calibration on the system to account for wires or other objects effecting the distance calculation.

Medium	Typical Transmission Velocity (m/ns)	Relative Permeability (E_r)
Ice	0.15	4
Dry Sand	0.2-0.12	2-6
Wet Sand	0.095-0.055	10-30
Concrete	0.1	9
Dry Soil	0.1	9

Table 2.1: Common Permeability and Velocities

Data Processing

Following a GPR scan, it is often desired to perform some sort of processing to enhance the constructed image. This generally is important for all GPR applications, but the chosen method and it's importance varies between scans. When purchasing commercial GPRs, users are often provided filters that can seamlessly be applied to the data. It should be noted using the wrong filter could prove detrimental to the image. This section will detail some of the more important filtering techniques that are applied to most GPR systems. Sample data was found prior to the development of the radar and was used to implement some basic techniques [30] such as background removal, zero time removal, and variable gain.

In most cases, the GPR system is mounted on some sort of mobile chassis and scans are taken at linearly incremented distances. Physically speaking, the antennas are relatively close to the ground which produces an undesired peak at the start of the A scan. This is referred to as an air wave, or ground reflection and generally is constant throughout scans. There are a variety of techniques to mitigate or even completely remove the artifact. The simplest being a simple trimming of the data, but there are involved methods that produce a more desired response.

Below shows the implementation of the air wave removal. There is no information associated with the example data, but it appears that there is one pipe-shaped object buried 4.5 meters into the scan. The medium being detected through is also unknown so the y-axis has been left in terms of two-way travel time. On the left image the raw

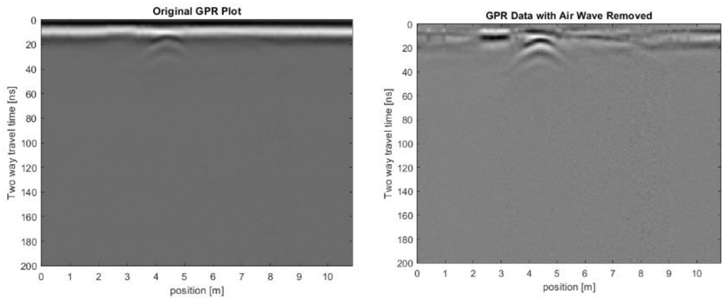

Figure 2.4: Sample Data Air Wave Removal

data is plotted. At about $10ns$ there is a constant air wave that is filtered with the results in the right plot. This processing is completed by leveraging a moving average filter that subtracts the signal mean from each sample. The filter effectively is a smoothing function removing noise residing in each scan and is trivial to implement. This process greatly enhances the image quality and visibility of the object. The surface reflections still appear in the plots, but aren't overbearing the buried object.

Figure 2.5 displays two additional processing techniques. The first is a varied

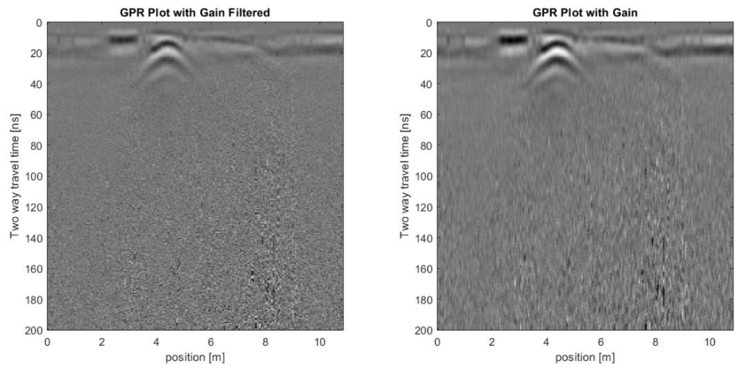

Figure 2.5: Sample Data Gain and Filtering

gain that is applied to both plots. This range of gains accentuates deep object by increasing as the signal begins to attenuate. The values are not high, but a slight increase with depth allows deeper objects to be seen easier. Because of the additional gain, some of the noise is also amplified which calls for a mitigating smoothing filter. This was conducted in the form of a 5 point moving average filter. This final result is displayed on the left side of the figure. Comparing this image with the original GPR plot, it is apparent that the image quality is greatly enhanced and the object is significantly more pronounced. Another important technique that was not implemented here was background removal. This technique removes data consistent between scans, resulting in the object being the only plotted data. All of these are highly beneficial to implement and will be leveraged in future scans.

2.4 Conclusion

This chapter covers the background required to understand later sections in this thesis. Basic concepts involving ground penetrating radar were discussed, along with some of the main concepts of post processing. This lays the foundation not for just ground penetrating applications, but other similar radar techniques such as SAR and ISAR. These details are relatively simple to understand, but the implementation of the technology is significantly more difficult. The remainder of the thesis will be discussing these different technologies along with the design and results of the chosen method.

Chapter 3

Methods

The existence of radar technology has been around long enough for the development of various imaging techniques and processes. It appears that the early stages of this technology did focus on GPR, and currently more advanced techniques are being applied for airborne imaging. This thesis will be focusing on the SFCW chirp, but there is also impulse, SFCW sine wave, and FMCW. Each one of these methods can be used for GPR, but they all have varying degrees of effectiveness and hardware requirements. This chapter will present the theory behind these methods and explain their potential flaws and difficulties, along with the reasoning for developing the chirp SFCW. No simulations will be done in this chapter, just the conceptual reasoning behind each method described above.

3.1 FMCW

FMCW stands for Frequency Modulated Continuous Wave, and is the backbone for any object detection involving motor vehicles. This method most commonly uses high frequency chirp radar as it allows for a high range and velocity resolution. The basis of this concept is that the radar will emit a chirp signal and receive the reflection, then phase offset between the two signals is used to calculate the distance of the object. Generally there is no stepped frequency involved, just wide bandwidth continuous chirps which is where the name "continuous wave" comes from.

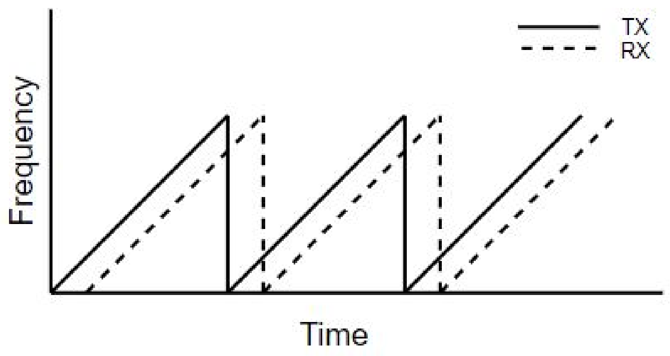

Figure 3.1: Continuous Wave Frequency vs Time

The term frequency modulation generally means a linear frequency increase in the time domain which is commonly known as the chirp signal. The operation in the frequency domain can be seen in figure 3.1 and clearly outlines range ambiguity limitations presented by the continuous wave. If the received chirp arrives after the transmit has reached its peak frequency, no phase correlation can be made resulting in phase ambiguity and distance cannot be measured [6]. This same concept can be applied to calculate the maximum range allowed by the system. This can be represented mathematicall as,

$$R_{max} = c\frac{\triangle f_{max}}{2K} = \triangle f \tag{3.1}$$

where $\triangle f_{max}$ is the chirp bandwidth, c is the speed of light, and K is the chirp rate calculated through the bandwidth and period of the pulse. Since $\triangle f_{max} = KT$, the above equation is equivalent to,

$$R_{max} = c\frac{T}{2} \tag{3.2}$$

suggesting that FMCW is only suitable to short to mid range detection due to its

dependence on the chirp period. It has been found that FMCW has been used in some forms of GPR [13], but using continuous wave limits the effective bandwidth to the radars instantaneous frequency.

3.2 Impulse

Impulse radar systems are used for a variety of imaging applications such as GPR and SAR. For GPR, impulse radar has disadvantages that are easily overcome by the SFCW system, and therefore this thesis will not focus on impulse radar. This method is used by most commercial systems [19] and operates by transmitting a narrow pulse. This signal requires high power and therefore is extremely inefficient and is best paired with a fast processor. This is undesirable in drone SAR applications, which the proposed system has potential to be be used for. The side lobes of the impulse systems are significantly lower than the SFCW waveforms [19] which is advantageous, but excess side lobes can be corrected through filtering and lobe suppression techniques. Impulse radar has low depth penetration and other imaging limitations. Overall the performance through either system can be sufficient in most scenarios, but the SFCW method has enough benefits to make the impulse method undesirable.

3.3 SFCW

Unlike FMCW and impulse radar, SFCW has some extreme advantages with imaging. The main benefit from this method is that effective reconsutrcted bandwidth can be leveraged for high resolution. SFCW theory of operation is as follows, a LFM signal is emitted and reflection data is captured, then the center frequency of the radar is incremented and the process repeats. Reconstructing each received sub-pulse generates the effective bandwidth to be used. Stepped frequency was previously used by the UVM lab with sine wave implementation as GPR. The downside to using the sine wave is that a large amount of pulses are required to construct the same total bandwidth as a chirp. This thesis presents the stepped frequency radar using a linearly modulated signal.

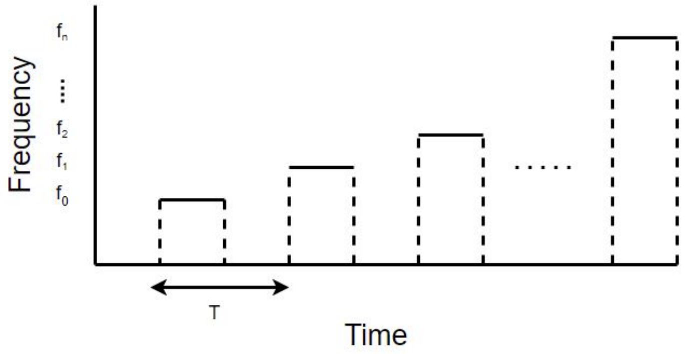

Figure 3.2: SFCW signal in time-frequency plane

Figure 3.2 shows the frequency domain representation of SFCW. Each sub-pulse has its own bandwidth that is then stacked adjacent with subsequent pulses. Provided that the bandwidth of the sub-pulse is given by B_i, and the number of sub-pulses is denoted by N, the effective bandwidth of the reconstructed signal is,

$$B_{eff} = N \cdot B_i \tag{3.3}$$

The advantage of SFCW is immediately seen when incorporating the effective bandwidth into the theoretical range equations of resolution and maximum distance. To begin, range resolution is defined as the smallest distinguishable distance between two objects, and maximum range is the greatest distance at which the radar can detect an object. Resolution is mathematically represented in theory as

$$\triangle r = \frac{c}{2B} \tag{3.4}$$

where B is the bandwidth of the signal and c is the speed of light. Introducing the stepped frequency effective bandwidth augments this equation to be

$$\triangle r = \frac{c}{2N \cdot B_i} = \frac{c}{2B_{eff}} \tag{3.5}$$

This representation highlights that the effective bandwidth has an inverse relationship with the range resolution. In summary, as the effective bandwidth increases, the range resolution decreases in value, therefore improving the imaging quality. Similarly, the maximum range can be represented as,

$$R_{max} = N \cdot \triangle r \tag{3.6}$$

The equation shows that the maximum detectable range can be designed using an SFCW system making it more versatile and is not directly limited to the period of the chirp like FMCW. Therefore it can be concluded that the stepped frequency methods can have range resolution and maximum range characteristics designed simply by altering the effective bandwidth either through increasing sub-pulses or the instantaneous frequency of the signal. There are caveats to these performance increases, but they can be corrected and are addressed in the results section of Chapter 6.

3.4 Conclusion

Through the analysis different methods and desired system specifications, it is clear that SFCW chirp is the most beneficial and the best choice for the thesis. Solely from the fact that any wideband bandwidth can be achieved within specific SDR range, this is the obvious choice. The synthesis of the wideband signal will impose several challenges with compounding difficulty, but the performance increases with lessening system requirements. Other methods have extremely stringent hardware requirements when attempting to emulate wide bandwidth performances of SFCW chirp systems. The following chapters will detail the development of this system and the challenges that have been overcome. The reconstruction proved to be more difficult than appearing on paper, but the results and future projects will be more successful with the LFM SFCW signal.

Chapter 4

Hardware Implementation

4.1 Introduction

Implementing a stepped frequency radar requires the use of complex RF hardware that must to be selected carefully. As with any circuit design, specifications for all components are checked for compatibility. This chapter will detail a number of critical topics conerning the hardware from this project. The first section will elaborate on SDRs, what they are, and why they are leveraged for SFCW systems. Some of the key components in the SDR will be explained as they are relevant to the generation of IQ data transmission, but not the entire device as this is unnecessary. Later the chosen SDR, the Ettus UHD B210 board, will be analyzed in detail including its specifications along with some integral performance simulations. Within this set up, the AD9361 will also be briefly mentioned as it is one of the more common RF chips used for radar applications and is featured in multiple SDRs. This project has used two types of antennas, of which they both will be discussed. This chapter will conclude with a section detailing the troublshooting process with the SDR signal strength.

4.1.1 Software Defined Radio

Software Defined Radio (SDR) can be considered one of the most important technological advancements for the research community in recent times. These devices provide a wide range of benefits, and allow for rapid prototyping in various forms. At the University of Vermont, most of the electrical engineering departments labs use SDRs in one way or another. At its core, software defined radios are devices that are highly modular and can implement a large number of RF components at the cost of one device. Instead of designing hardware with objects such as mixers, amplifiers, and converters, SDRs have the capability to put these components in place via programming. In particular, SDRs are even more useful for GPR applications. Often times when performing GPR scans, it may be desired to omit specific frequency ranges to reduce noise, or to alter frequency steps to save time. This is readily accomplished by SDRs along with any other parameter that needs to be changed. As an example of SDR versatility, the University of Vermont labs have been developing both a sine wave SFCW in addition to the chirp SFCW. These tasks have been completed on the same board. SDRs come with significant complexity and cost, but they can accomplish multiple tasks based on the input code.

4.1.2 Ettus USRP Devices

Ettus Research is a branch of the National Instruments company. They develop and sell various types of SDRs that are commonly used for these applications. For this project the Ettus B210 SDR was chosen for a number of reasons. The specifications for this device can be seen in table 4.1. Most SDRs generally have somewhat similar specifications, but this SDR was chosen particularly for its wide instantaneous bandwidth. The B210 has a bandwidth ranging from $70MHz$ all the way to $6GHz$. This means that with frequency tuning, a signal can be reconstructed leveraging this entire bandwidth as long as the hardware permits.

Instantaneous frequency of a radar is the maximum bandwidth that the device can achieve at any single moment. For the B210, given the instantaneous frequency of $56MHz$, the sub-pulse can only generate a signal ranging from 0-56MHz. The instantaneous frequency is shared by all channels of the device, meaning all transmit and receive channels must share this value. In the configuration of this device the full bandwidth is not put toward a single sub-pulse since both channels are used so the SDR can provide a phase coherent reference signal.

Specification	Value
RF Coverage	70 MHz to 6 GHz
Architecture	2x2 MIMO
Operation	Full Duplex
RX Gain Control	0 to 76 dB
TX Gain Control	0 to 89 dB

Table 4.1: B210 Main Specifications

In addition to the wide bandwidth, the radar is a $2x2$ full duplex system. Referring to a radar as full duplex means that the transmit and receive operations can be conducted at the same time which is important when synchronizing channels. In combination with the full duplex capabilities of the radar, the stepped frequency routine can be designed using a stop and go technique. An in depth analysis of this technique and how the system operates can be found in Chapter 5 in the Software Implementation section. Ensuring phase synchronization is achieved by using one channel as the reference signal and is also detailed in Chapter 5.

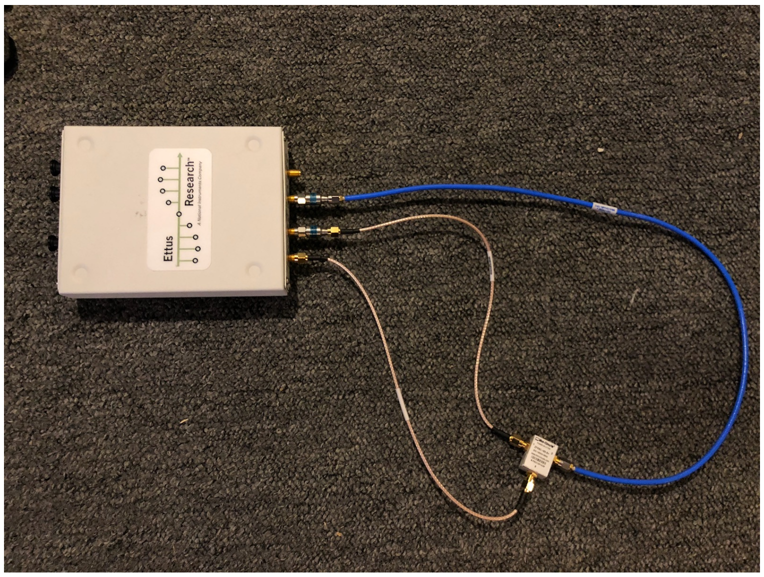

Figure 4.1: Loopback Configuration

A basic loopback configuration can be seen in 4.1. This is used to generate calibration data and shows the configuration of the device. One cable runs from the transmit channel into a splitter, where one output returns directly to the device to be used as a reference, and the other travels out to collect the response data. In a loopback test scenario, both cables are configured similarly, but a longer cable is used to emulate a time delay. In actual testing the second line feeds into the antenna to collect data.

4.1.3 SDR for Signal Generation

As mentioned in the previous sections, SDRs can be configured to generate a wide range of signals for transmission. These can have variable magnitudes, center frequencies, and patterns. Specifically for the chirp signal, there are a few key hardware components that the SDR has to internally configure to generate the chirp. This section will detail the LFM generation from the SDR perspective.

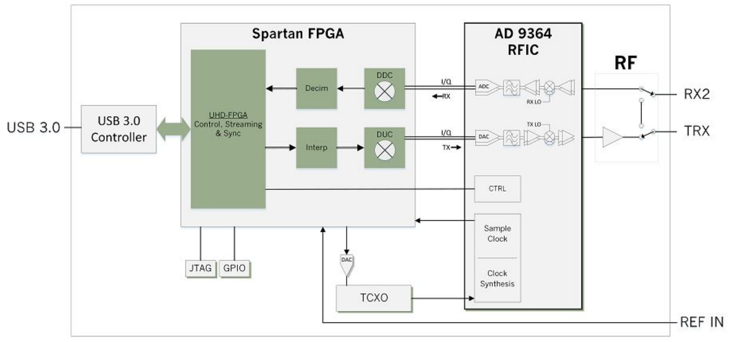

Figure 4.2: B200 Series Block Diagram

The block diagram in figure 4.2 shows the basic configuration of the B200 series SDRs. The SDR used for this project is the B210 which is a $2x2$ system as where the B200 is just a single transmit and receive. There is no block diagram provided by Ettus since it would be the same as the B200 with double the components. There are three main blocks of this SDR that allow the user to operate the device. The first thing to notice from the diagram is that there is USB 3.0 capability that allows for data transfer speeds up to $5\frac{GB}{s}$. This transfer rate is ten times the performance compared with USB 2.0. The other components of the SDR include the Spartan FPGA and the AD9361 transceiver chip.

The AD9361 is the chip that equips the SDR to have all of the high performance RF specifications. This chip is commonly used in other SDRs, and was also found in the AD-FMCOMMS5 board used at the beginning of this project. Specifications can be seen in table 4.1 since they are the same as the SDR. The most important quality in this section would be the total RF coverage allowing the SDR to operate over the $6GHz$ range with stepped frequency. When looking at SDRs that use this chip, often times the FPGA is the performance bottleneck and the full range cannot be realized. The user interface of the SDR is extremely important when it comes to accelerating the project. The AD-FMCOMMS5 had significant shortcomings with the user interface. The board was full duplex, but to achieve this capability HDL code was required to be written for not only the operation of the transceiver, but also for the interface between the AD-FMCOMMS5 and the Xilinx FPGA. Fortunately, Ettus Research has developed the UHD interface for all of the SDRs. This is commonly used with GNU radio, C applications, MATLAB, and finally Python. For these reasons the B210 Board was chosen to implement the SFCW GPR.

4.1.4 Additional Hardware

Antennas

Another key component to this SDR system is having proper antennas that can effectivly accomplish the frequency tuning and wideband performance. Although there are several types of antennas, for this project the testing was completed using the set of Log Periodic antennas seen in figure 4.3. Log periodic antennas generally provide a wide range of frequencies usually ranging from $30MHz$ to $6GHz$ and above. The set in use has a frequency range of $850MHz$ to $6500MHz$ which is ideal for the GPR application. This frequency range is close to the full range of the SDR itself. These antennas get their name from having their impedance be logarithmically periodic as a function of frequency. A general antenna pattern can be seen in figure 4.4 The manufacturer did not provide specific antenna patterns so this can be used just as a reference. This pattern is sufficient for GPR applications as it provides enough directed signal strength and will not impose too much noise to the neighboring antenna

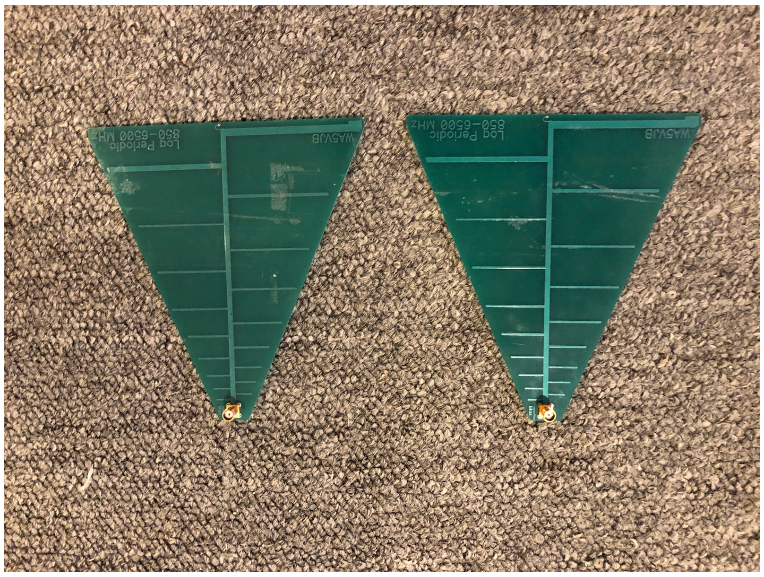

Figure 4.3: Log Periodic Antennas

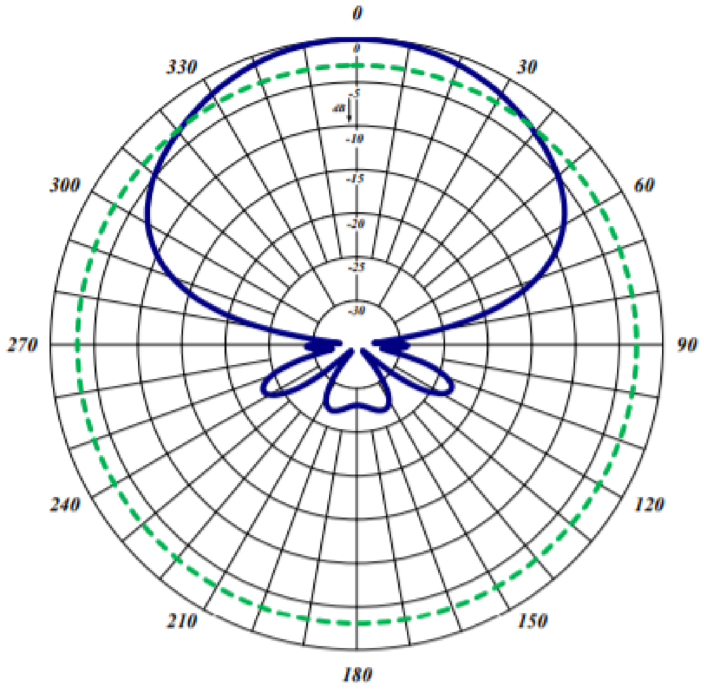

Figure 4.4: Log Periodic Antenna Pattern [31]

in the GPR system. During testing, foam is usually placed between the transmit and receive antennas to reduce the direct coupling, but because of the COVID-19 situation, working from home did not permit the use of foam isolation. To account for this, the antennas were placed slightly farther apart than normal, but the direct coupling is still apparent.

Another consequence of the COVID-19 pandemic was the lack of lab access to use the custom made antennas. For previous GPR testing, the UVM lab used custom made antennas by the Mechanical Engineering department that were mounted on the test bed. As there was no access to the lab, these high quality antennas were not accessible.

Attenuators, Splitter, and Cables

The remainder of the hardware consisted of attenuators, splitters, and cables. Ettus Research notes that a 20dB attenuator is required for any direct loopback. With this being said, two $20dB$ attenuators were used. The splitter is a Mini-Circuits DC-$18000MHz$ power device. The cables were also from Mini-Circuits and were rating from DC to $18GHz$. After a number of issues it was found that the magnitude response of the system dropped significantly at specific frequencies. A solution was produced via software and is explained in later sections.

4.2 Hardware Testing

Before writing the algorithms and beginning work on chirp SFCW, a PhD student by the name of Yan Zhang generated the code to have the B210 operate as a sine wave

SFCW. This code was used to detect objects with a method similar to FMCW GPR. With this data a few things were measured to ensure all of the components operated correctly.

The magnitude response of the radar was calculated over the entire frequency range of the equipment. This ranged from $800MHz$ to $6GHz$ and the results can be seen in Chapter 6 with the frequency spectrum being plotted centered at zero. The radar itself has a a gain versus frequency response for both transmit and receive, showing a drop no more than $15dB$ throughout the entire bandwidth of the SDR. Before constructing a radargram, each A scan must be shifted by a calibration response. This accounts for the length of the cables used, allowing the calculations to show the true measured distance of the object.

4.3 Gain Correction

While performing loopback tests, the results appeared to be accurate within the designed resolution, but when observing the stacked spectrum, it appeared that the signal strength was varying by upwards of $100dB$ throughout the frequency range. At the time, it was believed that this was a hardware problem and maybe an issue with the transfer function of the splitter or other hardware in use. After a significant amount of troubleshooting, it was realized that the channels also had different gain levels when all parameters were matching. Channel B was about 20dB weaker than Channel A. During loopback matching attenuators were used to help troubleshoot this and the device appeared to be working better. The magnitude of the received time domain signals seemed to hover around values of 0.5mV to 0.9mV and decreasing as the frequency increased. When plotting the spectrum it was obvious that the magnitude was unstable, and without saturating the signal it was difficult and tedious to correct the gains. The overall wideband frequency range was separated into ten sections that all had various gains attempting to correct the problem. Once implemented, the results seem to remain the same as they were before, but now the spectrum was only dropping by 30 to 50dB. To solve this persisting issue, a look-up

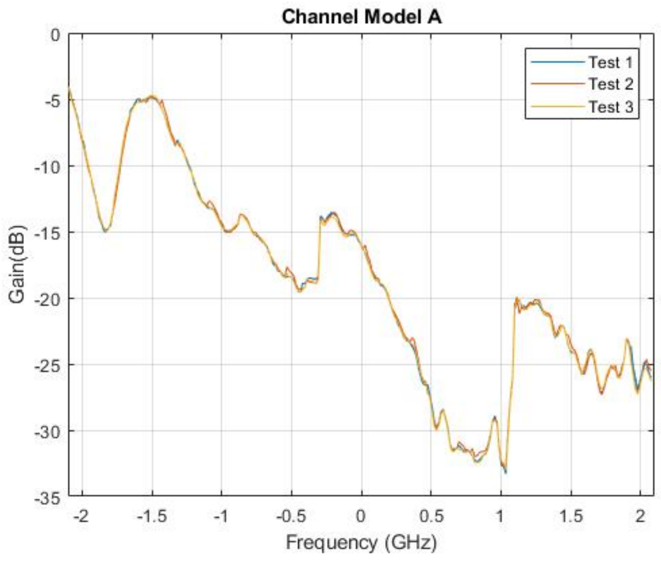

Figure 4.5: Channel A Transfer Function, Center Frequency 800MHz-5GHz

table was generated because sectioning the gains was not sufficient. The idea was to created a table that the python code would reference at each center frequency. The values inside this table would be the correct gain to provide a magnitude normalized to the same value. Through experimental testing it was concluded that the transfer function of the hardware was deterministic, so the look-up table could be generated based on loopback data. Because of this, the first step in constructing this table was to perform a loopback test with the radar and then calculate the gain required.

Figure 4.5 shows the drop in gain over the entire frequency range of 800MHz-5GHz centered about zero. After some time using the radar, the gain changes appeared to happen at the same frequencies. To quantify the changes and prove that the transfer function was in fact deterministic, figure 4.5 was produced for multiple trials. Three are shown in the figure, but after a significant amount of use it is safe to say the transfer function can be corrected with a gain table. As previously noted, the two channels received signals at different strengths so the process was repeated for the second channel. The results from performing the same procedure on the second channel can be seen in figure 4.6.

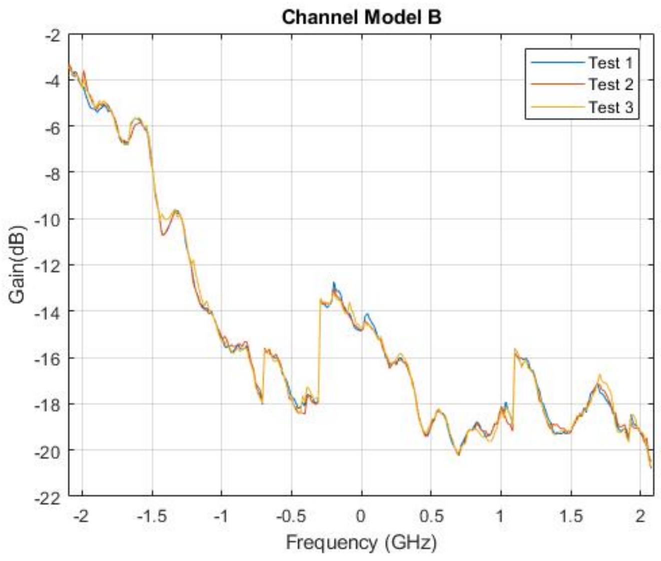

Figure 4.6: Channel B Transfer Function, Center Frequency 800MHz-5GHz

Comparing the two channel models illustrates that the channels are not off by a set value and a gain table is required for each channel. These values were stored in a matrix, one column for channel A and the other for channel B. The function to assign receive gains was implemented at each frequency step calling this look-up table and pulling the correct value. Plotting the necessary gains to correct the magnitude can be seen in figure 4.7.

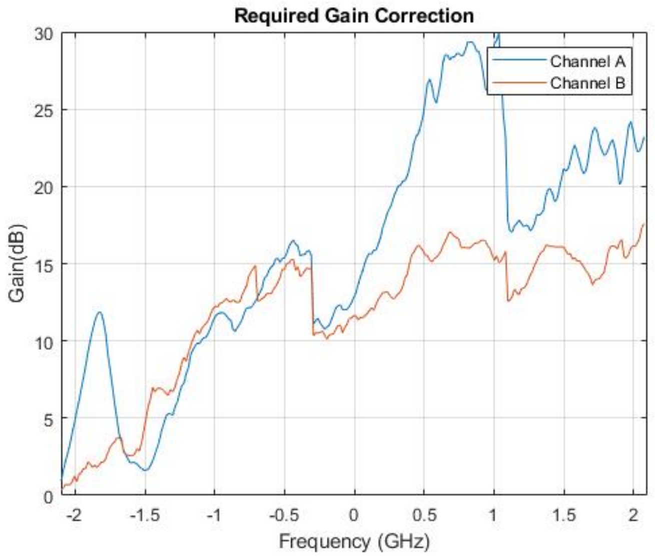

Figure 4.7: Required Gain for Signal Correction, Center Frequency 800MHz-5GHz

Prior to implementation of the gain control, the A scan produced by the uncorrected data still measures the length of the cable correctly, but will not have the designed resolution. Additionally if the gain was not high enough on the radar, the receive buffer would detect only zeros and the general operation of the device is not always consistent. Pulling the values from figure 4.7 and including them in the code as a look-up table produces a consistently working system. The corrected channel model can be seen in figure 6.4.

Using these values allows for the reconstruction process to leverage the full effective bandwidth. The correction of this transfer function actually introduces more side lobes which is expected. Before the low power sub-pulses did not contribute to the resolution, so now every single pulse is taken into account and the radar achieves the designed effective bandwidth. The additional contribution is also the explanation for the side lobes since they originate from the spectral discontinuities.

4.4 Conclusion

Selecting the hardware was a relatively difficult task. Choosing things such as the cables and antennas was simple, but selecting the proper SDR required a strong knowledge base around SDRs and exactly what the capabilities would be. To begin the research project, ordering the AD-FMCOMMS5 seemed like the most obvious option. Analog Devices provided and SDR that utilized the AD9361 chirp so the device could operate in the full range up to $6GHz$. It also provided multiple platforms for operation, including windows support, which is not often given with SDRs. The lack of synchronization between transmit and receive prevented the stop and go method of transmission. This board will function for FMCW applications or for sine wave SFCW, but without HDL coding it could not be used for this project. This was a frustrating portion of the project as several months were spent conducting research and setting up the board.

Switching to the B210 board in March of 2020 was a dramatically better process. The company provided proper documentation and an API that could operate without any bugs. The downside to this switch was learning python from scratch, but this was still better than using the AD-FMCOMMS5. Although there were several issues, the final hardware set up is fully capable of performing the task at hand.

Chapter 5

Signal Processing and Software Implementation

5.1 Introduction

Following the correct implementation of the stepped radar via the SDR, the signal transmission can be represented mathematically. The following equations were used not only to understand, but to first simulate and finally develop the SFCW system. This section will present a detailed analysis of the channel model and the software development techniques applied to obtain the functioning system. The first section will describe the matched filer and its relevance, the proceeding sections will cover the signal modeling and software implementation of the SDR.

5.2 Matched Filter

As discussed in Chapter 3 there are several methods available for object detection and avenues for constructing a GPR system. For stepped frequency systems, the design frequently accomplished through processes using a matched filter resulting in a range profile. There are several reasons for using this filter, with the most critical being that the signal to noise ratio is excellent. Additionally it is well known that the matched filter is the optimal linear filter for range detection as the filter is provided a known reference signal [28].

For reference, the matched filter is often referred to as pulse compression [7] when talking about LFM signal for range detection. This is simply because the period of the chirp is directly related to the bandwidth.

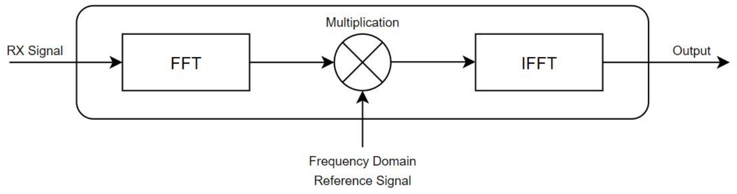

Figure 5.1: Matched Filter Process

The block diagram in figure 5.1 represents the basic theory of the matched filter. There are two signals involved, one being the received signal and the other is the reference signal. The reference is a replica of the expected receive signal, allowing for return signals with low power to be realized. The signal to noise ratio of the match filter can be defined and shown as

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{5.1}$$

where P_{signal} and P_{noise} respectively are the signal power and noise power.

Referencing the derivations below and equation 5.1 it is recognized that the SNR of the matched filter is frequency independent which is ideal for the wideband synthesis process. The matched filter seen in Figure 5.1 can be written in the time domain as,

$$y(t) = x(t) * h(t) \tag{5.2}$$

where $x(t)$ is the received signal while $h(t)$ is the reference. The impulse response of the match filter provides insight into how the resulting signal is characterized by a sinc shape through a derivation of taking the Fourier Transform. Beginning with the impulse response of the matched filter from [7],

$$h(t) = \mathcal{F}[H(f)] = [\int_{-\infty}^{\infty} S_{in}(f) e^{j2\pi f(t-t_0)} dt]^* \tag{5.3}$$

then simplifies to,

$$h(t) = S_{in}^*(-t) \tag{5.4}$$

Performing the match filter operation of the impulse response, 5.4, with equation 5.2 allows for the output to be computed from

$$S_{out}(t) = \int_{-\infty}^{\infty} s(t-\tau) h(\tau) d\tau \tag{5.5}$$

as,

$$S_{out}(t) = e^{j2\pi(f_c(t-t_0)+\frac{K}{2}(t-t_0)^2} \int_{-\frac{T1}{2}}^{\frac{T1}{2}} e^{-j2\pi[K(t-t_0)\tau]} d\tau \tag{5.6}$$

and after further simplification,

$$S_{out}(t) = T_1 e^{j2\pi(f_c(t-t_0)+\frac{K}{2}(t-t_0)^2)} \cdot sinc[KT_1(t-t_0))] \tag{5.7}$$

In this form it can finally be seen that the response of pulse compression takes the form of the sinc function as seen in figure 5.2. From the t_0 term in 5.7, it is apparent that the delay will shift the peak of the output at the distance the target is located. Performing the match filter operation on one signal with itself simulates having a transmit and receive signal with an ideal SNR and a t_0 of zero.

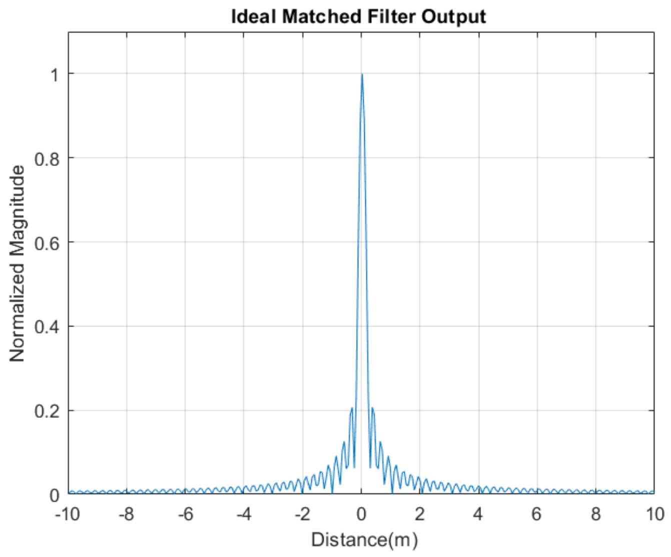

Figure 5.2: Ideal Match Filter Output

The typical response is shown in figure 5.2. From observation, it can be realized that the width of the main lobe also has a relationship with the bandwidth of the signal and this is proven mathematically. The $4dB$ width of the lobe occurs at the inverse of the signal bandwidth, $\frac{1}{B}$ [7]. Similarly, the troughs of this lobe are found at $\pm(t_0 + \frac{1}{B})$. From here is it easily seen that as you increase the bandwidth, the resolution of the matched filter increases.

This behavior relates to the range resolution of the radar which is $r = \triangle t \cdot \frac{c}{2}$. Supposing there are two scatters detected by this system, there will be two main peaks from the matched filter and if they are spaced apart less than the resolution, they will be indistinguishable since the lobe widths will be too wide. Increasing the bandwidth will allow objects to be distinguishable from one another at decreasing distances. Given that $\triangle t = \frac{1}{B}$, the range resolution can be simplified as in 5.8.

$$\triangle r = \frac{c}{2B} \tag{5.8}$$

5.3 Signal Modeling

For previously mentioned reasons, the chirp will be the transmitted signal of choice. For this project, the chirp signal is created via the Python and pushed to the transmit buffers of the B210 board. At this point the signal is in discrete time and converted to an analog signal through the B210's DAC's. The SDR modulates the chirp and transmits the signal at the specified center frequency. The signal is reflected back to the board and converted into digital form through the ADC and can be stored on a Host PC for processing.

In its digital form, the chirp can be expressed as,

$$w[t_m] = A[t_m]e^{j\pi K t_m^2} \tag{5.9}$$

where square brackets denote discrete time. The amplitude of this signal is represented by $A[t_m]$ and the chirp rate variable by K. Given specific parameters, the chirp rate is an important variable that controls the length of the signal along with the frequency range starting from zero. This variable is controlled by the input sub-pulse bandwidth, B_i, and period, T_p, and is defined in 5.10

$$K = \frac{B_i}{T_p} \tag{5.10}$$

The notation used in the signal representations coincide with those found in [1] as the following algorithm can also be found in this reference. Converting the chirp signal to its analog form is conducted using the Whittaker-Shannon formula, or more commonly referenced as sinc interpolation. This is expressed as,

$$w(t) = \sum_{m=-\infty}^{\infty} w[t_m] \frac{sin(\pi f_s(t - t_m))}{\pi f_s(t - t_m)} \tag{5.11}$$

Based on the Ettus B210 architecture, there is a non-coherent relationship between the TX and RX LO phases, meaning that without taking special care to ensure synchronization, there is no phase relationship between the channels. With this being said, the phase errors for each sub-pulse, n, will be represented and accounted for in the algorithm seen in 5.12 as the difference between transmit and receive phase.

$$\phi_{e,n} = \phi_{t,n} - \phi_{r,n} \tag{5.12}$$

Following the digitization of the received chirp, and considering phase errors and time delay, the signal after demodulation is represented as,

$$z_n[t_m] = w[t_m - \frac{2R_s}{c}] e^{-j2\pi f_n \frac{2R_s}{c}} e^{j\phi_{e,n}} \tag{5.13}$$

Inherently these errors need to be corrected which can be accomplished in 5.14 from [1] or from using the secondary channel for phase-coherent loopback. This equation was constructed through various filters detailed in [5] and performs sinc interpolation, frequency, phase, and time shifting. This filter must be applied to the return signal before performing any operations, otherwise these errors will corrupt the signal deeming it useless. This filter is shown in equation 5.14.

$$g_n[t_m] = \frac{1}{2\pi\triangle f_c}\frac{sin[\pi B_s(t_m - \triangle T_n)]}{\pi(t_m - \triangle T_n)} e^{j\pi\triangle T_n \triangle f_n} e^{j2\pi(t_m - \triangle T_n)\triangle f_n} \tag{5.14}$$

At this point in the process, the frequency stacking algorithm proposed in [1] is performed. The basis behind this is that the matched filter operation will be applied to each individual sub-pulse prior to reconstruction. The frequencty response of each sub-pulse will be stacked adjacent to each other for N pulses resulting in the effective bandiwdth of $N \cdot B_i$. Taking the inverse Fourier Transform will allow for utilization of the reconstructed wideband signal. Represented in the time domain the process is as follows:

The aforementioned filter 5.14 will be applied to each frequency shifted sub-pulse via the conjugate operator and summed over the N frequency steps as in 5.15 to obtain the synthetic wideband signal.

$$z[t_m] = \sum_{n=0}^{N-1} (z_n[t_m] e^{j2\pi\triangle f_n t_m} e^{-j\phi_{e,n}}) \circledast g_n[t_m] \tag{5.15}$$

This signal is the corrected version of the receive signal and will be used in the matched filter process. The filter will use a similarly constructed synthetic wideband reference expressed as,

$$v_n[t_m] = w[t_m]e^{j\phi_{e,n}} \tag{5.16}$$

Depending on the situation the reference signal is often ideal, but a loopback configuration can be used as a phase coherent reference. The output of the matched filter shown in 5.17, is the pulse compressed wideband signal which is the proper signal to be used in radar range detection.

$$d[t_m] = z[t_m] \circledast v^*[-t_m] \tag{5.17}$$

Following this process with sufficient lobe suppression allows for equivalent performance to that of sending a full wideband chirp. Since frequency stacking is required, it is seen that the reconstruction process contributes some issues and additional filter is required. As this proceedure is conducted on each sub-pulse, N, the sub-pulse prepresentation is,

$$d_n[t_m] = z_n[t_m] \circledast v_n[t_m] \tag{5.18}$$

and accordingly in the frequency domain,

$$D_n[f_k] = Z_n[f_k] \cdot V_n[f_k] \tag{5.19}$$

There are multiple benefits of this method, but one of the main advantages comes with sampling. When reconstructing a wideband signal one sub-pulse at a time, the processing requirement placed on the SDR is based off of the individual sub-pulse. This means that if following the Shannon-Nyquist rule, the required sampling rate of the SDR is equivalent in magnitude to the bandwidth of the sub-pulses. Without the individual pulse compression it would be necessary to perform the compression on a massively upsampled signal, placing an exorbitant burden on the signal processing. [19]. In most modern day programming platforms, parallel processing can be performed on each sub-pulse drastically increasing the algorithm efficiency and allowing for real time processing. Given the mathematical representations of the signals in

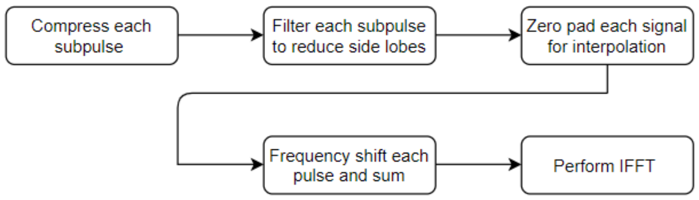
Compress each subpulse
Filter each subpulse to reduce side lobes
Zero pad each signal for interpolation
Frequency shift each pulse and sum
Perform IFFT

Figure 5.3: Frequency Stacking Algorithm

each step, a frequency stacking algorithm [1] can be employed to generate the reconstructed signal. This is the synthetic wideband waveform that when used can create the radar range profile, or the A Scan as mentioned before. This result is useful for B scan construction along with any general radar applications. Below is a concluding summerization of the frequency stacking algorithm and a simplified version can be referenced in figure 5.3.

1: For each sub-pulse signal, take the discrete Fourier Transform to compute the compressed signal 5.19.

2: Filter each sub-pulse with a desired window function. The bandwidth of this filter should be equivalent to the frequency spacing of the sub-pulses.

3: Zero pad each sub-pulse at both ends. To sufficiently upsample the signal pad the front with $L_{up} = N \cdot L$, where $L = f_s \cdot L$

4: Perform a circular shift in the frequency domain by $\triangle f_n$.

5: Sum each of the compressed sub-pulses to obtain the frequency domain response

$$D[f_k] = \sum_{n=0}^{N-1} D_n[f_k - \triangle f_n] rect[\frac{f_k - \triangle f_n}{B_s}] \tag{5.20}$$

6: Perform the inverse Fourier Transform to obtain the full response

$$d[t_m] = \frac{1}{L_{up}} \sum_{k=0}^{L_{up}-1} D_n[f_k] e^{j2\pi km/L_{up}} \tag{5.21}$$

The above algorithm was used to properly reconstruct the synthetic wideband signal. For this project each subpulse was passed through a Turkey window in MATLAB based on its suppression of distance lobes [1]. Given that the sent pulses each have a bandwidth of $16MHz$, the width of the filter covers only these frequencies. The zero padding was peforming on each sub-pulse while being stored into a pre-allocated array. The front end of the pulses was provided $\frac{L_{up}}{2}$ zeros, while the remaining pulses were provided enough zeros to allocate room for the remaining pulses with $\frac{L_{up}}{2}$ foll-wing. In the frequency domain, the concatenation of zeros before and after the signal effectively performs sinc interpolation in the time domain. Generally this upsampling is by a factor of n [4]. Although this introduces a slight error in the reconstructed signal, this has been deemed insignificant in terms of signal resolution [5]. The remaining portion of the algorithm was performed exactly as stated and the inverse Fourier transform operated on the reconstructed signal. This process was simulated in full to ensure proper functionality and can be seen in Chapter 6.

5.4 Software Implementation

5.4.1 Overview

Following the mathematical representation as previously described, the radar signal can be implemented with a variety of programs. Since the B210 board is a common SDR, GNURadio is a popular option along with MATLAB and Simulink. Given the task at hand, any of these are viable solutions, but some pose limitations. Currently the laptop in use in a windows based computer, but setting up the radar and all of the proper kernels is best completed using the Linux operating system. The access provided by Linux to the computer structure is significantly less problematic than windows, so for radar operation an external boot drive with all of the Linux based dependencies was created. GNU radio was installed with its most recent Master, and it was found that UHD has their own blocks for the radar. These would have automatically configured time synchronized transmit and receive signals, but custom blocks would have been required to perform the frequency step. With this being said, Python was the program of choice to allow for full customization. Ettus Research has developed a Python API to control all of their SDRs and provided basic examples to accelerate progress.

The chirp signal was designed using equation 5.9, using a bandwidth of $16MHz$ distributing these values between the positive and negative frequency spectrum. The overall instantaneous bandwidth of the SDR is $56MHz$, but this is a maximum value for all channels. With this being said, phase synchronization of the sent and receive chirps was accomplished by using a second receive channel that consequently shared

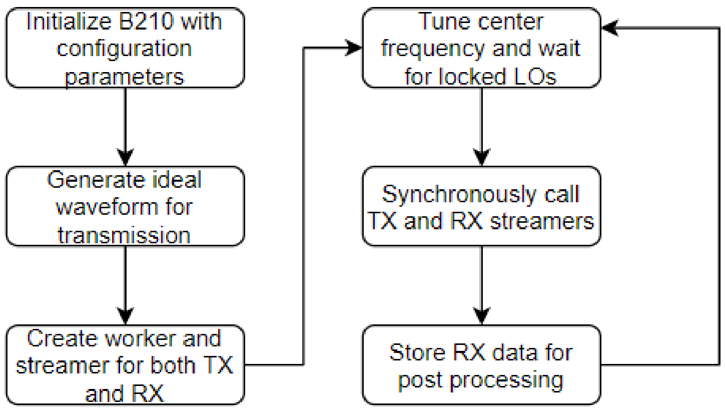
Initialize B210 with configuration parameters
Generate ideal waveform for transmission
Create worker and streamer for both TX and RX
Tune center frequency and wait for locked LOs
Synchronously call TX and RX streamers
Store RX data for post processing

Figure 5.4: Data Transfer Routine

the instantaneous bandwidth. Also the received signal was slightly oversampled as a best practice. Unfortunately this design was not able to reach the full capacity of the instantaneous frequency. The laptop being used would begin to fail when sampling greater than $16MSPS$ which is why this bandwidth was chosen.

The subsequent sections will detail the process seen in the block diagram of figure 5.4. First the construction of the transmit and receiver workers will be discussed along with the streamers. Later the center frequency tuning will be discussed as this is one of the most essential components to a SFCW system. There are no simulations for this section as most of this was accomplished through reading and understanding the documentation. USRP provides a user manual explaining some basic functions, but often sufficient descriptions were omitted. Some basic examples were provided, but were not used as they did not accomplish anything worth using.

5.4.2 Transceiving a Specified Number of Samples

Transmit Streamer

Operating the radar via python API allows for complete control over the device. With this being said, there is no direct and obvious way to transmit and receive information aside from some basic examples. To perform the frequency stacking algorithm and being able to properly correlated transmit and receive signals, the device must send a designated amount of samples and at the same time, receive another specified number of samples. Crafting the streamer presents the SDR with the information necessary to correctly operate. This mostly involves the manner in which the transmission is conducted and this will be referred to as "stop and go" transmission.

Transmitting a signal on the device is done via buffers containing samples described by the metadata. The metadata is additional information along with the signal being sent that contains the buffers content. Required to transmit a signal, the configuration of not only the transmit worker, but the streamer is necessary for proper operation. Constructing the streamer defines how the signal is sent to the host PC in regards with the internal DSP. The manner in which the streamer is crafted depends on a specified number of samples, a "stream now bool", the stream mode, and finally a timing specification provided by the metadata. While the number of samples is self explanatory, the streamer modes consist of begin and end continuous, and two other settings for contiguous streaming. Since each transmission is stored in a buffer, if the designed signal exceedes the size of the buffer then contiguous modes will be required to not lose any information. The time specification will be described with the transmit worker. For the remaining inputs, they are self explanatory bool functions.

Transmit Worker

The generation of the transmit worker contains more specific information regarding timing specifications and the actual signal being processed. The transmit worker uses the "send()" command to specify the buffer being sent along with the number of samples to send per buffer, metadata, and the time to wait per packet. The function returns the number of samples sent, meaning that in a simple while loop the device can be designed to send a designated number of samples in total. The most important aspect to the transmit worker is incorporating the proper pulse per second (PPS) value discussed in the next section. The timing information is held within the metadata therefore constructing the metadata outside of the transmit worker does not allow for true synchronization. The metadata is constructed via calling the metadata function, but there is an additional input for synchronization. By default, through the python API the transmitters and receivers are not synchronized so with each call to the metadata the current time must be pulled from the system clock. This PPS information when called will hold its value until the next clock rising edge, then reset its timer to zero seconds. Both receive and transmit buffers depend on this value, and the timing reset synchronizes the two.

Threading and Receive Worker

In the design of the worker, the metadata is called to align the transmit point as time zero. Performing a similar operation with the receiver in conjunction with threading successfully synchronizes the two channels. Without this action there would be no way to correlate transmit and receive using matched filter implementation. Once the transmit worker is created, the receiver worker is a simple task, but ensuring that these functions are synchronized themselves is challenging. Currently most processing softwares offer an option for parallel processing and this is possible using Python 3.0. Threading is a form of parallel processing used in the Python language that allows a user to operate two functions at the exact same time. This command provides the capability of constructing a set of functions and aligning them in time before passing the command to begin operation. Once this is done all of the functions will operate in parallel using the computer's multi-core processors and will join together when called. As previously stated this is in addition to sending time specified data and completes the phase synchronization.

5.4.3 Center Frequency Tuning

The previous section outlines the procedure followed to conduct the signal transmission. To implement the stepped frequency radar, after creating proper sub-pulse transmission, the next step is to construct an algorithm for stepped the center frequency of the transmission between pulses. As this is a common application of UHD SDR's, the API provides functions allowing for center frequency tuning. This stepping technique is the reason the signal transmission has to be implemented in the "stop and go" fashion, where another pulse will not be transmitted until the sent signal is received. From previous sections, the frequency step is denoted as $\triangle f$ and for the testing was a value of $16MHz$.

To ensure the SDR is correctly operating, the tuning must be completed before moving on. This was implemented in a secondary function and was declared as a python state. The USRP device has a set of sensors capable of detecting if the transmit and receiver local oscillators are "locked". In this scenario, locked means that the user specified center frequency is set and the device is ready for operation. Placing this function along with the transmit and receiver workers as states, the python code was constructed to run as depicted in the block diagram in figure 5.4. For the AD9361 devices, absolute timing synchronization, if implemented correctly, is guaranteed with sample-level precision. For operations such as local oscillator tuning this is not the case, but the delay will be constant between frequency steps. This does not pose any issues in this project and simple means the total completion time of a scan is not constant. The frequency spacing was included in a separate radar parameters file, allowing for the user to easily change the step value without reading through the main code.

5.4.4 Processing Improvements

The total time from running the python code to developing a B scan image has room for improvement. Currently there are several methods in place to decrease the run time, but a single scan still takes 10 seconds. For this project the SDR is controlled and completes testing via a 2018 Lenovo Flex laptop. The specifications of this device include a baseline i5 processor. Although functional, this laptop is by no means a processing powerhouse and therefore the data is stored and moved to a custom built desktop PC. The PC has and overclocked Ryzen 1600 processor with a GTX 1060 graphics card. Similarly to the laptop, these specifications are not impressive, but provide significant improvements over the other device. The code used to generate the A scans takes the most amount of time because of the frequency stacking algorithm. To achieve the wideband signal, each sub pulse of 270000 samples is stacked adjacent with the others. This means that the total vector is $270000 \cdot N$ samples long and will be used in every itteration making the code inefficient.

To increase the A scan processing time the parallel toolbox in MATLAB was used. This toolbox provides a set of functions that can designate lines of code to separate cores of the processor and running them in parallel. After several attempts, the desktop does not seem to be able to handle running the loop in parallel and takes significantly longer. As an alternative GPU processing was implemented and shows a drastic improvement. The parallel computing toolbox allows for matrices to be made and stored in the GPU instead of the default CPU. Since the CPU is almost always the bottleneck in these types of operations, using the GPU results in a 200 percent time decrease. The real benefit of GPU processing comes into play when operations such as the DFT and IDFT are called on large matrices, so this is a good use in the frequency stacking algorithm. Before running the loop, the matrices are all passed to the GPU and then the algorithm is continued as usual. The part of the code that requires so much time is the large pre-allocated matrix that all of the frequency domain data is stored into. There is room for further optimization, but a better processing unit would be the best solution. It is fortunate that using this PC was available because of the performance increase.

5.5 Conclusion

Overall this chapter was one of the most important in terms of completing the research project. Although there is not a large amount of technical information such as equations and simulations, there was a significant amount of work put into the signal creation. Developing the frequency stacking algorithm and testing did in fact have simulations, but setting up the device itself took a significant portion of time and there was much to learn. Learning python from scratch was a challenge, but fortunately python is not an extremely difficult language. This section illustrates how powerful this language can be. This chapter also outlined the construction of the TX and RX protocols and how their routines were scheduled. Creating a synchronous system is highly important and is not a simple task.

The signal modeling portion of this chapter is fundamental to creating simulations prior to implementing on the radar. All of the equations provided were used directly in simulations. These will be discussed in Chapter 6. In the timeline of the project, once the tasks presented in this chapter were completed, it was time to begin testing the overall system.

Chapter 6

Simulations and Results

This is the final chapter of this thesis and it will display the current results with simulations conducted throughout the project. The simulations were performed at each step of the research to ensure proper functionality before proceeding. There was additional testing completed at the beginning of the project for the AD-FMCOMMS5 but this data will be omitted as it is not valuable to the thesis. Testing conducted for the transmit and receive workers will also not be included as the evidence can be seen in the properly functioning radar. Before simulating the full SFCW system, a matched filter response of one send and receive chirp will be produced. Following this a full-scale SFCW will be conducted and the grating lobe suppression will be analyzed. The later sections will contain the actual results and detail the troubles in the process. The primary and secondary testing set up will be presented with loopback verification and Bscan testing. The final section of this chapter will contain final words regarding the process, the troubles endured, future work, and a conclusion of the entire project.

6.1 Simulations

6.1.1 Individual Signal Operation

The most difficult component of setting up the radar involves transmitting one chirp and synchronously receiving. Before attempting to generate any type of chirp with the radar, simulations were done in full to limit the possibility of problems. There are several ways to generate a chirp signal and there also are built in functions in various programs. For both MATLAB and Python, the built in functions generate a real signal that will be passed to the SDR. This is problematic as it will not contain any phase information, therefore for all simulations and for the SDR, equation 5.9 is used. A separate function was created in both MATLAB and Python taking inputs of instantaneous bandwidth, a time vector, and a number of samples. With this the chirp rate is calculated and passed into equation 5.9. Negative time indices are used to leverage the benefits of the complex chirp. Twice the bandwidth is available with the complex chirp without massively upsampling.

Using MATLAB as the simulation software of choice, the matched filter was simulated using two ideal signals. The first will represent the reference signal, as it is an ideal chirp with no noise. The second signal will emulate a received chirp from the radar. This was created by generating an ideal chirp, delaying the signal via MATLABS circle shift function, and finally introducing noise.

Representing the delayed chirp signal can be done in multiple ways. The delay in the chirp can be written in the time domain as $\frac{2R}{c}$ where R is the range of the target. Using this, the delayed signal can be represented mathematically in 6.1 as,

$$w_r = w[t - \frac{2R}{c}] = e^{j2\pi K(t-\frac{2R}{c})^2} \tag{6.1}$$

In simulation, this time delay is directly related a specific number of samples, and therefore performing a circular shift effectively simulates a received signal. Performing the matched filter operation on the two signals produces the simulated sinc output. At this point the transmission of a single signal produces the correct output and the next step is to implement the stepped frequency.

6.1.2 Stepped Frequency Continuation

This section will detail the simulation process presented in the frequency stacking algorithm. Compared to the ideal chirp this reconstruction will have spectral discontinuities resulting in non-ideal behaviors. The expected output of the reconstruction will be a typical sinc response, but with some possible side lobes from the repetitive stacking. It is well known and documented that the addition of stacked spectra introduces grating lobes [1]. Reducing these will be addressed in later sections. Although this simulation will introduce some lobes, their effect should be minimal, but in practice the grating lobes will become more prominent.

To achieve ideal performance, the discontinuities must be non-existent. This means that the quality of the reconstruction is important in terms of the A-Scan performance. Achieving the best spectral spacing was completed by shifting each sub-pulse by slightly greater than the length of the spectrum in samples. This was not an exact procedure as the chirp spectrum has what is known as fresnal integrals, which causes the ripple like behavior at the edges. These ripples impose the difficulty of reconstruction and generate the grating lobes.

In this simulation the created signal emulates the signal used by the SDR, meaning that the bandwidths and signal lengths were exactly the same for ease of implementation. After bandwidth and sampling rate optimization, the simulation was updated to reflect the current system. The reconstructed SWW was plotted and can be seen in figure 6.1

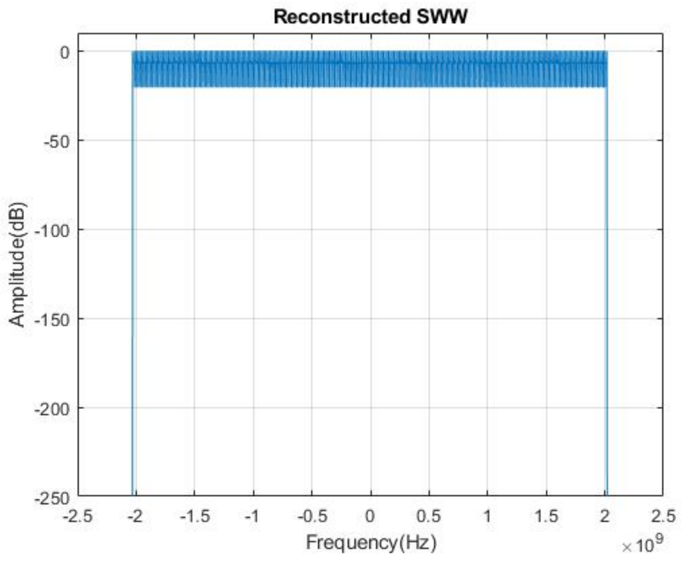

Figure 6.1: Simulated SWW

The only difference between reconstruction and the ideal pulse is the discontinuities in the spectra. The addition of these grating lobes can dramatically reduce the performabce of the radar. It was found in practice that without removing these lobes, the SDR is only good for long range, single object detection or loopback testing. This confirms the importance of applying a grating lobe suppression (GLS) filter which will be discussed in the following section. The results of this simulation prove the algorithm for reconstruction serves its purpose and can be used and after additional filtering can be effective for GPR.

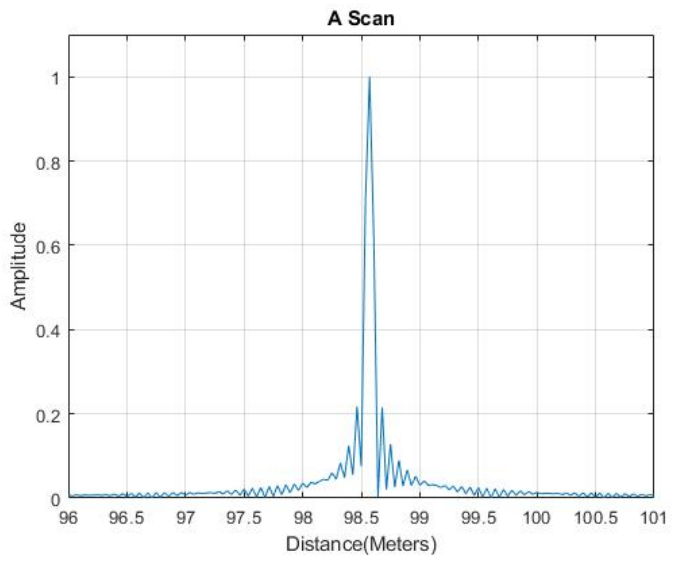

Figure 6.2: Simulated Frequency Stacking Algorithm Output

Performing the inverse transform on the reconstructed spectrum produces the result in 6.2. These results are expected, the waveform holds the sinc shape and is shifted accordingly with the delayed samples imposed by the simulation. In this case the delay is arbitrary and can be altered to any designed value.

Improvements and Grating Lobe Suppression

The theory behind using the frequency stacking method seems relatively straight forward and easily to implement. At a high level, the concept of the signal processing says that for each sub-pulse, perform the match filter operation, shift the signal to it's proper frequency, and then take the inverse Fourier Transform. Unfortunately the addition of theses stacked sub-pulses introduces grating lobes which has been well documented [1]. To understand the magnitude of the grating lobes multiple cases have been simulated to observe their impact. Two categories of simulations were conducted both of which perform the stacked frequency algorithm with the same overal reconstructed bandwidth. The difference between the two lies within the instantaneous frequency used which results in a varying number of sub-pulses required to achieve the same effective bandwidth. Comparing the simulations conducted, it can be seen that the addition of sub-pulses introduces grating lobes which matches the results of [1]. In practice this effect is even more prominent meaning GLS filters are necessary.

Reducing the negative effects of frequency stacking can be achieved by shaping the spectra and removing the discontinuities. From [1] it is realized that using a calibration channel along with an ideal chirp can be leveraged to produce an inversion filter. This filter applied to the raw SWW can reshape the spectra and inherently remove the discontinuities. Theoretically this will greatly suppress the grating lobes and will be explored in simulation prior to implementation.

The suppression filter is generated using an ideal chirp, $M[f_k]$, and a channel model, $W[f_k]$, that is constructing using frequency stacking. Defining the filter as seen below and then normalizing, provides a GLS filter that corrects for the hardware transfer function and discontinuities. This filter can then be applied directly to the frequency spectrum of the reconstructed chirp to achieve the corrected spectrum $\bar{D}_k$.

$$H(f) = \begin{cases} \frac{M[f]}{W[f]} rect[\frac{f}{BW}] & ,W[f] \neq 0 \\ 0 & otherwise \end{cases}$$

The inversion filter by definition only operates across the reconstructed total bandwidth and outside of these values, the filters magnitude is zero. Applying the numerically calculated GLS to the reconstructed spectrum generates the results seen in figure 6.8 of the results section. Applying the GLS filter in simulation guarantees exact reconstruction of the ideal wideband signal, and therefore has no benefit of analysis. This behavior is also seen with the loopback data, therefore the usefulness of this filter is seen best in the results section when it is applied to a separate set of collected data.

6.2 Results

6.2.1 Loopback Verification and A Scan Testing

Before testing the radar with antennas, confirming the loopback operation is important. This verification is completed by replacing the antennas with a direct cable connection to the B210 board. There are two additions to the system, one being a second 20dB attennuator, and the other being a female to female connector allowing for full calibration of the system. Performing this test will place the peak of the matched filter output located at the difference between cable paths effectively measuring the difference in cable lengths. Later this data is used to shift the values in actual antenna measurements allowing for proper distance calibration. Once the loopback has been verified, some basic A scans were conducted to analyze if the radar is correctly performing object detection. After this is confirmed, the system is ready to be used as a GPR and B scan testing will be successful.

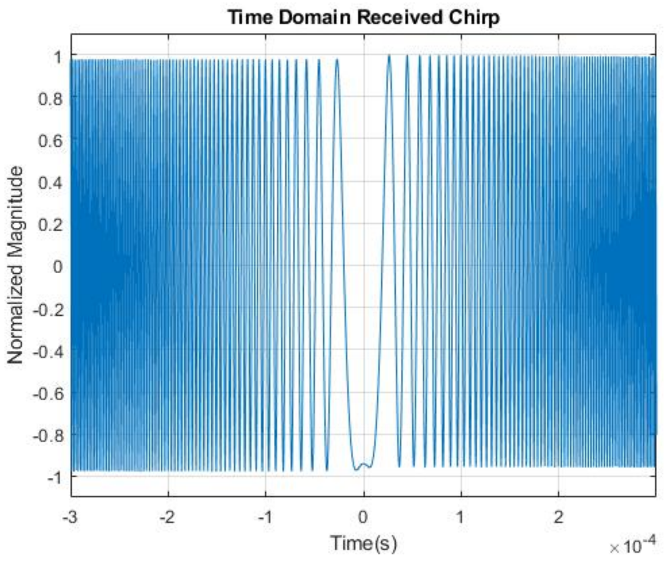

Figure 6.3: Received Chirp Signal

After completing the work detailed in Chapter 5, the signal in figure 6.3 was able to be consistently received with the radar. The shown signal is from a loopback configuration explaining the low noise characteristics. Similar to the frequencies, the time domain signal is centered around zero and the magnitude is normalized by its maximum value. This value is not normalized in the processing algorithms as it would corrupt the data. This was attempted instead of the gain correction early in development, but it was found that the resulting A scan was incorrect. Hardware gain correction for each center frequency yielded proper results and a reasonably similar magnitude for each sub-pulse.

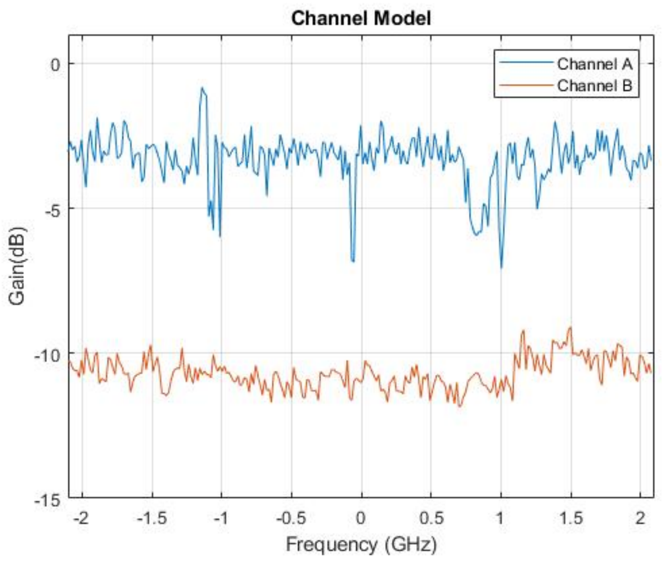

Figure 6.4: Loopback Configuration Channel Model

After the gain correction processes, the channel model of the system can be plotted as in 6.4. Comparing the two channels shows significant improvements relative to the uncorrected models. The correction in place for channel B is slightly better than channel A, but despite additional corrections channel A consistently has issues at a few center frequencies. The problem frequencies in channel A differ no more than $5dB$ which is acceptable and will not cause any issues in practice. Using this data the SWW can be reconstructed as in figure 6.5. The effects of channel A can be seen in the SWW but it seems there is a limit as to how much gain correction can improve the channel model.

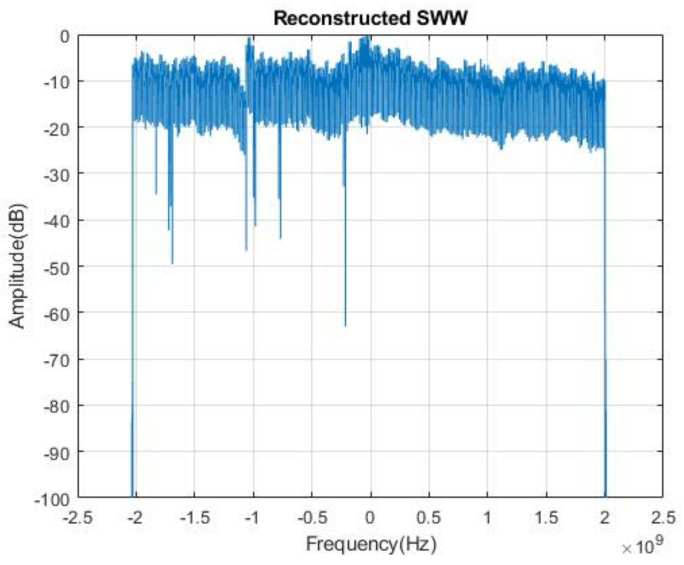

Figure 6.5: Loopback SWW

As mentioned in previous sections, the discontinuities produced by the frequency stacking algorithm are seen in 6.5 and also cause some non-idealities in the A scan seen in 6.6. These additional side lobes are reduced with a decreasing number of sub-pulses. Simulations previously conducted show this relationships, but for implementation the entire range being used consists of 263 sub-pulses that total to a $4.2GHz$ bandwidth and the SWW will be corrected with a GLS filter. The grating lobes are apparent, but the gain correction and object detection are functioning. Experimentally the cable difference was measured to be 1.5 meters and the calculated distance using the frequency stacking method matches at just longer than 1.5 meters. This is within the range resolution of the designed system.

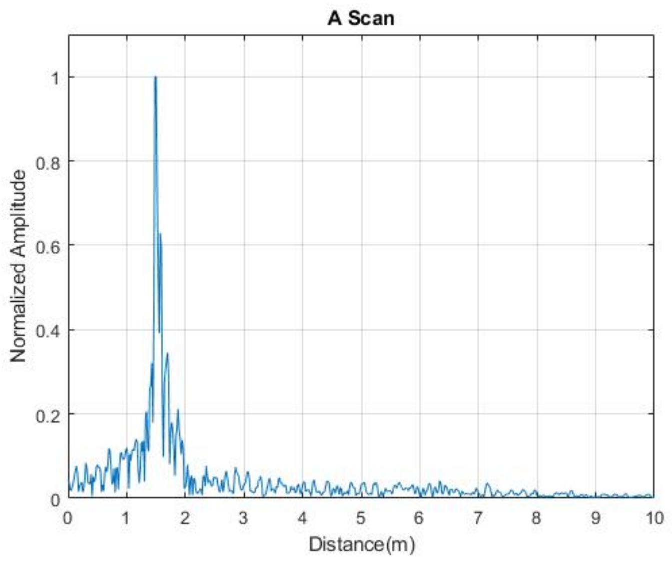

Figure 6.6: Loopback A Scan

After applying the GLS filter the spectrum is seen in figure 6.7. The final step in the algorithm is to perform the IFFT and the resulting A scan can be seen in 6.8. Comparing the corrected and uncorrected scans it is clear the GLS filter provides a dramatic increase in performance. Before the filters, the signals from close objects were likely to be hidden by grating lobes.

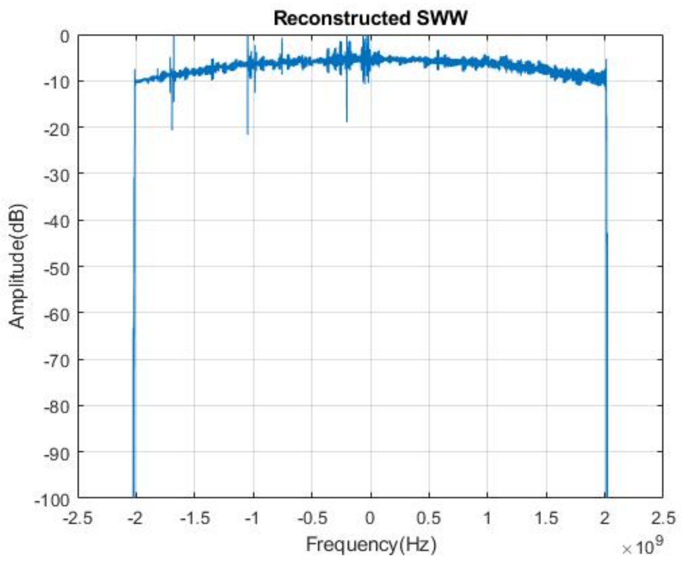

Figure 6.7: Corrected Loopback

This improvement is seen in both the reconstruction frequency response and the time domain A scan. Even though the derivation of the filter appears to be simple, there were significant difficulties when developing the filter. The original plan was to omit simulating the filter since it would provide perfect reconstruction, but this became unavoidable while troubleshooting. The slight differences between the simulated and reconstructed components of the figure needed to be fixed before proceeding. The addition of the GLS filter provides a constant shift of the matched filter response as you are multiplying signals in the frequency domain. This is account for in a simple axis shift.

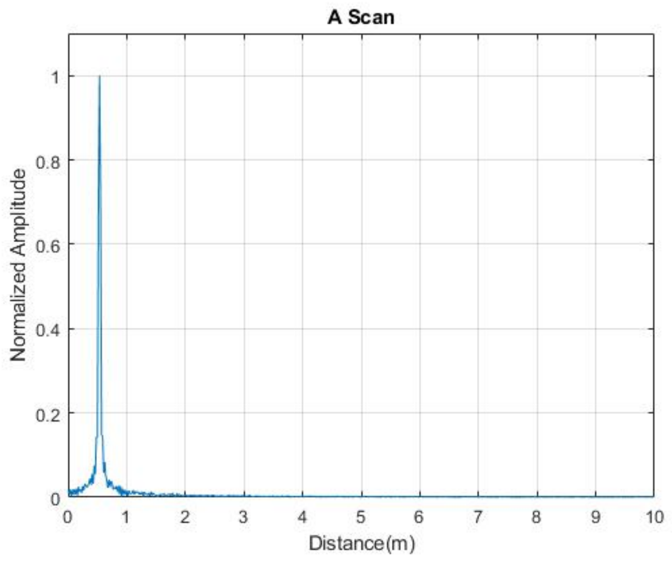

Figure 6.8: Corrected Loopback

6.2.2 B Scan Testing

After the verification of the A scans, the next logical step would be to produce a B scan. As discussed in earlier chapters, the B scan is constructed through placing A scans adjacent to each other in a matrix and plotting. The behavior of the A scans will construct peaks that define images in the B scan when done correctly. The B scan image shown would ideally be conducted in lab using figure 6.14 with a buried metal can. In this setup the mediums between the antennas and the object are air and then a layer of sand in which the object is buried.

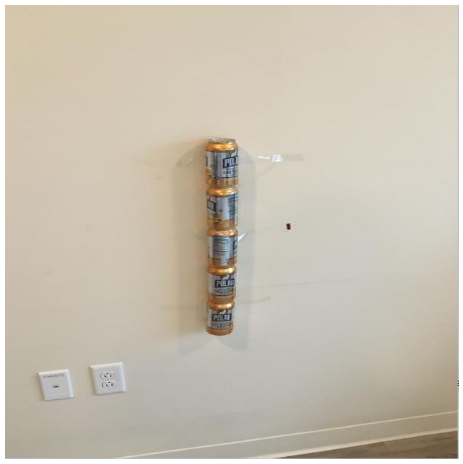

Figure 6.9: Cans to Simulate Buried Object

For the final results, testing was conducted with coke cans taped to the wall to emulate a buried object. The radar was stepped laterally in linear increments of 0.05 meters. A test was conducted prior with a trash can wrapped in aluminum foil taped to the wall, but it was found that a hyperbolic shape was not possible with this sized object. The second test set up in figure 6.9 placed radar 155cm away from the wall. The resulting scan is plotted in figure 6.10 and after performing background removal the improved image is found in figure **??**. The shown figures already have the majority of the direct coupling removed.

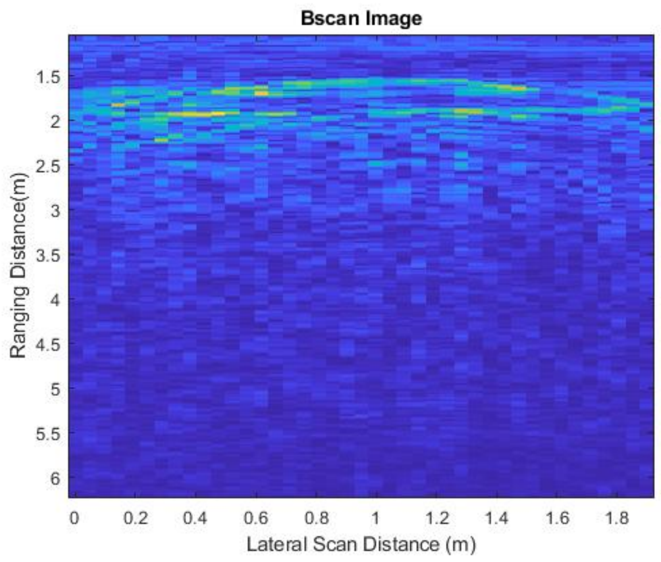

Figure 6.10: Bscan Test, Object 1.5m and back-wall 1.9m

Since the object was far enough away for the direct coupling to not directly impact it, the plotted vector was simply shifted to isolate the can reflection. In attempt to remove additional noise and the reflection from the wall, background removal was performed. This process involves summing each Ascan and then subtracting the mean value from the Bscan image. Results from the removal are plotted in figure **??** and show some improvements. Background removal subtracts the direct coupling, but shifting the data allowed for a higher quality image.

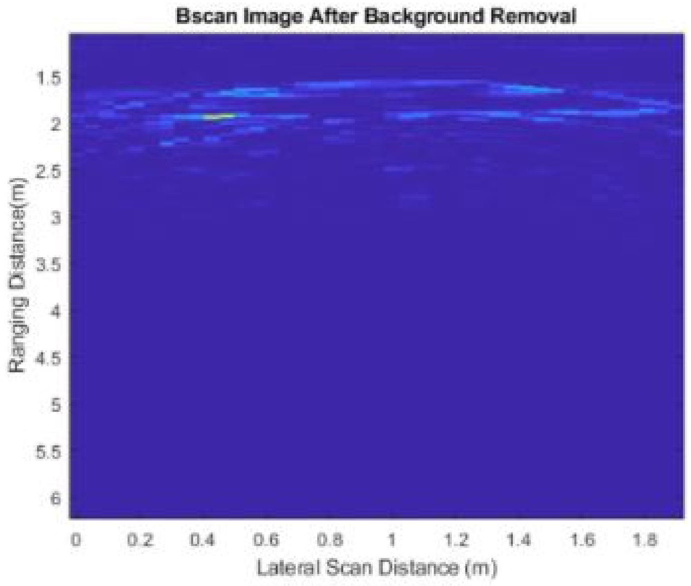

Figure 6.11: Bscan Test Background Removed, Object 1.5m and back-wall 1.9m

The results displayed were expected given the test environment. The hyperbolic shape is seen with its peak at 155cm, matching the actual distance within design specifications. The return signal strength was less than expected, but when conducting Ascan testing with a large sheet of aluminum foil, this reflection was significantly stronger. This means that if the testing environment and object allowed for a strong reflection from the object, the results would be extremely clear. The wall reflection remained somewhat constant throughout the scan and is just below the can reflection. These results prove the successful operation of the radar system conducting GPR scans. The image could be further improved via Eigenvalue background removal [29], but this is outside the scope of this thesis. A better test environment and set up would have yielded better results, but given all circumstances these results are convincing.

Test Environment

In order to obtain clear results, an open environment with a strong reflective material is preferred. Due to the situation at the time, testing was conducted outside with a buried object, and then again inside in a cart, and finally inside with cans taped to a wall. The majority of the scans were conducted inside with the set up shown in figure 6.12 because of personal circumstances preventing any walking or lifting of objects. Final testing was conducted with the help of Yan Zhang as the lab was not available. The indoors at-home testing posed some significant difficulties for a number of reasons. The first being that the sand used in figure 6.12 was relatively rocky and contributed a decent amount of noise to the result. The depth of the cart measured to be about 18 inches which was not significant when attempting to bury a can with a 4 inch diameter in 12 inches of sand. At the time only $3GHz$ of bandwidth was being used due to hardware constraints, which calculates to a resolution of $.05m$, meaning the peak from the sand surface and the can would only be a few samples apart. Technically the system will still be able to distinguish the two objects, but they will be so close together, without background removal techniques the recovered signal was difficult to see.

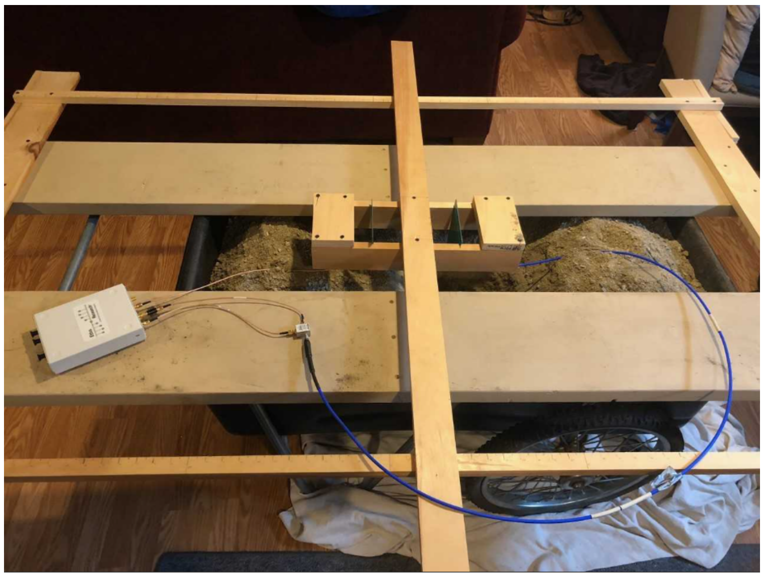

Figure 6.12: Homemade Test Setup

The testing rig was constructed with several pieces of wood, a table saw, and a screw gun. The chassis was created to provide a track for the antennas to slide laterally on. The wood was notched so the antennas would remain stable, and the increments were properly measured to ensure equal distant A scans. The construction of the test piece was successful and allowed for quick B scan testing. Unfortunately the rest of the set up was not conducive to low noise imaging, which was problematic and at this time the GLS filter had not been created. Performing this scan with the current system would have been significantly better.

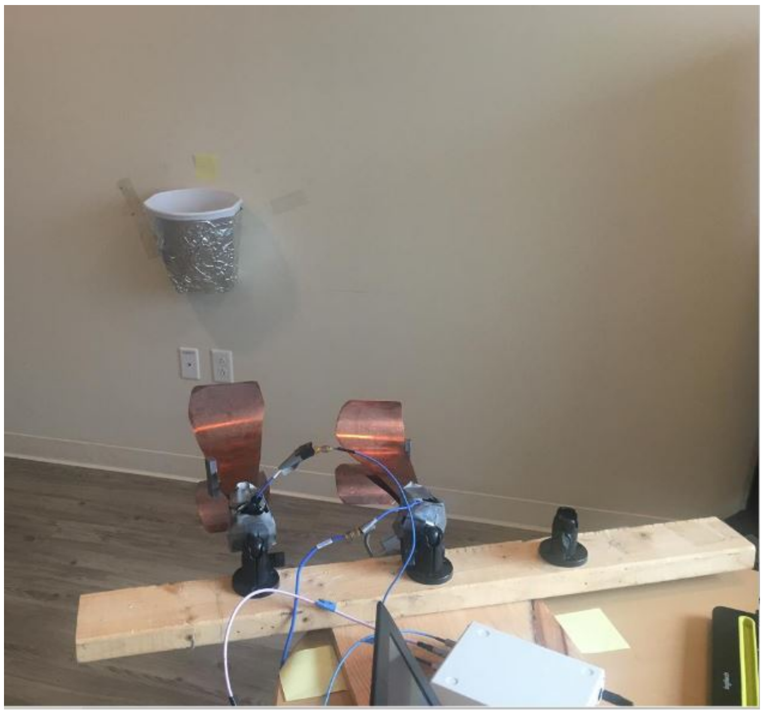

Figure 6.13: Homemade Test Setup 2

The laboratory set up seen in 6.14 provides a cleaner signal for a number of reasons. The medium in which the objects were buried is a fine grain sand that has an estimated relativity permeability of 2 which is a significant improvement over an estimated value with the rock filled sand. Another advantage this set up has GIMMA antennas which were custom made in the UVM lab and have high quality S-parameters. The track used to move the antennas is also automated which improves the ease of use when B scan testing.

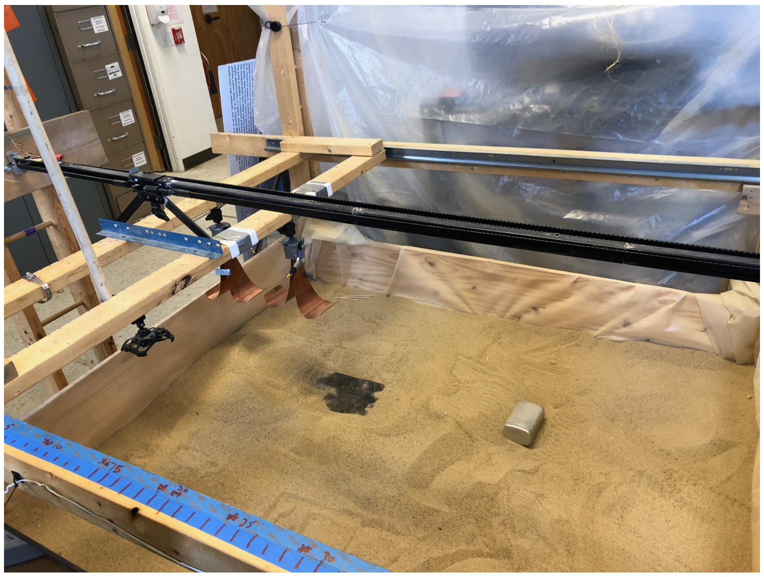

Figure 6.14: Lab Test Setup

Through troubling shooting and multiple test environments, it was found that having an open space is essential. A multitude of Ascans were tested inside and the results proved to be cluttered unless sufficient space was available. The final testing conducted as in figure 6.13 was open enough to allow good results.

6.2.3 Hardware Difficulties

When it comes to troubling shooting circuits, the most difficult task is to correct a system where parts are not only just broken, but partly damaged. During the testing of the radar, the loopback was verified and a B scan was conducted. After generating the final result, it was found that the system was not performing as expected. To begin troubling shooting the loopack was tested again and showed proper measurement of the cables. There were several contributing factors to this problem and it was believed that there were hardware issues in the system. Each channel as described in Chapter

4 had its own transfer function. This meant that observing the collected data made it appear that the splitter was not working properly and producing different outputs from each of its channels. It was also found during this testing that one channel was emitting about 10dB less power. Again this was addressed in Chapter 4, but in the hardware configuration two different sized attenuators were used making this confusing to troubleshoot. It wasn't until after matching attenuators and plotting the response for each individual channels multiple times that it was found the SDR transfer function was causing the problem. The reason the loopback was working is because of the low noise environment. The transfer function of the data showed a massive drop in power at the higher frequencies resulting in these sub-pulses not contributing to the effective bandwidth. When applying the system to A scan testing with antennas, the combination of low power and not yet using a GLS filter resulted in poor scans. Although this appeared to be a hardware issue that could not be fixed through software, the gain tables functioned properly and high quality A scans were realized.

The second main hardware issue that arose during testing actually was due to hardware and the problem was not overcome. In the current system the A scans work during loopback, but when using the antennas the results are filled with noise. After a great deal of time and effort, it was determined that one or both of the antennas was not functioning properly. When referencing the sine wave SFCW system, the same responses were obtained. To achieve the collected results a second set of testing was conducted by Yan Zhang with his antennas and set up. These scans functioned properly and were used to generate the successful B scan. Fortunately the second set up was available to troubleshoot.

6.3 Future Work

With completion of this thesis is it realized that there are various avenues for future work. These can stem from multiple places, but the primary two are a continuation of what has been done, and an improvement of the current project. To begin, there are a few improvements that can be made to the current system. In regards to operation of the radar, there are no realized improvements to be made outside of speed and further background removal and lobe reduction. The system functions well for GPR, but if used for other applications it may not be fast enough. Re-writing the code in *C* instead of Python would provide some improvements, but using a better computer along with code optimization would provide the most benefit. As mentioned earlier, there are other methods of background removal that may be beneficial in future research. It appears there were slight grating lobes in the plots, but it was unclear if this was from the environment or lack of lobe suppression.

Operating this device to produce SAR imaging is currently undergoing research and is a great application for this system. Being able to generate images from airborne radar will pose several challenges that may require speed improvements. The device could have to perform at a higher level to decrease scan time and also have some way to run wireless from an operating system. The potential for this system is endless and exciting to watch as technology progresses.

6.4 Final Words

This thesis proposed the development of a SDR system capable of performing in GPR applications. The developed system is fully capable of transceiving phase coherent LFM signals, synthesizing the wideband signal, and finally conducting B scans. As with any project there were a number of difficulties that were overcome. The main factor that hindered the project was the lack of lab access during the COVID-19 pandemic. At the time of the crisis, the project had reached the point where lab testing was necessary. The at-home setup was not sufficient for testing the device with antennas, and therefore the results yielded were not as easily obtained. With that being said, multiple environments were tested in and the system performs exactly as expected. Hardware issues face throughout the project were overcome via a combination of software compensation and component replacements.

The designed system is fully capable of a wide range of object detection based applications. The synthetic wideband signal has successfully been synthesized and if using a powerful computer the system performance could increase dramatically. Currently the processing rate of the computer in use limits the radar to a $16MHz$ sampling rate, meaning an instantaneous bandwidth of equivalent value. The radar itself is capable of $56MHz$ and if the computer could support this transfer rate, a dramatic performance increase in both speed and image quality would be seen. There would be less grating lobes and the device would be able to complete the wideband chirp in less time. Optimizing the processing equipment would be a worthwhile task.

It is unfortunate that concluding this thesis does not include lab-setting results as these would be exciting to display. Previous testing in such an environment resulted in extremely low noise and clear object detection with no background removal. Implementing the current system with this environment would show clean results and would make comparisons between systems easier. On the contrary, a high quality test set up is only beneficial for testing. The system will have to operate outside of a borderline "ideal" environment, so although the testing was not perfect it does prove functionality in a relatively cluttered environment.

Bibliography

[1] S. Prager, T. Thrivikraman, M. Haynes, J. Stang, D. Hawkins and M. Moghaddam, "Ultra-wideband synthesis for high-range resolution software defined radar," 2018 IEEE Radar Conference (RadarConf18), Oklahoma City, OK, 2018

[2] S. C. Carey and W. R. Scott, "Software defined radio for stepped-frequency, ground-penetrating radar," 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, 2017

[3] S. Sharma, P. Jena, R. Kuloor (2013), âAmbiguity Function Analysis of SFCW and Comparison of Impulse GPR and SFCW GPRâ In: 9th International Radar Symposium India â 2013.

[4] Yanhua Wang, "Bandwidth synthesis for stepped chirp signal: A multichannel sampling prospective," IET International Radar Conference 2013

[5] R. T. Lord and M. R. Inggs, "High range resolution radar using narrowband linear chirps offset in frequency," Proceedings of the 1997 South African Symposium on Communications and Signal Processing. COMSIG '97, Grahamstown, South Africa, 1997

[6] Soumekh, M., [Synthetic aperture radar signal processing with MATLAB algorithms], Wiley-Interscience (1999)

[7] Ozdemir, Caner. Inverse Synthetic Aperture Radar Imaging with MATLAB Algorithms. Wiley.

[8] Utsi, Erica Carrick. Ground Penetrating Radar: Theory and Practice. Butterworth-Heinemann, an Imprint of Elsevier, 2017

[9] Dang Hongxing, "Stepped frequency chirp signal SAR imaging," 2007 1st Asian and Pacific Conference on Synthetic Aperture Radar, Huangshan, 2007

[10] W. Zhai and Y. Zhang, "A stepped frequency chirp scaling algorithm for high resolution SAR imaging," 2011 3rd International Asia-Pacific Conference on Synthetic Aperture Radar (APSAR), Seoul, 2011

[11] Yuan Haojuan, Gao Meiguo and Liu Guoman, "Coherent spectrum synthesis of frequency-stepped chirp signal," 2009 IET International Radar Conference, Guilin, 2009

[12] X. Shibo, G. Jialong and W. Bocai, "Research on high resolution SAR based on frequency-stepped chirps," 2009 2nd Asian-Pacific Conference on Synthetic Aperture Radar, Xian, Shanxi, 2009

[13] H. J. MartĂnez, S. Alvarez and M. A. YarlequĂŠ, "Assessing the performance of three type of UWB antennas for FMCW GPR imaging," 2018 International Conference on Electromagnetics in Advanced Applications (ICEAA), Cartagena des Indias, 2018

[14] Z. Zhao, M. Yao, X. Deng, K. Yuan, H. Li and Z. Wang, "A Novel Ionospheric Sounding Radar Based on USRP," in IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 10, pp. 1800-1804, Oct. 2017

[15] A. B. Suksmono, âA simple solution to the uncertain delay problem inusrp based sdr-radar systems,âarXiv preprint arXiv:1309.4843, 2013

[16] D. J. Rabideau, "Nonlinear synthetic wideband waveforms," Proceedings of the 2002 IEEE Radar Conference (IEEE Cat. No.02CH37322), Long Beach, CA, USA, 2002

[17] Y. Yu et al., "Radar polarimetry analysis applied to fully polarimetric Ground Penetrating Radar," Proceedings of the 15th International Conference on Ground Penetrating Radar, Brussels, 2014, pp. 642-646, doi: 10.1109/ICGPR.2014.6970504.

[18] S. Ye, J. Chen, L. Liu, C. Zhang and G. Fang, "A novel compact UWB ground penetrating radar system," 2012 14th International Conference on Ground Penetrating Radar (GPR), Shanghai, 2012

[19] G. Tronca, I. Tsalicoalou, S. Lehner and G. Catanzariti, "Comparison of pulsed and stepped frequency continuous wave (SFCW) GPR systems," 2018 17th International Conference on Ground Penetrating Radar (GPR)

[20] P. T. W. Wong, W. W. L. Lai and M. Sato, "Time-frequency spectral analysis of step frequency continuous wave and impulse ground penetrating radar," 2016 16th International Conference on Ground Penetrating Radar (GPR), Hong Kong, 2016, pp. 1-6, doi: 10.1109/ICGPR.2016.7572694.

[21] V. Kyovtorov, C. Kabakchiev, V. Behar, G. Kuzmanov, I. Garvanov and L. Doukovska, "FPGA Implementation of Low-Frequency GPR Signal Algorithm using Frequency Stepped Chirp Signals in the time Domain," 2008 International Radar Symposium, Wroclaw, 2008, pp. 1-4, doi: 10.1109/IRS.2008.4585775.

[22] R. T. Lord and M. R. Inggs, "High resolution SAR processing using stepped-frequencies," IGARSS'97. 1997 IEEE International Geoscience and Remote Sensing Symposium Proceedings. Remote Sensing - A Scientific Vision for Sustainable Development, Singapore, 1997

[23] S. Hamran and K. Langley, "A 5.3 GHz step-frequency GPR for glacier surface characterisation," Proceedings of the Tenth International Conference on Grounds Penetrating Radar, 2004. GPR 2004., Delft, The Netherlands, 2004, pp. 761-764.

[24] F. Parrini, F. Papi and M. Pieraccini, "An ultra high resolution stepped frequency GPR for civil engineering applications," 2015 8th International Workshop on Advanced Ground Penetrating Radar (IWAGPR), Florence, 2015, pp. 1-4, doi: 10.1109/IWAGPR.2015.7292624.

[25] M. Sato, F. Kong, Z. Zeng and G. Fang, "Antenna development and a stepped-frequency GPR system for landmine detection," Proceedings of the 2nd International Workshop on-Advanced Ground Penetrating Radar, 2003., Delft, Netherlands, 2003, pp. 168-171, doi: 10.1109/AGPR.2003.1207313.

[26] T. Kido, Y. Yokota, F. Kawahara and M. Sato, "Wide band stepped-frequency ground penetrating radar," 2011 IEEE International Geoscience and Remote Sensing Symposium, Vancouver, BC, 2011, pp. 55-58, doi: 10.1109/IGARSS.2011.6048896.

[27] S. B. Thomas and L. P. Roy, "A Comparative study on calibration technique for SFCW ground penetrating radar," 2017 International conference on Microelectronic Devices, Circuits and Systems

[28] R. N. Strickland and He Il Hahn, "Wavelet transform methods for object detection and recovery," in IEEE Transactions on Image Processing, vol. 6, no. 5, pp. 724-735, May 1997

[29] U. S. Khan and W. Al-Nuaimy, "Background removal from GPR data using Eigenvalues," Proceedings of the XIII Internarional Conference on Ground Penetrating Radar, Lecce, 2010

[30] Near Surface Geophysics. GitHub, github.com/NSGeophysics.

[31] "945-70 Log Periodic Antenna." Comprod Inc., comprodcom.com/shop/antennas-clamps/log-periodic-antennas/945-70-log-periodic-antenna/.

Appendix A

Source Code

A.0.1 Python Code

B210 SFCW Code

```python
# import libraries
import uhd
from uhd import libpyuhd as lib
import numpy as np
import threading
import time
import scipy.io
from scipy.signal import chirp, spectrogram
from radar_parameters2 import *  # import the parameters used in the radar
from gain_parameters import*     # import the gain tables
import matplotlib.pyplot as plt
from scipy.io import loadmat



# These delays are used to align the buffers, used later in code
INIT_DELAY = 0.01
TX_DELAY =0.012

# setup general parameters, see documentation for detailed explanation of params
def setup_device(samp_rate, master_clock_rate, tx_gain, rx_gain):
    """
    set up everything except center frequency
    :return: a MultiUSRP device object, a tx_streamer, a rx_streamer
    """
    # define device parameters for daughter board and clock of the system
    args = "type = b200"
    subdevice = "A:A A:B"  # select subdevice in the daughterboard
    freq_clock_source = "internal"
    rx_band_width = samp_rate
    tx_band_width = samp_rate

    channel_list = [0, 1]

    # create a usrp device and set up it with the device parameters defined above
    usrp = uhd.usrp.MultiUSRP(args)

    # set clock ant time
    usrp.set_clock_source(
        freq_clock_source
    )  # this sets the source of the frequency reference, typically a 10 MHz signal
    usrp.set_master_clock_rate(master_clock_rate)

    # select subdevice
    subdevice_spec = lib.usrp.subdev_spec(subdevice)
    usrp.set_rx_subdev_spec(subdevice_spec)
    usrp.set_tx_subdev_spec(subdevice_spec)
    print("Using Device: {}".format(usrp.get_pp_string()))

    # set sample rate of ADC/DAC
    usrp.set_tx_rate(samp_rate)  # this will set over all channels
    usrp.set_rx_rate(samp_rate)
    print("Actual RX0 rate: {} Msps".format(usrp.get_rx_rate(0) / 1e6))
    print("Actual RX1 rate: {} Msps".format(usrp.get_rx_rate(1) / 1e6))
    print("Actual TX0 rate: {} Msps".format(usrp.get_tx_rate(0) / 1e6))
    print("Actual TX1 rate: {} Msps".format(usrp.get_tx_rate(1) / 1e6))

    # set bandwidth
    usrp.set_tx_bandwidth(tx_band_width, channel_list[0])
    usrp.set_tx_bandwidth(tx_band_width, channel_list[1])
    usrp.set_rx_bandwidth(rx_band_width, channel_list[0])
    usrp.set_rx_bandwidth(rx_band_width, channel_list[1])
    print("Actual RX0 bandwidth = {} MHz".format(usrp.get_rx_bandwidth(0) / 1e6))
    print("Actual RX1 bandwidth = {} MHz".format(usrp.get_rx_bandwidth(1) / 1e6))
    print("Actual TX0 bandwidth = {} MHz".format(usrp.get_tx_bandwidth(0) / 1e6))
    print("Actual TX1 bandwidth = {} MHz".format(usrp.get_tx_bandwidth(1) / 1e6))

    # create stream args and tx streamer
    st_args = lib.usrp.stream_args("fc32", "sc16")
    st_args.channels = channel_list

    tx_streamer = usrp.get_tx_stream(st_args)  # create tx streamer
    rx_streamer = usrp.get_rx_stream(st_args)  # create rx streamer

    return usrp, tx_streamer, rx_streamer

```

```

def init_usrp_device_time(usrp):
    """
    set the usrp device time to zero
    :param usrp: a MultiUSRP Device
    """
    usrp.set_time_now(lib.types.time_spec(0.0))
    return


# define tx worker: the tx worker should always
#send data until all the freq data are saved
def tx_worker(tx_streamer, tx_data, tx_md):

total_num_samps = 2000 # this defines the number of samples to send
num_acc_samps = 0
num_tx_samps = 0
 # define timing specifications
tx_md.time_spec = uhd.types.TimeSpec(usrp.get_time_now().get_real_secs() + TX_DELAY)

while num_acc_samps < total_num_samps:
    num_tx_samps += tx_streamer.send(tx_data, tx_md) * num_channels
    num_acc_samps += min(total_num_samps - num_acc_samps,tx_streamer.get_max_num_samps())
# print(num_tx_samps)


# the tune_center_freq represents the tx/rx tune state
def tune_center_freq(usrp, target_center_freq):
    global state
    channel_list = [0, 1]

    # tune center freqs on all channels
    usrp.set_rx_freq(lib.types.tune_request(target_center_freq), 0)
    usrp.set_rx_freq(lib.types.tune_request(target_center_freq), 1)
    usrp.set_tx_freq(lib.types.tune_request(target_center_freq), 0)
    usrp.set_tx_freq(lib.types.tune_request(target_center_freq), 1)


    if current_freq < 2.6e9 :
        usrp.set_tx_gain(tx_gain, channel_list[0])
        usrp.set_rx_gain(rx_gaina+gain_a[0,gain_index], channel_list[0])
        usrp.set_rx_gain(rx_gainb+gain_b[0,gain_index], channel_list[1])

    elif current_freq >= 2.6e9:
        usrp.set_tx_gain(tx_gain2, channel_list[0])
        usrp.set_rx_gain(rx_gaina+gain_a[0,gain_index],channel_list[0])
        usrp.set_rx_gain(rx_gainb+gain_b[0,gain_index],channel_list[1])

    # wait until the lo's are locked
    while not (
        usrp.get_rx_sensor("lo_locked", 0).to_bool()
        and usrp.get_tx_sensor("lo_locked", 0).to_bool()
    ):
        pass

    state = "rx_and_save_data"
    return


def rx_and_save_data(usrp, rx_streamer, rx_buffer, rx_md, num_rx_samps, current_freq):
    global state

    if current_freq > end_freq:
        state = "Done"
        return

    stream_cmd = lib.types.stream_cmd(lib.types.stream_mode.num_done)
    stream_cmd.num_samps = num_rx_samps  # receive 10 periods
    stream_cmd.stream_now = False
    stream_cmd.time_spec = usrp.get_time_now() + lib.types.time_spec(0.01)
    rx_streamer.issue_stream_cmd(stream_cmd)  # tells all channels to stream

    rx=rx_streamer.recv(rx_buffer, rx_md)
    # print(rx)
    # print "Current freq =  {}".format(current_freq)

    state = "tune_center_freq"


# utility function: prepare tx data
def generate_tx_data(samp_rate):
    channel_list = (0, 1)
    # generate chirp params
    fs = samp_rate
    N2 = 85000                  # this is length of chirp
    bw = 8e6                    # bandwidth of the chirp
    n = np.arange(0,N2-1)-N2/2
    t = n/fs

    send_chirp = np.array(np.exp(1j*np.pi*.5*(bw/t[-1])*(t**2)), dtype = np.complex64)
    N=4096
    send_chirp = np.pad(send_chirp, (N), 'constant', constant_values=(0))
    wave_ampl = .8

    # since we have two channels to transmit, we tile tx_data
    tx_data = np.tile(send_chirp, (1, 1)) # also tiles to send just one period
    tx_data = np.tile(tx_data[0], (len(channel_list),1))  # 2 is the length of channel

    # create tx metadata
    tx_md = lib.types.tx_metadata()
    tx_md.start_of_burst = True
    tx_md.end_of_burst = False
    tx_md.has_time_spec = True

    length_wave_one_period = send_chirp.size
    chirp_duration = length_wave_one_period * samp_rate

    return tx_data, tx_md, send_chirp, length_wave_one_period, chirp_duration


start_state = "tune_center_freq"
# state has 3 values: tune_center_freq, rx_and_save_data, Done
state = start_state  # initialize state to start_state

# set up a usrp device object and get usrp, tx_streamer, rx_streamer
usrp, tx_streamer, rx_streamer = setup_device(samp_rate, master_clock_rate, tx_gain)



# prepare rx_buffer_list and rx metadata
# the rx_buffer_list is preallocated
num_rx_samps = length_wave_one_period *3
rx_md = lib.types.rx_metadata()

# create the receive buffers and freq step
num_freqs = (end_freq - start_freq) / float(freq_step) + 1
rx_buffer_list = []
rx_buffer_idx = 0
for _ in range(int(num_freqs)):
    rx_buffer = np.zeros((2, num_rx_samps), dtype=np.complex64)
    rx_buffer_list.append(rx_buffer)

# start the tx work and set up basics for B210

init_usrp_device_time(usrp)  # set device time to 0
num_channels = tx_streamer.get_num_channels()
max_samps_per_packet = tx_streamer.get_max_num_samps()
current_freq = start_freq
gain_index = 0

# create the tx and rx workers before running
t1_tx_worker = threading.Thread(target=tx_worker, args=(tx_streamer, tx_data, tx_md))

start = time.time()
```

```python


# begin the loop
while True:

    if (
        state == "tune_center_freq"
    ):
        tune_center_freq(usrp, current_freq)
        current_freq = current_freq + freq_step
        gain_index = gain_index + 1

    elif state == "rx_and_save_data":
    # this is the state to call both workers and save the data in the rx buffer


    r1_rx_worker.start()
    t1_tx_worker.start()

    r1_rx_worker.join()
    t1_tx_worker.join()

    rx_buffer_idx = rx_buffer_idx + 1
    elif state == "Done":
        break
end = time.time()
print("total scan time = {} seconds".format(end - start))


# save all data to a file
rx_buffer_list = np.array(rx_buffer_list)
scipy.io.savemat("./Final/inside/loop2.mat", {"loop2": rx_buffer_list})
print(rx_buffer_list.shape)



a = rx_buffer_list[50,1,:]
b = rx_buffer_list[100,1,:]
c = rx_buffer_list[150,1,:]
d = rx_buffer_list[190,1,:]



g = rx_buffer_list[50,0,:]
h = rx_buffer_list[100,0,:]
i = rx_buffer_list[150,0,:]
j = rx_buffer_list[190,0,:]

```

```



plt.figure(1)
plt.plot(a)
plt.plot(g)
plt.figure(2)
plt.plot(b)
plt.plot(h)
plt.figure(3)
plt.plot(c)
plt.plot(i)
plt.figure(4)
plt.plot((d))
plt.plot((j))


plt.show()
```

Gain Parameters

```


import scipy.io

# Load data and save
gain_a = scipy.io.loadmat('gain_a.mat')
gain_b = scipy.io.loadmat('gain_b.mat')
gain_a= gain_a['gain_a']
gain_b = gain_b['gain_b']
# repeat for channel B
gainhw_a = scipy.io.loadmat('gainhw_a.mat')
gainhw_b = scipy.io.loadmat('gainhw_b.mat')
gainhw_a= gainhw_a['gainhw_a']
gainhw_b = gainhw_b['gainhw_b']
```

Radar Parameters

```
# define radar parameters
start_freq = 800e6
freq_step = 16e6
end_freq = 4e9
# device parameters
samp_rate = 16e6
master_clock_rate = 16e6
```

```

# the second tx gain was used, but not needed if using gain tables
tx_gain = 80
tx_gain2 = 85
rx_gaina = 20
rx_gainb = 20
rx_gain =  20
```

A.0.2 MATLAB Processing

SFCW Processing

```
%---------------------------------------------------------------
%---Patrick Fiske-----------------------------------------------
%---Chirp SFCW Ascan Processing Code----------------------------
%---7/22/2020---------------------------------------------------
%---------------------------------------------------------------
 clear all
 close all
%% load data and parameters

C1 = rawdata;         % enter raw data here
start_freq = 800e6;   % enter the start frequency
end_freq = 4e9;       % enter end frequency
sample_rate = 16e6;   % enter the sampling rate of the radar

%



fc = start_freq;          % initial frequency
BWf = 8.4e9;              % frequency bandwidth
N = length(C1(1,:));      % this is the length of your rx buffer
fs = sample_rate;         % sub-pulse sample rate
fs2 = 16.8e9;             % sample rate to scale wideband signal
To = N/fs;                % sub-pulse period
N2=(75346063/2);          % half the length of rx buffer
To2 = N2/fs2;             % period for scaling
c = (3e8);                % speed of light
t = -To/2:To/(N-1):To/2; %time vector
fmax = 4.2e9;             %wideband bandwidth *2?
ttmeter = 0:c*To2/(N2-1):c*To2;                    %time domain vector - meters
ttinch = 0:39.3701*c*To2/(N2-1):39.3701*c*To2;   %time domain vector - inches
f  =-fmax:BWf/(2*N2-1):fmax;                       %frequency domain - centered at zero
w = gpuArray(tukeywin(277574)');                   %turkey window length of rx buffer

```

```
fs_up = 15e6;           %used to calculate upsampling, L
Tp = 40001 / fs;        %period for upsampling -
L = (fs)*Tp;            %upsampling factor
Lup = 2 * L;
n_zeros = round(Lup / 2);
Dn1 = zeros(1,75346063,'gpuArray'); % pre-allocation of wideband vector

%% Frequency Stacking
% set up parameters for zero padding
tic
delta_f = 0;

%this first loop loads the data
   for n = 1:201
        rx(n,:) = gpuArray(C1(n,2,:));   % allocate to gpu for faster processing
        rx(n,:) = squeeze(rx(n,:));      % remove extra dimension
        rx(n,:) = rx(n,:)-mean(rx(n,:)); % remove dc component
        b(n) = max(abs(rx(n,:)));        % variable for gain tables

        rx2(n,:) = gpuArray(C1(n,1,:));
        rx2(n,:) = squeeze(rx2(n,:));
        rx2(n,:) = rx2(n,:)-mean(rx2(n,:));
        a(n) = max(abs(rx2(n,:)));

   end
%this second loop performs frequency stacking
    for n = 1:201

    DFT_rx = gpuArray((fft(rx(n,:)))./363);          %fft on gpu
    DFT_rx = DFT_rx(1000:278573);
    DFT_reference = gpuArray((fft(rx2(n,:)))./363);  % repeat for reference
    DFT_reference = DFT_reference(1000:278573);
    DFT_D0 = (DFT_rx .* conj(DFT_reference));
    DFT_D0_shifted = gpuArray(fftshift(DFT_D0));     % fftshift the result
    DFT_D0_shifted = [zeros(1, n_zeros) DFT_D0_shifted zeros(1, 75028488)]; % pad
    DFT_D0_shifted = circshift(DFT_D0_shifted, 23746744 + round(delta_f));  % shift
    delta_f = delta_f+2*69000;

    count2 =n % displays counter
    Dn1 = Dn1+DFT_D0_shifted; %sum each shifted pulse
    end
toc
%% Plotting Frequency Response and A Scans
%
D1 = gather(Dn1);       % bring data back to processor
D = D1./max(D1);        % normalized data

L = length(D);          % length for use in filter
```

```
h = hamming(L)';        % hamming filter


set(0,'defaultfigurecolor',[1 1 1])
figure()
plot(f,(20*log10(abs(h.*D.*GLS))))  % hamming * data * gls filter
title('Reconstructed SWW');xlabel('Frequency(Hz)');ylabel('Amplitude(dB)');
ylim([-100 0])
grid on
hold on


%creat time domain signal
y=(fftshift(ifft(h.*D.*GLS_yan))); % ifft of the filtered data
y = y/max(y);                      % normalize data
y = circshift(y,-77);
y = y(:,[37672972:75346002]);      % matched filter output


figure()
plot(ttmeter,((abs(y)))); title('A Scan');
xlim([0 10])
ylim([0 1.1])
grid on

figure()
plot(ttinch,(abs(y))); title('A Scan'); xlabel('Distance(Inches)'); ylabel('Amplitude')
xlim([0 100])
ylim([0 1])
grid on
```

GLS Filter

```
%% THIS FILE CREATES A GLS FILTER



%% This creates M of the GLS fitler
Bi = 4.025e9; % bandwidth of w(t) in Hz
Bwf = 8.4e9;
fmax = 4.2e9;
% Tp = 0.0024; % pulse duration in seconds
fs = 8.4e9;
samps = 75346063; %75346063 %36210000
Tp = samps/fs;
 % sample freqeuncy of baseband ADC and DAC
t = -Tp/2: 1/fs : Tp/2;
```

```
s = w(t, Bi, Tp);
S = fftshift(fft(s)).*conj(fftshift(fft(s)));
f_sim = -fmax:Bwf/(samps):fmax;

figure()
plot(abs(S))

%% Creates the GLS Filter
M1 = S(1,[1:75346063]);
M = (M1./max(M1));      % normalize
W = data;               % loopback data

% plot both and GLS
plot(20*log10(abs(M)))
hold on
plot(20*log10(abs(W)))

GLS = M./W;             % this is the actual GLS
figure()
plot(20*log10(abs(GLS)))


GLS2 = GLS;
GLS2(~isfinite(GLS))=0; % remove infinite values

%% Save the filter
GLShw2 = GLS2;
save('hw2_GLS.mat','GLShw2');




% GLS_s = GLS_yan(1,[23860000:51456000]);
% plot(abs(20*log10(GLS_yan)))
% GLS_yan2 = [ones(1,23859999) GLS_s ones(1,23890063)];
```

Simulation Code

```
%%%%%% THIS CODE SIMULATES BOTH INDIVIDUAL OPERATION AND
%%%%%% FULL STEPPED FREQUENCY WITH LFM SIGNAL

%%%%%% FOR MORE DETAILS SEE OFFICIAL SFCW CODE, THIS IS THE SAME THING JUST
%%%%%% USES SIMULATIONS, VARIABLES WERE FREQUENTLY CHANGED TO PROVIDE
%%%%%% FURTHER UNDERSTANDING OF DATA. DO NOT USE THIS TO PROCESS ANY REAL
%%%%%% DATA

%% create the signal
```

```
Bi = 20e6;                            % bandwidth of sub-pulse
fs = 20e6;                            %  sampling rate (2BW)
samps = 138000*2                      % number of samples per chirp
Tp = samps/fs                         % pulse duration
t = -Tp/2: 1/fs : Tp/2;               % negative time vector centering pulse at 0
w_tm = w(t, Bi, Tp);                  % create the chirp from w() function
w_tm1 = w_tm;                         % simply rename chirp to not mess up original

N = length(w_tm1);                    % length of chirp
%%%%%%%%% w_tm1 can be used as a reference signal. Now we need a simulated
%%%%%%%%% receive signal. Must add noise and delay

noise = .2*rand(1,N);                 % create noise vector
w_tm2 = circshift(w_tm1, -10)+noise; % add the noise

set(0,'defaultfigurecolor',[1 1 1])  %plot
figure()
plot(t,real(w_tm2))
title('Simulated RX Signal')
ylim([-1.1 1.2])
xlim([-.0005 .0005])


set(0,'defaultfigurecolor',[1 1 1])
figure()
% plot time series of w_t
plot(t,real(w_tm1))
title('Simulated TX Signal for Reference')
xlim([-.0005 .0005])
ylim([-1.1 1.1])


%--- Matched Filtering -----------------------------------------
% this section performs the matched filter using the paramters of an ideal
% wideband signal

X=fft(w_tm1)/N;
S = conj(fft(w_tm2)/N);
H=S;
Y=X.*H;                                % perform match filter
y1=fftshift(ifft(Y));                  % fftshift the response
N2 = length(y1)/2;                     % number of samples used for vector
y = y1(:,[138000:276000]);;               % take the middle sample as t=0 there
y = y./max(y);                         % normalize
fs2 = 4.4e9;
N3=length(y);                          % used for axis scaling
To2 = N3/fs2;                          % pulse duration
c=3e8;
```

```
tt = 0:c*To2/(N3-1):To2*c;            % time vector
fc1 = 800e6; BWf1 = 4.2e9;
N = length(Y);
f1  =fc1:BWf1/(N-1):(fc1+BWf1);
 %time vector in micro seconds
%----Plot matched filter output----------------------------
% figure()
set(0,'defaultfigurecolor',[1 1 1])
plot(abs(y))
set(gca,'color','w');
xlabel('Distance(m)')
ylabel('Normalized Magnitude')
xlim([0 30])
ylim([0 1.1])
title('Simulated Matched Filter Output');
grid on

figure()
plot(f1,abs(fftshift(Y)))

%%%%%%%%%%%%%%%%%%%%% at this point the match filter simulation is working
%%%%%%%%%%%%%%%%%%%%% for one scan...can change the sampling frequency to
%%%%%%%%%%%%%%%%%%%%% represent the resolution of sfcw..this plot uses 6ghz
%%%%%%%%%%%%%%%%%%%%% bw


%% set up
Bi = 4.2e9; % bandwidth of w(t) in Hz
Bwf = 8.4e9;
fmax = 4.2e9;
fmax2 = 2.1e9;
% Tp = 0.0024; % pulse duration in seconds
fs = 8.4e9;
% t = -Tp/2: 1/fs : Tp/2;
samps2 = 75346063;


To = N/fs; %pulse duration
Beta = Bwf/To;
N2=(samps2/2); % number of DTFT samps / 2
To2 = N2/fs;
c = 3e8;
t = 0:To/(N-1):To; %time vector
ttmeter = 0:c*To2/(N2-1):c*To2; %time domain vector - meters
ttinch = 0:39.3701*c*To2/(N2-1):39.3701*c*To2; % time domain vector - inches
f_sim = -fmax:Bwf/(samps2-1):fmax;

% filter each subpulse ------------------dont need because ideal
```

```
w=1;
%% this process takes forever only do once if need be
% set up parameters for zero padding
tic
fs = 15e6;
Tp = 40001 / fs;
L = (fs)*Tp;
Lup = 2 * L;
n_zeros = round(Lup / 2);
Dn1 = 0;Dn2 = 0;Dn3 = 0;Dn4 = 0;Dn5 = 0;
delta_f = 4*69000;

   for n = 1:10
        rx(n,:) = w_tm1(1,:);
        rx2(n,:) = w_tm2(1,:);
        count=n
   end
    Dn1 = zeros(1,75346063);
    for n = 1:10
    DFT_rx = fft(rx(n,:));
    DFT_rx = DFT_rx(1000:276001);
    DFT_reference = fft(rx2(n,:));
    DFT_reference = DFT_reference(1000:276001);
    DFT_D0 = DFT_rx .* conj(DFT_reference); % outputs the dft of each subpulse

    DFT_D0_shifted = w.*fftshift(DFT_D0);

    % append zeros
DFT_D0_shifted = [zeros(1, n_zeros) zeros(1,2572) (DFT_D0_shifted) zeros(1, 75028488)];
    % freq shift
    DFT_D0_shifted = circshift(DFT_D0_shifted, 19367244 + round(delta_f));
    delta_f = delta_f+139000;
    count2 =n
    Dn1 = Dn1+DFT_D0_shifted;
    end
     toc
%% plotting
Dn1 = Dn1./max(Dn1);
% figure()
set(0,'defaultfigurecolor',[1 1 1])
plot(f_sim,20*log(abs(Dn1)))

title('Reconstructed SWW');xlabel('Frequency(Hz)');ylabel('Amplitude(dB)');
% xlim([7000000 3100000000])
ylim([-250 10])
grid on
hold on
```

```
y=fftshift(ifft(Dn1));
y = y/max(y);
y = y(:,[37673002:75346032]);

set(0,'defaultfigurecolor',[1 1 1])
figure()
plot(ttinch,((abs(y)))); title('A Scan'); xlabel('Distance(m)'); ylabel('Amplitude')
% xlim([90 104])
grid on
set(0,'defaultfigurecolor',[1 1 1])
figure()
plot(ttmeter,(abs(y))); title('A Scan'); xlabel('Distance(Meters)'); ylabel('Amplitude')
xlim([0 10])
ylim([0 1.1])
grid on

%% PLOT TO COMPARE GRATING LOBES

D2 = Dn1./max(Dn1);
AX = 20*log10((abs(fftshift(ifft(D2))))./max((abs(fftshift(ifft(D2))))));
AX2 = AX(1,[37670002:37700002]); % take middle 4k samples
AX2 = (circshift(AX2,9300));
LENGTH = length(AX2);
c=3e8;
fs3 = 15000e6;
period = LENGTH/fs3;
ttmeter2 = -((c*period)/2):c*period/(LENGTH-1):(c*period)/2;




set(0,'defaultfigurecolor',[1 1 1])
plot(ttmeter2, AX2)
xlim([-200 200])
grid on
title('SWW Autocorreclation')
xlabel('Range(m)')
ylabel('Amplitude(dB)')
```