Software Defined Radar

A Major Qualifying Project

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Shahil Kantesaria

Nathan Olivarez

Date: October 13, 2011

MIT Lincoln Laboratory Supervisor: Vito Mecca

Approved:

Professor Edward A. Clancy, Major Advisor

This work is sponsored by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and not necessarily endorsed by the United States Government.

Abstract

This paper describes the design and implementation of a software defined phased radar receiver array developed for MIT Lincoln Laboratory. This receiver array was constructed with the USRP2, an inexpensive software defined radio to allow for scalability to larger receiver arrays. The team worked closely with Lincoln Laboratory staff to conduct time synchronization testing between the receivers in the array in order to characterize the internal oscillator drift of the radios. The team was able to get the receiver array synchronized within less than 2 ns. In addition to building the array, the team implemented radar processing algorithms in Matlab which had the ability to detect a target's range, radial velocity and direction from the receiver with an angular resolution of 20 degrees.

Acknowledgements

We would like to extend our gratitude to all the individuals who contributed to the software defined radar project. Throughout the course of the project, many individuals took time out of their day to provide us support on various tasks.

We would especially like to thank the members of Group 33 who supported the design and implementation of the software defined radar project. Specifically, our project supervisor, Vito Mecca, and fellow engineers Matthew Morris, Kyle Pearson, Jeffrey McHarg, and James Montgomery for their contributions to the hardware and software design. We also owe great thanks to Walter Dicarlo and Robert Piccola for their technical support. We would also like to thank all the members of Group 33 who attended our weekly design meetings to provide us with advice and feedback.

We owe our thanks to our project advisor, Edward Clancy, for his guidance throughout the project. We appreciated his timely feedback, attendance of weekly meetings and help to organize the final written paper. Throughout the project, he kept the team on track to meet the project's goals.

Statement of Authorship

In this section we outline the various contributions to the project by each team member. This project has been completed in partial fulfillment of the requirements for the degree of Bachelors of Science in the field of Electrical and Computer Engineering.

Shahil Kantesaria
Primary Author: Executive Summary, Receive System, Transmitter, Hardware Methods, Conclusion
Testing: Time Synchronization Test, Direction Finding, Radar Field Test, Real-time GNU Radio Test
Performed Background Research
Researched and selected Time Synchronization Hardware
Assembled Mobile Receiver Array

Nathan Olivarez
Primary Author: Introduction, Transmit, Software, Software Methods, Radar Processing Methods
Testing: Time Synchronization Test, Direction Finding, Radar Field test
Installed and modified GNU Radio Software
Developed Radar Processing in Matlab

Executive Summary

Since their founding in 1951, a majority of the nation's radar technologies can be attributed to developments at MIT Lincoln Laboratories. Although the lab today conducts research in a wide array of fields, Group 33, this project's sponsor specifically focuses on radar research. One of the projects in development is Over the Horizon Radar (OTH). OTH radar systems use large antenna arrays to reflect signals off of the earth's atmosphere to view targets several thousands of kilometers away that are beyond line-of-sight range. Our project is of particular interest to Group 33 because it has the potential to create a new radar test bed that costs less than 15% of their existing receive array.

OTH radar arrays are very expensive because each antenna in the array requires its own radio. The purpose of this project is to develop the framework for an inexpensive phased receive radar array using commercial off-the-shelf software defined radios (SDR) with the capability to determine range, direction, and velocity of a target. A software defined radio is a radio system that implements components, such as modulators, filters, and amplifiers through software instead of hardware; for this reason, the SDR platform is very flexible. The first objective of the project was to test the synchronization of multiple Ettus USRP2 SDRs in a receiver array. The second major objective was to implement radar processing algorithms to process the data collected from the receiver array.

Radar, short for Radio Detection and Ranging, is a method of detecting targets using electromagnetic waves. Radar works by broadcasting a radio signal and receiving the reflections, or echoes. Processing the received signals using various algorithms can reveal the direction, range, and velocity of a target. The two most common types of radar systems are pulse and continuous wave (CW). Pulse radars alternate between transmit and receive modes; CW radars continuously and simultaneously transmit and receive. OTH radar systems are continuous wave because it allows for continuous high powered transmissions which are necessary to locate distant targets. Thus we developed a CW radar system.

In our system the transmitter broadcasts a repeating chirp, a narrowband frequency sweep. When the signal reaches a target, echoes are reflected towards the antenna array. Next, these reflections are filtered, sampled, and demodulated by each antenna's receiver, and finally sent to a computer for signal processing. The specific radio used in the project was the Ettus USRP2. The USRP2 has an analog RF front end filter, analog-to-digital converter and an FPGA for signal processing. A 100 MHz local oscillator circuit drives the internal ADC. A phase-locked loop can lock this oscillator to an external 10 MHz reference clock. Synchronization across the array is critical since time delay of arrival is used for radar

processing. A tolerable offset between each receiver is 5 ns, half the period of the 100 MHz ADC clock. Receivers with offsets greater than 5 ns risk missing samples which degrades system accuracy.

Our receive array consisted of four major function blocks – a GPS clock, signal amplifiers/splitters, software defined radios (SDR), and a signal source. The GPS (reference) clock was essential to our system because it served as a common time reference for all SDRs in our setup. It outputted a 10 MHz reference clock signal to train the USRP2s' 100 MHz local oscillators and a 1 pulse per second (PPS) signal to prevent phase drift. Signal amplifiers and splitters were used to evenly distribute these signals to the SDRs. The amplifiers used met the 5V peak-to-peak voltage requirements for the PPS signal and the 10 dBm power requirements for the reference clock.

To test the level of synchronization, a signal generator supplied a 23 MHz sinusoid test signal to four USRP2s. A Jackson Labs Fury GPS clock provided the 10 MHz reference and 1 PPS signals, which were amplified and distributed to four radios using a Pulse Research Labs (PRL) fanout 50Ω TTL line driver. The GPS clock, fanout driver and USRP2s were connected with equal length 2 FT SMA cables to reduce the risk of introducing phase offsets. Once all the connections were made, GNU Radio was configured to enable the clock debug pins on the motherboard. Two methods were used to characterize the synchronization. The first method used a signal generator to transmit a sinusoid to four radios; the time offset was computed by comparing the phase of four received signals. The second method used an oscilloscope to analyze each radio's ADC clock waveform.

An Agilent Technologies E4420B signal generator transmitted a 23 MHz sinusoid to four USRP2s. We selected 23 MHz because this is the same frequency allocated to Group 33 for radar transmissions. The receivers recorded the transmission for 60 seconds; a raw data file was saved and imported into Matlab processing. The phase difference was computed between radio 1 and each of the other three receivers.

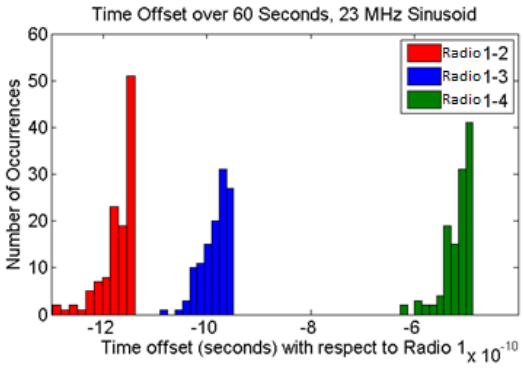


Figure 1: Time offsets for 4 radios over 60 seconds. Max time offset does not exceed 1.3 ns.

Each of the computed time offsets were calculated with respect to the reference, radio 1. The resulting histogram (Figure 1) shows the time offset of the three receivers with respect to the first. The maximum time offset is less than 1.3 ns, well within the 5 ns tolerance. Table 6 lists the specific results of the test.

Table 1: Time Offset Characteristics

	Data 1-2	Data 1-3	Data 1-4
Mean	-1.172 ns	-0.9841 ns	-0.5172 ns
Standard Deviation	0.03159 ns	0.0263 ns	0.0272 ns
Minimum	-1.139 ns	-0.9483 ns	-0.4871 ns
Maximum	-1.297 ns	-1.090 ns	-0.6278 ns
Spread	0.1590 ns	0.1412 ns	0.1407 ns

The purpose of the second experiment was to determine whether the individual clocks would drift away from the reference after several hours. Using an Agilent Infiniium oscilloscope we connected three probes to ADC clock debug pins on three USRP2s. The fourth probe was connected to the 1 PPS output which served as the trigger. Figure 2 shows the persistence plot after running for 30 minutes. The three ADC clock waveforms are centered on the rising edge of the 1 PPS signal; the spread of the waveforms indicates the maximum drift that occurred during the test. As shown, the clocks did not exceed 2 ns.

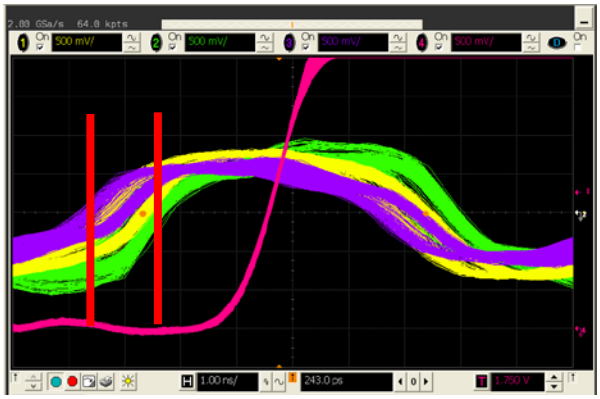


Figure 2: Oscilloscope Test. The 1 PPS trigger is shown as the pink trace; the other three channels (purple, yellow, and green traces) are connected to three USRP2s. After running for 30 minutes, the maximum time offset was less than 2 ns.

All of the radar processing methods for this project's receive system were conducted in Matlab. The data from six radios was imported into Matlab to create a six channel array. The radar processing in Matlab was broken into three steps: range processing, Doppler processing, and direction finding.

Range was computed by determining the time delay between the transmitted and received signals. This time delay was calculated by correlating the transmit chirp (linear frequency sweep) with the received signal. The correlation produced a peak in the received signal where the target existed. To determine a

target's speed, we determined the Doppler shift frequency which is the phase progression from sweep to sweep. The phase between chirps changes periodically with frequency *f*. This frequency can be found by taking the Fourier transform across each range cell.

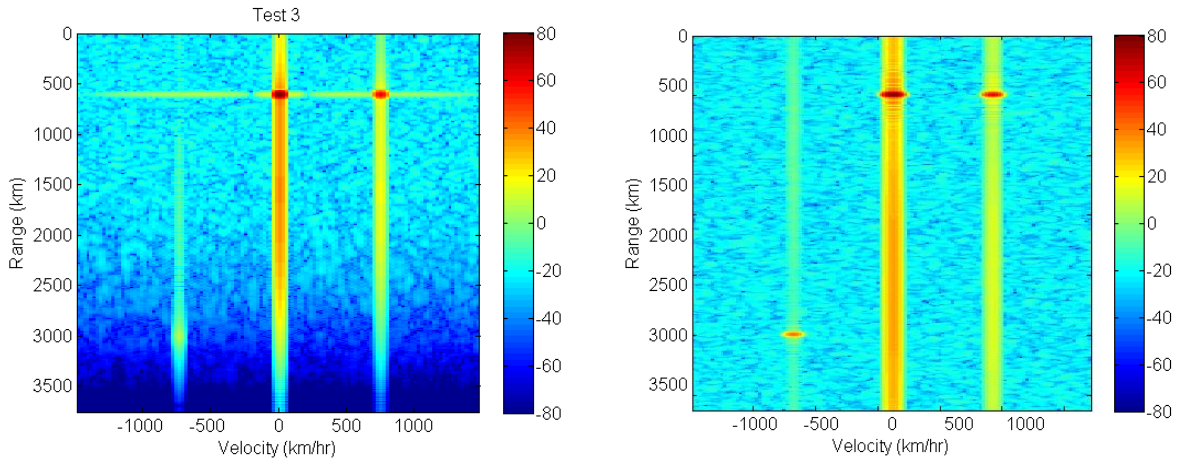


Figure 3: Simulated data test 3 - multi target. The left figure was created using our processing scripts and the right figure was generated by our sponsor's. This test shows three targets. There are two targets at 600 km; one is moving toward the receiver at 750 km/hr and the other is stationary. There is also a distant target at 3000 km moving away from the receiver at 750 km/hr. Left plot shows the Range-Doppler plot created using our radar processing scripts.

Because the Fourier transform of a periodic function generates a delta function, δ(F), at each harmonic, a peak occurs at the range and Doppler cell which corresponds to the target's range and speed, respectively, as shown in Figure 3. The plot on the left was created using our processing scripts while the plot on the right was generated by our sponsor's processing scripts; both used the same simulated data set.

The direction of a target is identified by comparing the phase difference between data points collected from each antenna. The incident angle of the transmission, θ, can be related to the instantaneous signal phase, ϕ(*n*), by

$$\Phi(n) = \frac{2\pi}{\lambda} d * (n-1) \sin\theta \, , n = 1,2, \dots, 6$$

where *n* is the antenna number (1-6), λ is the carrier wavelength (13 m in the case of our radar), and *d* is the spacing between antennas (6.5 m in the case of our system), assuming the antennas are linearly arranged and equally spaced. This equation assumes the transmitter is in the receiver array's far field and is not valid for close targets. The phase is computed for six complex points, each provided by a different channel. The incident angle theta that maximizes the sum of these points represents an optimal estimate of the transmission's incident angle.

To test the direction processing, we used Group 33's mobile transmit unit, a truck with a transmitter. We recorded the transmission as the truck drove in front of our array. The path of the truck and the direction plot over a three second period is shown in Figure 4. The peak of each waveform indicates the incident angle of the received signal; the angle at each point in time corresponds to the movement of the truck. Despite an approximate 10 degree error, our results show that our system can identify a moving target and determine its direction.

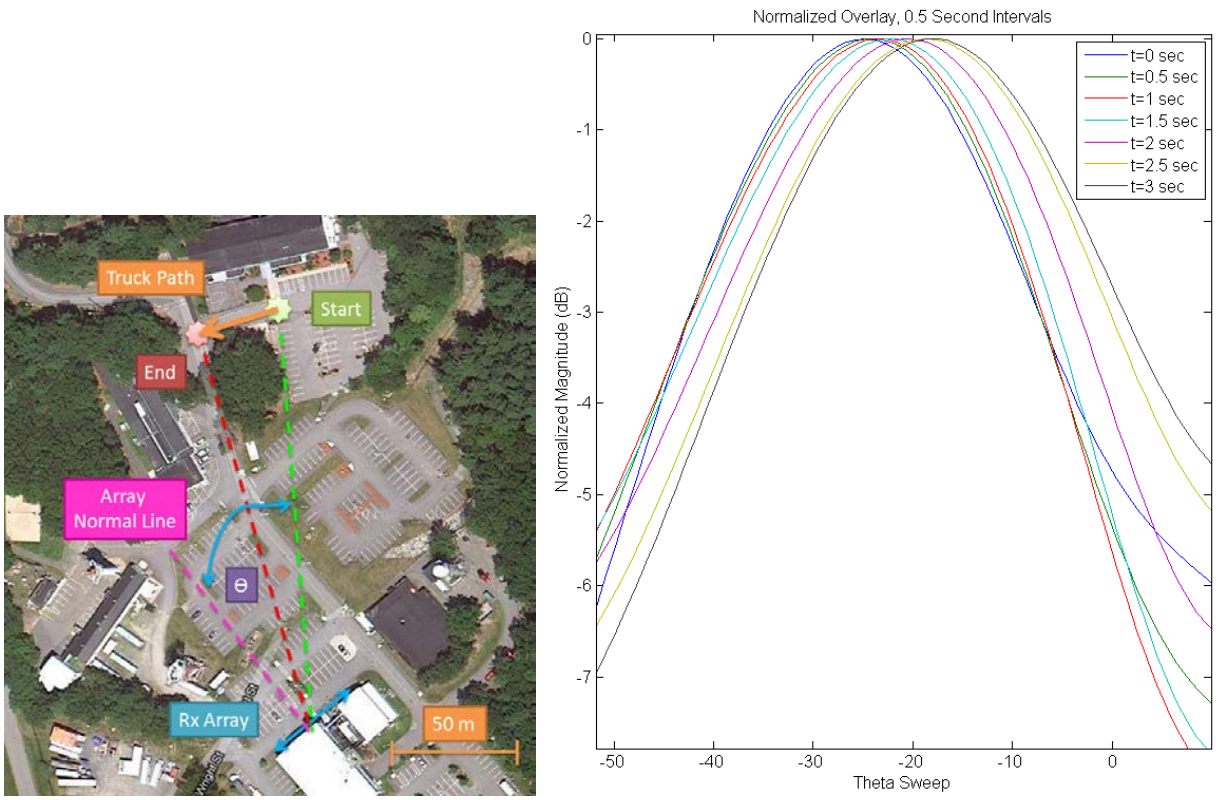


Figure 4: Left: Mobile transmitter path. Right: Direction plot of the transmitter when driving in front of the array.

The experiments conducted during this project characterized the time synchronization specifications across multiple USRP2s. The USRP2 was then used to build a phased receive array and collect data used to determine the range, velocity, and direction of a target. This project shows that an inexpensive phased receive array can be developed using the USRP2.

Table of Contents

1 Introduction

The purpose of this project was to develop the framework for an inexpensive phased receive radar array using software defined radios. A software defined radio uses software and programmable logic to replace components typically implemented using hardware, such as demodulators and filters. The objective was to assess timing synchronization between software defined radios in an array and to deliver a working prototype that could determine the direction, range, and movement of a transmission during post-processing.

1.1 Sponsor

Lincoln Laboratory was originally founded after World War II for the purpose of radar research and development. Since its founding in 1951, much of the nation's radar technology can be attributed to developments made at Lincoln Laboratory. Although the lab today conducts research in a wide array of fields, this project's sponsor, Group 33, specifically focuses on radar. In the recent decade, Group 33 has been working on the Over the Horizon Radar project, an initiative to use large antenna arrays to reflect signals off the earth's atmosphere to view targets several thousands of kilometers away that are out of line-of-sight range. Our project is of particular interest to Group 33 because it has the potential to create a new radar test bed that costs less than 15% of their existing receive array. If successful, the lab will have the opportunity to build larger, more accurate arrays for a fraction of the cost.

1.2 Scope

The students working on this project learned about multistatic radar and gained an understanding of various radar processing algorithms used with phased receive arrays, specifically direction finding, range processing, and Doppler processing. The first task of the project was to learn about radar theory including the math and algorithms associated with radar processing. The next task was to construct the phased receive array using the Ettus Research Universal Software Radio Peripheral (USRP2). The USRP2 is a reprogrammable transceiver (transmitter and receiver) that implements components such as filters and demodulators on a field programmable gate array (FPGA). The USRP2 is an open source model which can be reprogrammed using GNU Radio, a Linux program designed to rewrite the FPGA. Third, a method was devised to calibrate and time synchronize each receiver in the system. Because much of the radar processing was achieved by comparing the phase differences between signals received at individual antennas in the array, time synchronization between the receivers was one of the most important goals. Additionally, calibration was necessary to account for uncontrollable factors such as

uneven antenna gain and irregular cable lengths. Finally, processing code was written in Matlab that could determine the range, velocity, and direction of a target.

The following is a list of the project deliverables:

- Timing: Each of the radios were synchronized to within 5 ns of each other. The USRP2 ADC (analog to digital converter) sampled at 100 MHz, or once every 10 ns. If the delay was greater than half the period (5 ns), the receivers could drift by an entire sample period. Small time offsets were necessary to accurately determine the location of a target. The timing requirements will be further explained in the time synchronization testing section.
- Calibration: A Matlab script identified phase offsets that were out of the user's control (e.g. offsets due to varying cable lengths or materials, differing clock oscillators, antenna differences, *etc.*).
- Data processing: The final program recorded the signals received from each antenna in real time and saved it to a file. This file was accessed later during post-processing which was not accomplished in real time. The off-line program provided graphical means of displaying the range, direction, and movement of a target.
- Direction finding: The receive array detected the azimuth of a target with an error not exceeding 30 degrees. This value was derived from the angular resolution of an array with six equally spaced receivers, the number of USRP2s available for the project.
- Range finding: The project's sponsor, Group 33, has been allocated a 10 kHz bandwidth in the High Frequency (HF: 3-30 MHz) range. Due to this narrow bandwidth constraint for our transmitted signal, our range was only accurate to within several kilometers. Since our accuracy cannot be tested within the confines of Hanscom Air Force Base (not more than 10 km wide), simulated data was used to verify certain aspects of the program's functionality.

2 Background Introduction to Radar

Radar, short for Radio Detection and Ranging, is a method of detecting targets using electromagnetic waves. Radar works by broadcasting a radio signal and receiving the reflected echoes. Processing these reflections using various algorithms can reveal the direction of the target relative to the receiver, the range of the target, and its speed. Radar is used to track planes at busy airports, monitor hurricanes in the Gulf of Mexico, and to catch speeders on the Massachusetts Turnpike. These are each different applications but the underlying radar components and principles are all the same.

The four common components to all radar systems are a transmitter, a receiver, a signal processor, and a data output. A high level flow diagram is shown in Figure 5.

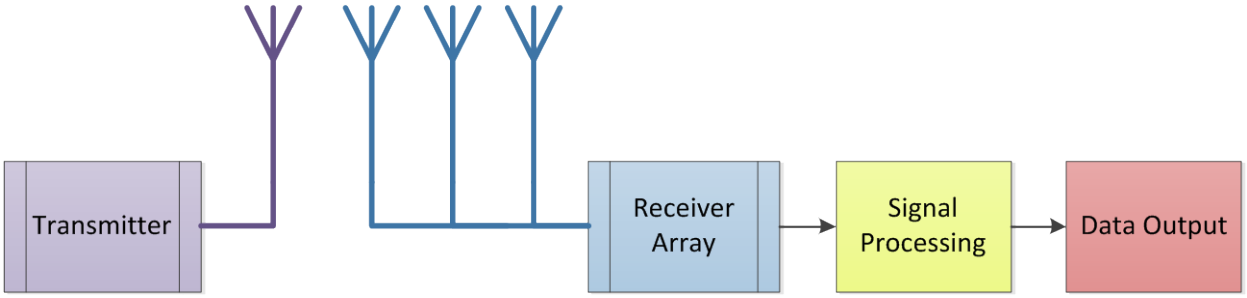


Figure 5: Radar component model showing the 4 main blocks: transmitter, receiver, signal processor, and data output. This particular models shows a multistatic radar configuration, the type of radar used in this project which will be explained in the following section.

The transmitter broadcasts a pulse during the transmission cycle of the radar system. The frequency of this cycle is based on the particular radar configuration. The waveform propagates until it reaches a target; some of the signal energy is reflected back towards the radar. This echo is detected by the receiver and then sent to the signal processer. The signal processor filters, demodulates, and performs various algorithms to calculate angle, range and movement of a target. After the processing is complete the results are sent to a graphical display or some other data output.

2.1 Radar Configurations

There are three common radar configurations for arranging the transmitter and receiver, as shown in Figure 6. A monostatic radar system, also known as a pulse radar, utilizes the same antenna for both transmitting and receiving. Pulse radars do not transmit and receive at the same time but instead rapidly alternate between the two states. Once the pulse is sent, the radar switches to receive mode to detect

the reflected signal. These pulse radars are most commonly used for short ranges (up to one hundred kilometers), such as the rotating airport surveillance radar shown in Figure 7. Bistatic and multistatic systems have distinct transmit and receive antennas at separated locations. Bistatic systems have singular receive and transmit antennas whereas multistatic systems can have multiple receive and transmit antennas. Bistatic and multistatic configurations are often continuous wave, including the Over the Horizon Radar in development at Lincoln Labs. In continuous wave radar, the transmitter and receiver run simultaneously. If the transmit and receive antennas were coincident, the transmitter's high power output would damage the receiver hardware and overpower any received signals.

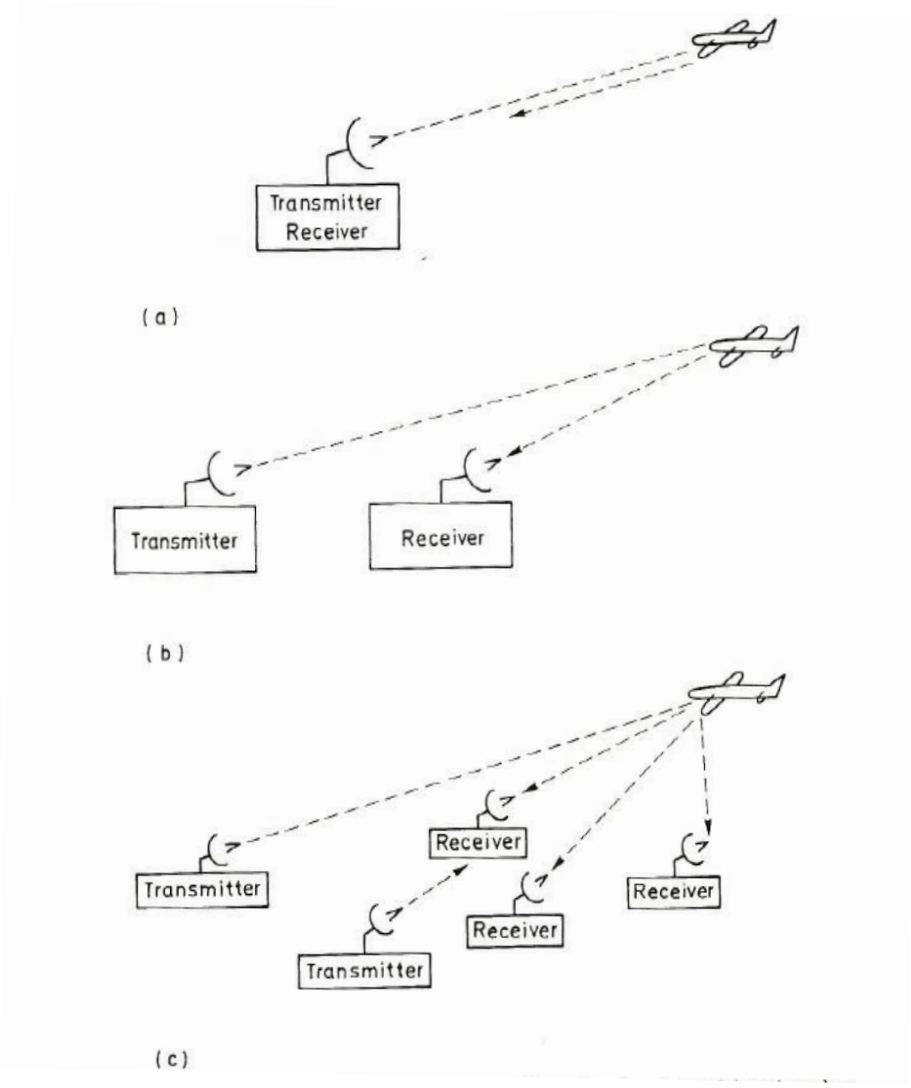


Figure 6: (a) Monostatic radar consists of a single transmitter and receiver at the same location. (b) Bistatic radar consists of a single transmitter and receiver but at different locations. (c) Multistatic radar consists of several transmitters and receivers at different locations, often in an array. (Hall)

Figure 7: Airport Surveillance Radar. Rotating dish radar with fixed aperture. (Wikipedia)

2.1.1 Pulse Radar

Monostatic pulse radar systems have all of their components, the transmitter, the receiver, the signal processing, and the data output, centralized at the same location, as shown in Figure 8. First, the transmitter sends a pulse to the antenna; the power of this pulse varies with the application. For example, an airport surveillance radar with a range of about 50-60 nautical miles has a peak power of about 1.3 MW while a radar gun used by law enforcement officers has peak power of about 20 mW (Skolnik). When transmitting, the transmit/receive switch disconnects the receiver from the antenna. The pulse propagates through free space until it reaches an object such as an aircraft. Some of the electromagnetic waves are reflected back to the antenna. At this time, the transmitter is no longer active because the switch has disconnected it and connected the receiver to the antenna to begin detecting reflected signals. These reflections shown in Figure 9 are acquired and sent to the signal processor to filter out background noise, determine the range, and identify any Doppler shifts that could indicate movement. Finally, processed data are directed to an output device such as a terminal display or a data file. This process then repeats several hundreds or thousands of times per second.

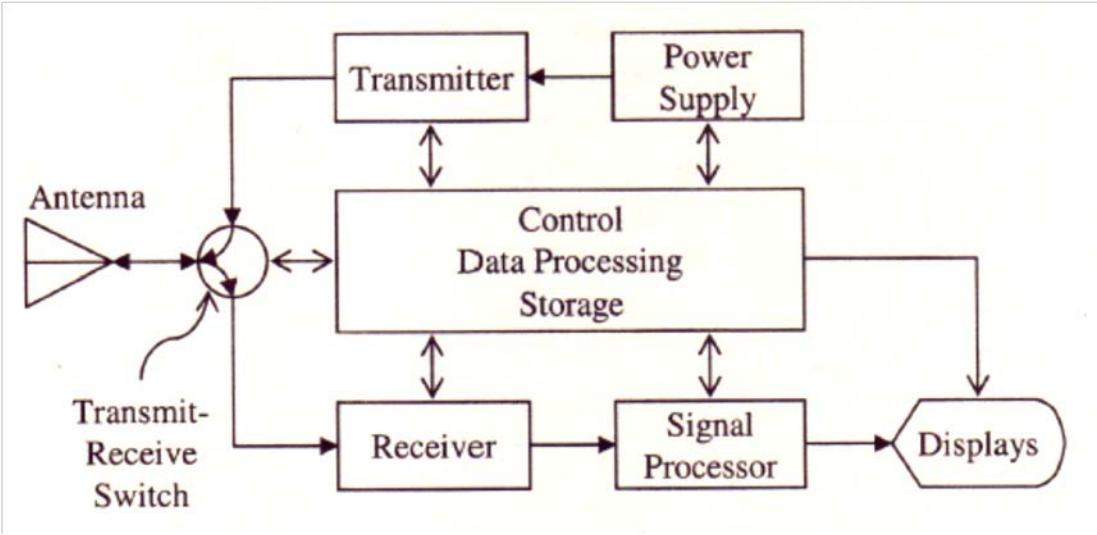


Figure 8: Pulse radar block diagram. The six different components that comprise a pulse radar setup are: control logic, power supply, transmitter, receiver, signal processor, and displays. (Skolnik)

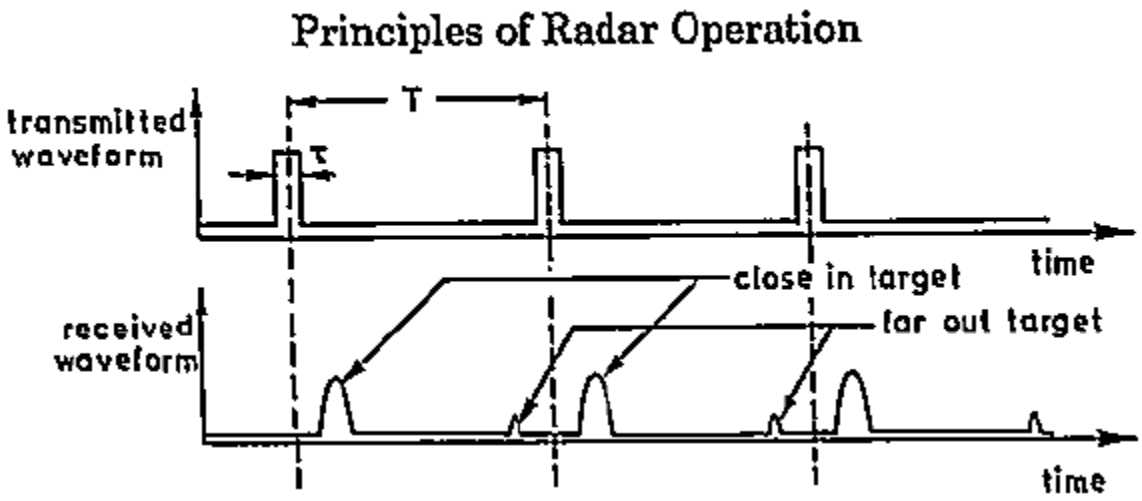


Figure 9: Pulse radar signal timeline. First, a pulse is transmitted, as shown in the top waveform. Once complete, the radar switches to receive mode. The receive time, determined by the pulse repetition frequency (PRF), varies per application. For example, long receive times (low PRF) are necessary to view distant targets. If the receive time was shorter in this example, the distant targets would return as ambiguous responses as described in the following section. (Hall)

In a pulse radar system, it important to receive target echoes before transmitting the next pulse. As shown in Figure 10, if an echo from the first transmitted pulse is received after the second transmitted pulse, this echo is an ambiguous response because the pulse it is returning from cannot be determined. Ambiguous returns occur when the time taken for the reflected signal to arrive is longer than the time between pulses. Thus, the maximum unambiguous range of a pulse radar is determined by the pulse

period, also known as the pulse repetition frequency (PRF). Note that because the radar is receiving when it is not transmitting, the PRF also defines the receive time. The maximum unambiguous range, *R,* can be determined by

$$R = c * \left(\frac{t_{\mathrm{Rx}}}{2}\right)$$

where *c* is the speed of the electromagnetic signal and t_{RX} is the receive time.

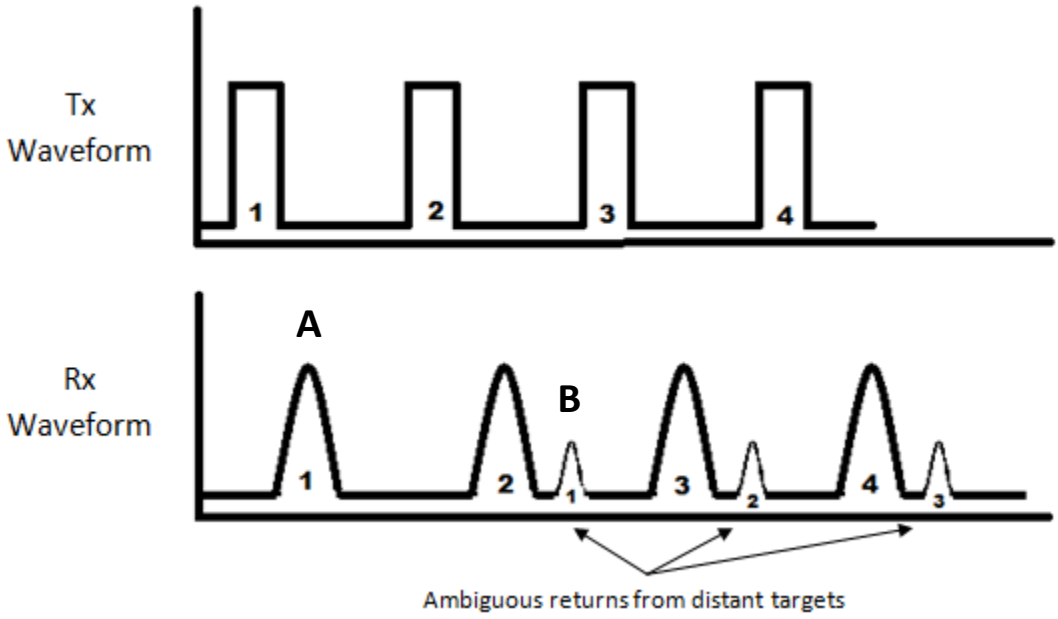


Figure 10: Pulse radar signal timeline depicting ambiguous returns. This timeline shows the transmit pulse (top) and its reflections (bottom). Echo 'A' is received before Pulse 2 is sent, thus it is a reflection of Pulse 1. Echo B appears to be a reflection of Pulse 2 since it arrives in the receive window between Pulse 2 and Pulse 3. However, it is actually a distant target beyond the unambiguous range reflecting Pulse 1.

2.1.2 Continuous Wave Radar

In continuous wave radar, there are no pulses. Instead, the transmitter and receiver are continuously running simultaneously. Unlike a pulse radar which outputs strong periodic bursts of power on the order of megawatts, continuous wave radar has a constant power output often on the order of kilowatts. Although pulse radar requires high power for short periods of time and continuous wave requires low power for long periods, the average power output for the two systems is about the same. The radar application usually dictates which method is used. Continuous wave radar is best suited for long range surveillance over several thousand kilometers. To achieve the same range as continuous wave, a pulse

radar system would require additional components to help shield the sensitive receivers and insulation to prevent arcing on the antennas. As a tradeoff to lower power consumption, however, continuous wave radar requires large receive arrays which can be spaced across hundreds or even thousands of meters.

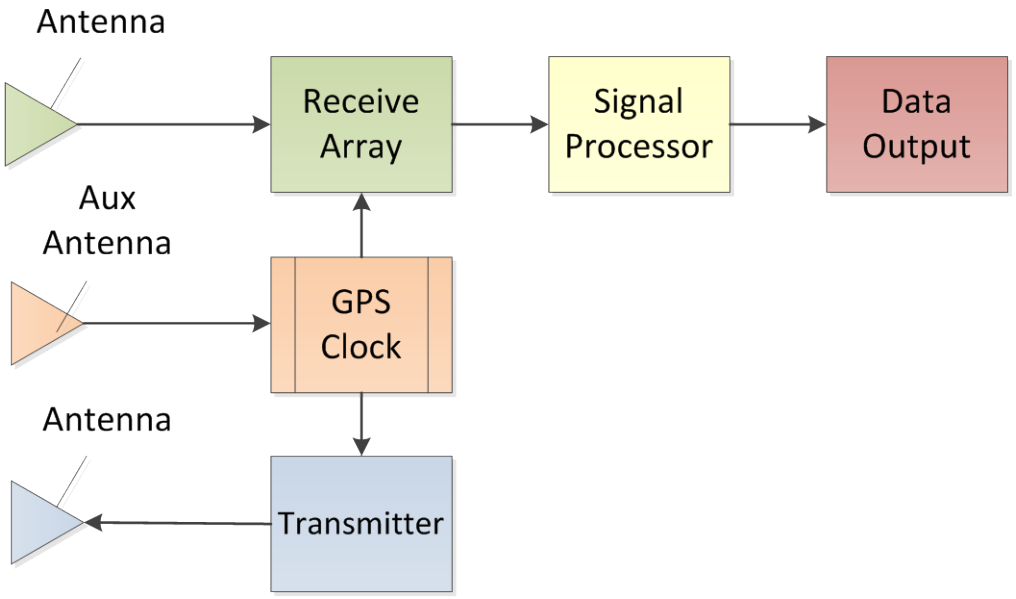


Figure 11: Continuous wave radar block diagram

The block diagram of a continuous wave radar is shown in Figure 11. Note that it shares many of the same components as the pulse radar; the two primary differences are the additional antennas and the GPS clock. Both the transmitter and the receive array have their own antennas because they are at different physical locations. Quite often the transmitter and receiver are several kilometers away which can pose a problem for synchronization between the two. As will be discussed in Section 2.2, synchronization is necessary for range processing. To solve this problem, the transmitter and receiver can use an external reference clock provided by global positioning system (GPS) satellites. Because all of the GPS satellites are synchronized, GPS receivers connected to the transmitter and receiver keeps them synchronized no matter how far they are separated.

2.2 Continuous Wave Radar: Specific to this Project

Recall that the purpose of this project was to develop the framework for a phased receive array for a continuous wave, multistatic radar system. This section presents each of the components of the radar system and concentrates on the receiver hardware and software. Figure 12 provides an overview of the components of the system.

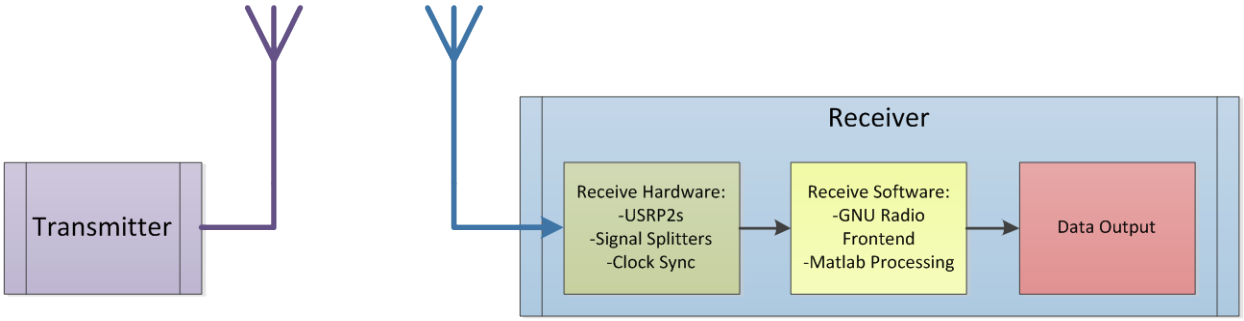


Figure 12: Continuous wave radar component overview. Transmitter continuously broadcasts a chirp; receiver hardware samples the reflected chirp through multiple USRP2 synchronized radios. Reflected chirps are stored into files by the GNU Radio frontend and later processed in Matlab, finally the results are displayed on Matlab plots.

First, the transmitter broadcasts a chirp, a narrowband frequency sweep similar to a pulse (see Section 2.2.1: Transmit). When the signal reaches a target, echoes are reflected towards the antenna array at the receiving end. Next, these reflections are filtered, sampled, demodulated, and down sampled by the radio array, and finally sent to a computer for signal processing. Each of the components mentioned in Figure 12 will be explained in the following sections.

2.2.1 Transmit

A single HF waveform generator is used for this multistatic radar system. It continuously broadcasts a chirp, a sweep across several frequencies as shown in Figure 13. Chirps have a bandwidth *F* and repeat with period *T*. In the time domain, the baseband chirp is represented by a frequency varying complex exponential, as shown in Figure 14. The chirp is defined by

$$s(t) = e^{j\frac{F}{2T}t^2}$$

where *F* is the swept bandwidth (in Hertz), *T* is the duration of a single chirp (in seconds), and *t* is the time vector defined by [-*T*/2 : *T*/2] (for a single period of a chirp).

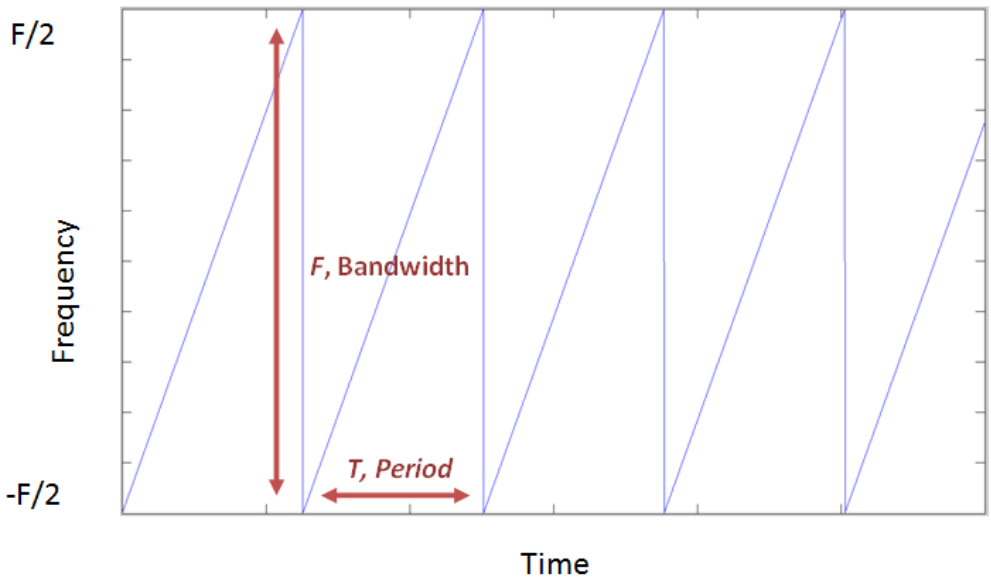


Figure 13: Repeating chirp shown on a frequency vs. time plot. Radar chirps are frequency sweeps with bandwidth *F* and repeat with period *T*.

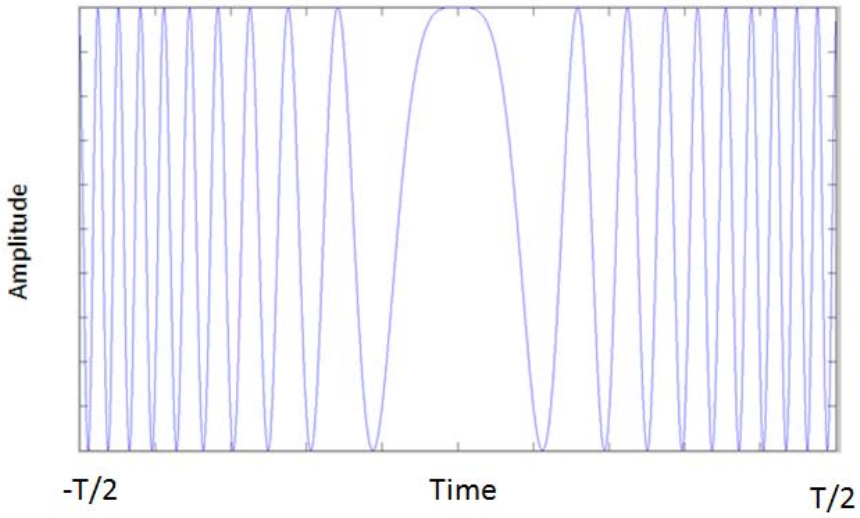


Figure 14: Real component of the radar chirp. In the time domain the chirp is modeled by a complex exponential with frequency that varies with time.

When transmitted, the chirp is modulated to a carrier frequency using quadrature multiplexing, as shown in Figure 15. Quadrature multiplexing is used to transmit the in phase and quadrature components of the radar chirp using orthogonal carriers. The in phase component, Re{*s(t)*}, as denoted

by $m_1(t)$ is modulated by a cosine with carrier frequency f_c. The quadrature component, Im{$s(t)$}, is modulated by a sine with the same carrier frequency, f_c. The two components are summed and transmitted.

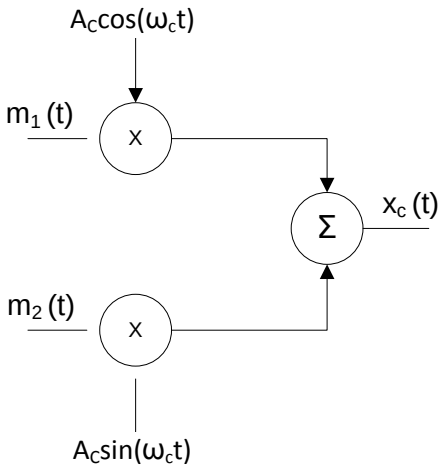


Figure 15: Quadrature multiplexing. The in phase component of the chirp (real component) is represented by $m_1(t)$. The quadrature component of the chirp (imaginary component) is represented by $m_2(t)$. (Ziemer)

Because the chirps are continuously broadcasted, the period must be long enough that the reflected signal return before the next chirp begins (Hall). The number of chirps transmitted per second is known as the waveform repetition frequency, or WRF. The principle which dictates the PRF in a pulse radar, as described in Section 2.1.1: Pulse Radar, also determines the WRF of a continuous wave radar.

For this project, the chirp was transmitted at 23 MHz with a bandwidth of 10 kHz because it was the largest spectrum that Group 33 had authorization by the FCC to use for transmissions. Thus, Group 33 chose to use this spectrum for the project because larger bandwidths increase the resolution of the receiver; this will be described in greater detail in the signal processing section.

2.2.2 Receive System

The receiving system is composed of receive hardware, receive software, and the data output as shown in Figure 16. The hardware component includes the array of USRP2 radios and their corresponding antennas, the software block includes the GNU Radio frontend and Matlab processing scripts; Range-Doppler plots and direction finding plots provide a graphical data output. The following sections describe the details of each block.

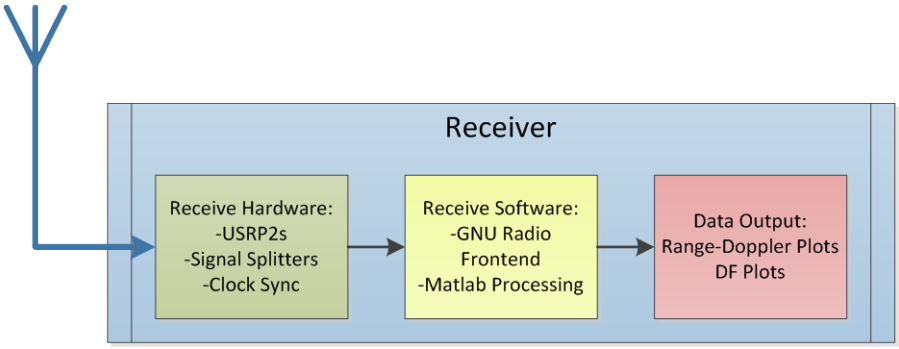


Figure 16: Block diagram of the radar system that was built for this project. The receiver is divided into a hardware component, a software component, and a data output block.

2.2.2.1 Hardware

The receiver hardware is the first module of the receive system. It is responsible for collecting chirp echoes reflected from a target, sampling the signal, demodulation, and exporting this information to the signal processing block. The receiver hardware includes the antenna array, the USRP2 receivers (one per antenna), a reference clock source (Global Positioning System Disciplined Oscillator – GPSDO), and all of the connections between these components. Figure 17 shows a block diagram of how these components are connected.

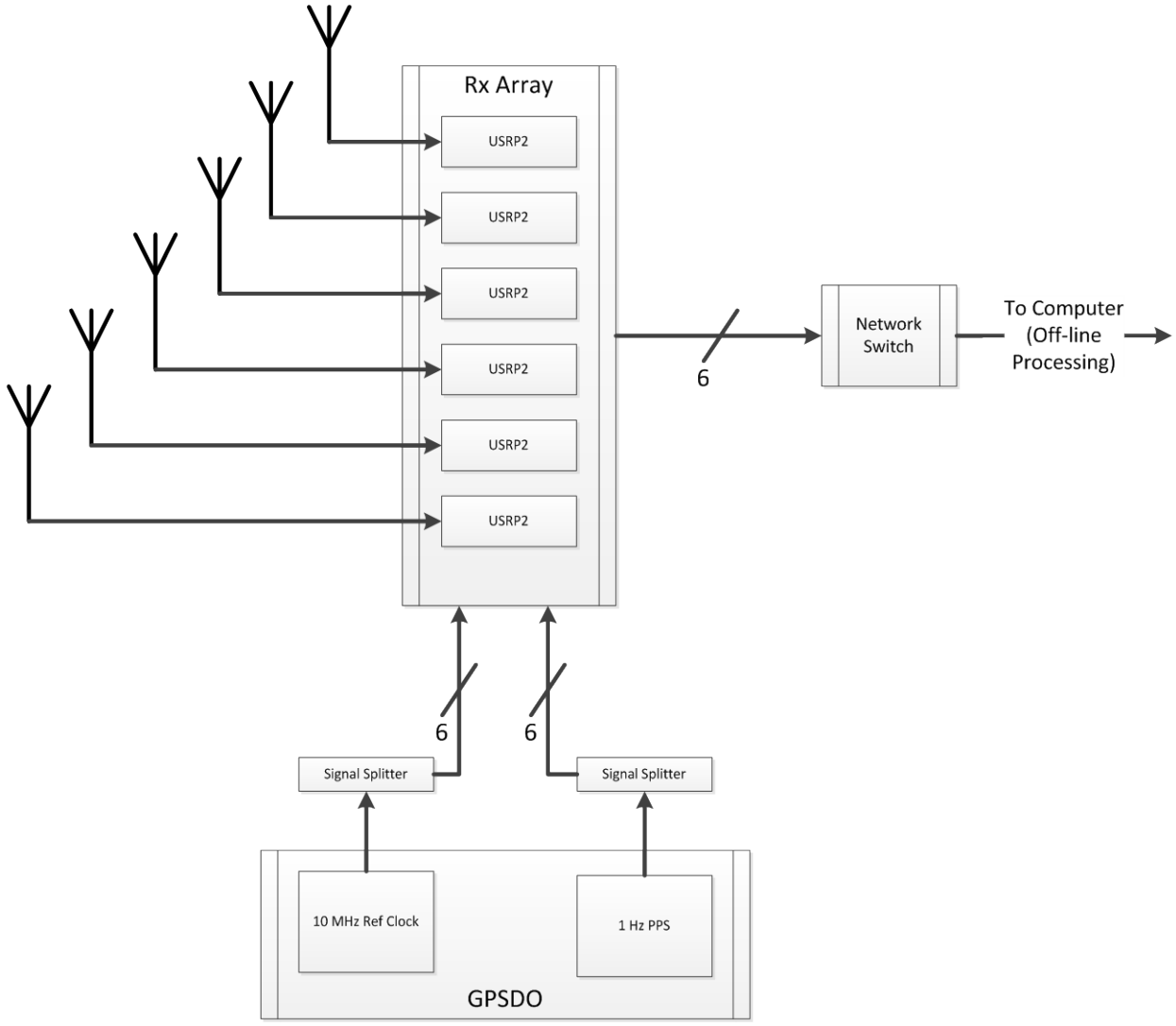


Figure 17: Receiver hardware flow diagram. The receive array consists of six USRP2s each connected to their own antenna. The GPS Dedicated Oscillator (GPSDO) provides a 10 MHz reference clock and 1 pulse per second (PPS) signal which are each split six ways to each of the receivers. Last, each USRP2 is connected to a host computer via a gigabit Ethernet connection and network switch.

2.2.2.1.1 Receive Antennas

Although antenna design is beyond the scope of the project, some general information regarding the antennas used in this project is necessary to explain our methodology. Important antenna characteristics pertinent to this project include the antenna type, length, shape of the antenna array, and the spacing between antennas within the array.

The receive antenna array consisted of monopole antennas. The length of each antenna needs to be an integer multiple of λ/4 where λ is the wavelength of the signal being acquired. As the frequency

increases, the antenna length decreases and vice versa. The wavelength of a 23 MHz signal is about 13 m, so the antennas we used for the array were about 3.25 m long, approximately λ/4.

Our antennas were arranged in a one dimensional linear array. As shown in Figure 18, the antennas were equally spaced in a single line and each receiver used its own antenna. It is possible to create a two dimensional array (i.e. a 2 x 3 element array) however we chose a one dimensional arrangement in order to maximize the array length because longer arrays have improved angular resolution. Angular resolution can be determined by computing the half power angle, $\theta_{-3\mathrm{dB}} = \sin^{-1}\frac{0.88\lambda}{L}$ where λ is the wavelength of the carrier signal and *L* is the total length of the array. The half power angle is the angular distance from a target where the power density is reduced by a factor of 2, or -3dB (Wirth). Table 2 shows the relationship between the array length and the half power angle at 23 MHz, one of Group 33's allocated frequencies. One consequence of using a linear array is that our direction finding capabilities are limited to a single dimension. In other words, this array can only determine the azimuth of a target, not the angle of elevation. Two dimensional arrays can observe both azimuth and elevation.

+	+	+	+	+	+	**(+ denotes antenna)**

Figure 18: Antenna arrangement. N antennas are equally spaced in a line.

Table 2: Half power angle at 23 MHz (λ/2 = 6.52 m)

Array Length (m)	Antenna Spacing for 6 antennas (m)	$\theta_{\text{-3dB}}$ (degrees)
20 m	4 m	35.02°
25 m	5 m	27.33°
30 m	6 m	22.50°
35 m	7 m	19.14° (Aliasing could occur)
40 m	8 m	16.68° (Aliasing could occur)
>40 m	n/a	Aliasing could occur

Antenna spacing is closely related to digital sampling. Each antenna in an array samples the incoming waveform, so the spacing between the antennas determines the spatial sampling rate. Recall that an analog to digital converter (ADC) must sample a signal at twice its highest frequency, also known as the Nyquist frequency, in order to prevent aliasing. Similarly, the antenna nodes cannot be separated by more than half the wavelength (λ/2) of the incoming signal.

2.2.2.1.2 Radios

Software defined radios were responsible for collecting chirp echoes reflected from a target. A software defined radio, or SDR, is a radio system whose components such as modulators/demodulators, filters, and amplifiers are controlled by software implementation instead of hardware. A basic SDR consists of an analog RF front end for filtering, an analog-to-digital converter (ADC) to sample the signal, and then an FPGA to implement a mixer to down sample the signal and perform further signal processing. The specific SDR model that was used for this project was the Universal Software Radio Peripheral, or USRP2. This radio is a second generation USRP created by Ettus Research.

The Ettus Research USRP2 SDR was chosen for this project for multiple reasons: open source software, second generation hardware design, and inexpensive cost of ownership. The USRP2 uses GNU Radio as its controlling software which is well developed and maintained by the Linux community. One of the most important reasons this radio was used is because of its price. It is one of the least expensive software defined radios available and costs less than $2000; in comparison, the British Aerospace Engineering (BAE) receivers Group 33 currently uses cost about $15,000 each.

The USRP2 samples the chirp echoes through a monopole antenna as discussed in the previous section. The RF signal from the monopole antenna travels through a series of components in the USRP2 for processing. The breakdown of the internal components of the USRP2 is shown in Figure 19.

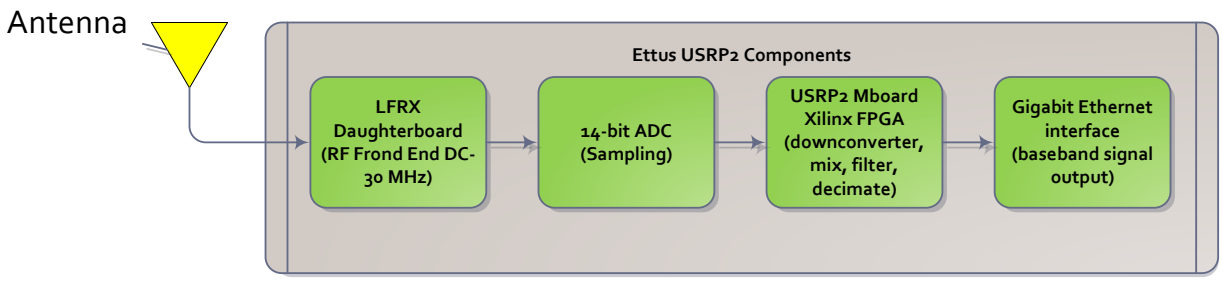


Figure 19: USRP2 internal component flow diagram. The USRP2 SDR consists of four major functional components: RF front end daughterboard, ADC, the motherboard and FPGA, and gigabit Ethernet interface.

The USRP2 is made up of four main functional blocks: RF front end, ADC, FPGA mixer, and a gigabit Ethernet interface. The first block is the RF front end daughterboard which low pass filters the incoming signal before it is sampled through a high speed ADC (Second Block). The third functional block is the motherboard of the radio which digitally down converts the digital signal from HF to baseband using quadrature demodulation. As shown in Figure 20, quadrature demodulation uses a sine and cosine

(orthogonal signals) to separate the received signal into its respective in phase and quadrature components. The USRP2 then combines I and Q components into a complex signal which is sent through the gigabit Ethernet interface (fourth block). Multiple USRP2 radios can be connected together to form a fully coherent multiple receive array. An internal 100 MHz local oscillator can be locked to an external reference clock and a 1 pulse per second (1 PPS) input via a phased-locked loop for precise timing applications (Ettus).

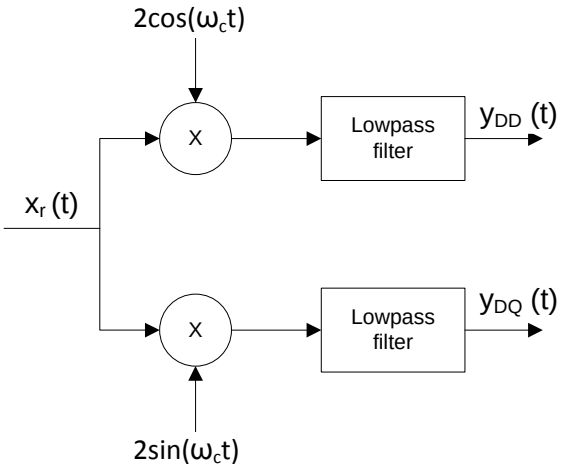


Figure 20: Quadrature demodulation. The received signal, $x_r(t)$ consists of both in phase and quadrature components. They can be separated through demodulation with a cosine and a sine because they are orthogonal. The USRP2 then recombines the I and Q components into a complex signal. (Ziemer)

There are several daughterboard models available for the USRP2, each with a different frequency range. As mentioned earlier our radar operates in the HF band (3-30 MHz), so we selected the LFRX daughterboard because it is compatible with a 23 MHz signal. The LFRX is composed of differential amplifiers and a 30 MHz low pass filter to prevent aliasing; it serves as an analog RF front end to the ADC.

The USRP2 has two 100 MS/s 14-bit analog-to-digital converters; the LFRX daughterboard only uses one of the 14-bit ADC's. The ADC is driven by an internal 100 MHz local oscillator which can be trained by an external reference source through a phased locked loop. At this stage in the hardware our 23 MHz input signal is sampled at 100 MHz. The signal is digitized and sent to the FPGA for further signal processing.

The USRP2 motherboard is made up of a Xilinx Spartan 3-2000 FPGA which implements two digital down converters (DDC) that mix, filter and decimate the 100 MS/s incoming signal into in phase and quadrature components, respectively. The DDC demodulates the 100 MS/s digitized signal from the ADC

to baseband. The DDC is implemented with four stages of cascaded integrator-comb (CIC) filters. These CIC filters are high performance filters because they use only additions and delays. For filtering out of band signals a 31 tap half band low-pass filter is cascaded with the output from the CIC filter to make up the DDC. The baseband signal is resampled at 200 KHz (fixed) before it is sent out through the Ethernet interface.

2.2.2.1.3 GPSDO and Time Synchronization

Building a continuous wave radar phased receive array required multiple USRP2 SDRs sampling multiple antennas at the same time. As mentioned earlier, the USRP2 has the ability to train the 100 MHz local oscillator to an external reference. By training the local oscillators in all six radios using the same external reference it was possible to achieve sampling synchronization in the receiver array. Each USRP2 has an external 10 MHz reference clock input and a one pulse per second input, where the reference clock trains the local oscillators through an internal phase locked loop in each USRP2. The 1 PPS signal was used to ensure the phase did not drift over time. To produce a common 10 MHz reference clock and 1 PPS signal a GPS disciplined oscillator was used. A GPS disciplined oscillator is a clock that uses GPS satellites to provide timing information to control the internal oscillator.

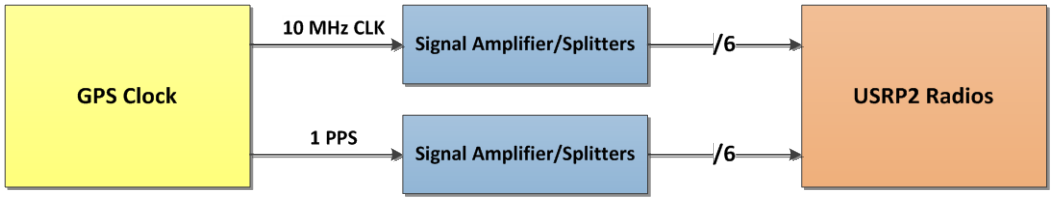


Figure 21: Time Synchronization High Level Setup

The USRP2 has specific power requirements for the external reference clock and PPS inputs listed in Table 3. To meet these specifications signal amplifiers and splitters were used as shown in Figure 21. The signal amplifiers kept all of the outputs in phase with each other to minimize adding external phase offsets to the system. To address phase offsets caused by cabling and connectors, equal length cabling was required with the same number of adaptors on each cable. Specific configurations are discussed in Section 3.1.2.1: Hardware.

Table 3: USRP2 Reference Clock Specifications	
10 MHz Ref. Clock Power	5-15 dBm
1 PPS Voltage	5V Peak-to-peak

2.2.2.2 *Software*

The receiver software module includes the GNU Radio frontend, Matlab radar processing scripts, and the data output; it is responsible for retrieving data from the USRP2 and using various radar processing techniques to determine and display a target's range, motion, and direction. Figure 22 shows the flow diagram for the receiver software components.

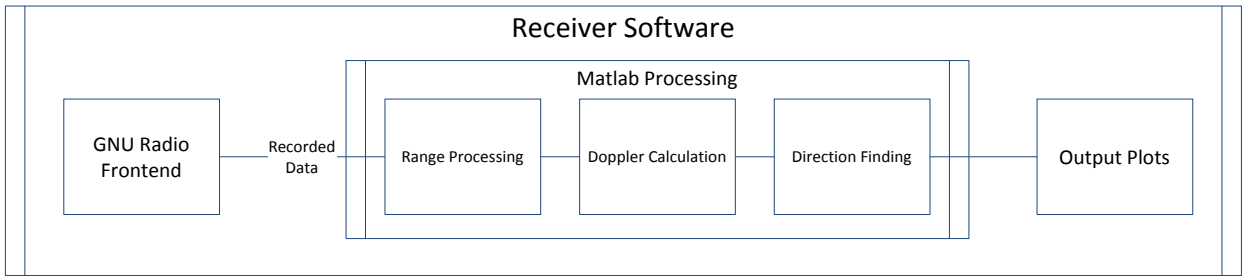


Figure 22: Receiver software module flow diagram.

The first component is GNU Radio, an open source interface used to control the USRP2 and acquire its received data. As mentioned previously, the USRP2 uses an analog RF filter for the incoming signal, samples at 100 MHz, demodulates the signal to baseband, resamples it at a lower frequency, and exports this resampled signal to a computer. All of these components, with the exception of the initial analog filter, are implemented digitally and GNU Radio serves as an interface to control their specifications including the demodulation frequency and the resampling rate. In addition, GNU Radio also makes it possible to record the received data for post processing in programs such as Matlab.

2.2.2.2.1 Processing Introduction

All of the radar post processing is conducted within the receiver software module and can be broken into three processing blocks: range, Doppler, and direction. Range processing determines how far the target is from the receiver, Doppler calculation identifies its radial velocity, and direction finding locates the azimuth of the target.

Target characteristics can be identified by comparing the transmitted signal to the received signal of each antenna and noting the differences. For example, the target's range is characterized by a time delay. A frequency shift between the transmitted and received signals is indicative of motion. Additionally, a phase offset between the signals acquired at each antenna can be used to determine the direction of a target.

2.2.2.2.2 Range Processing

Range is the distance between the receiver and a target. It is calculated by measuring the amount of time that has elapsed between transmitting a signal and receiving the reflected transmission (Skolnik). The electromagnetic signal travels at $3x10^8$ m/s, the speed of light, from the transmitter to the target and back to the receiver, as depicted in Figure 23. The signal travels a total distance of D1 + D2, where D2 is the range. D1 can be approximated by D2 + D3, assuming the target always remains to the right of the receiver and the transmitter is much closer to the receiver than the target.

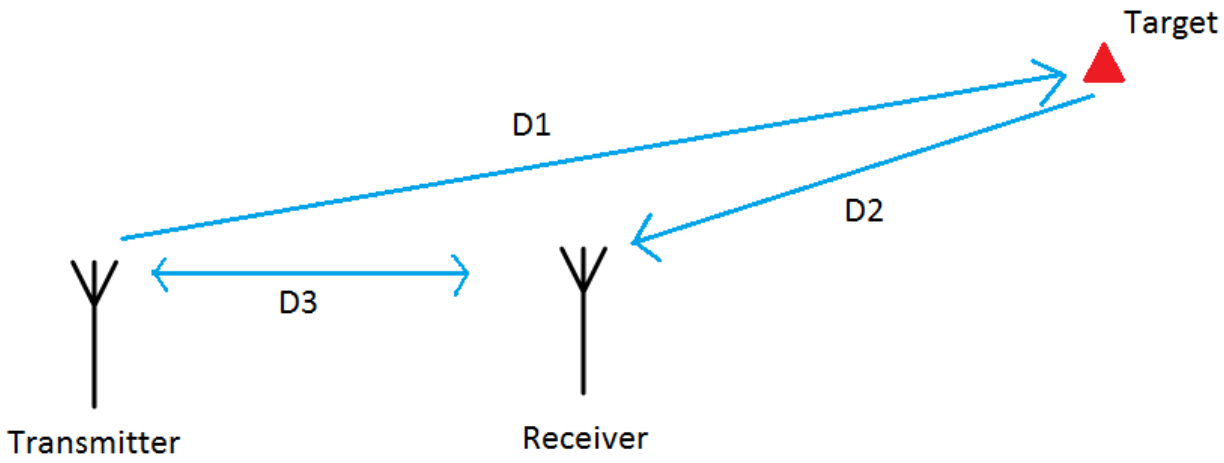


Figure 23: Signal propagation path. The transmitted signal travels a distance D1 to the target and the reflected echo travels a distance D2 to the receiver for a total distance of D1 + D2. D3 is the distance between the transmitter and receiver. D1 can be approximated by D2 + D3.

The arrays used by Group 33 only view targets with a narrow window (approximately 30°) so the targets they observe are always to the right of the receiver. Additionally, the targets are thousands of kilometers away while the distance between the transmitter and receiver is generally no more than 100 kilometers. The total distance covered by the transmission is approximated by *2*D2 + D3*. Therefore, the range D2 can be calculated by

$$D2 = \frac{c\Delta t - D3}{2} \approx \frac{c\Delta t}{2} \text{ (for D1} \gg \text{D3)}$$

where *c* is the speed of light, 3 X 10^8 meters/second, and Δt is the elapsed time (in seconds) between transmitting and receiving the chirp.

2.2.2.2.3 Doppler Processing

The next step in radar processing is known as Doppler processing which is used to determine the speed of a target in the direction to/from the receiver. As a target moves relative to the radar, the phase of the reflected signal at the receiver changes between chirps and Doppler processing works by determining these phase changes. The Doppler frequency F_D is defined as the change in phase over time:

$$F_D = \frac{d\Phi}{dt}$$

This frequency is generated as a result of the Doppler Effect, the same principle which causes an ambulance siren to appear to have a higher pitch as it approaches a bystander and a lower pitch as it drives away. For a target moving towards the receive array, as shown in Figure 24, the reflected chirp has less distance to travel and hence the phase change between chirps increases the received frequency by F_D.

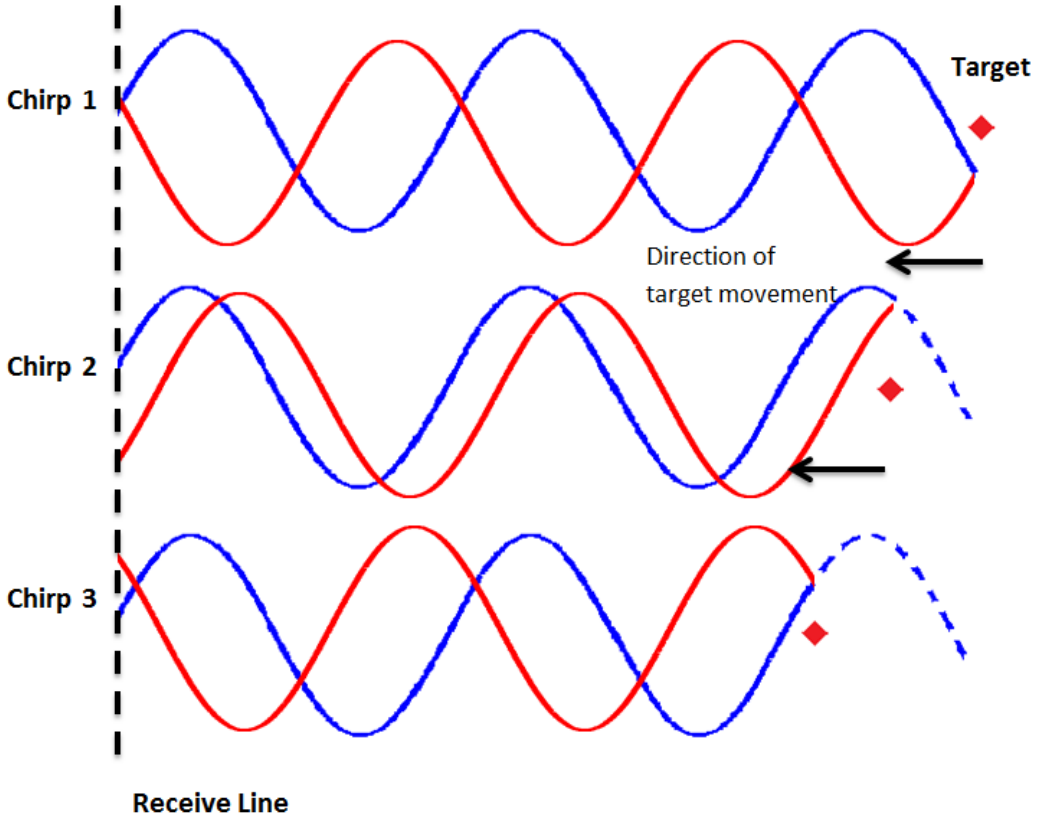


Figure 24: Doppler Effect: The blue sinusoid represents the transmitted chirp and the red sinusoid represents the reflection. As a target moves, the phase of the received signals changes.

2.2.2.2.4 Direction Finding

Direction finding is the process used to determine the location of a target relative to the receiver. Direction can be identified by computing the phase difference between each antenna in the array. If equally spaced, the phase difference between adjacent antennas is the same for the entire array. E.g. the phase difference between antennas 1 and 2 is approximately the difference between antennas *n* and *n+1*. So if we use the first antenna as a reference, we can determine the expected phase offset at each n^{th} element by

$$\Phi(n) = \frac{2\pi}{\lambda} d * (n-1) \sin\theta$$

where ϕ(n) is the phase offset of the n^{th} element, *d* is the distance between antennas, and Θ is the incident angle of the reflected chirp (Jenkins), as shown in Figure 25. This equation assumes that the receiver is in the far field of the chirp source (the target) such that the signal paths from the target to each antenna can be approximated as parallel line segments.

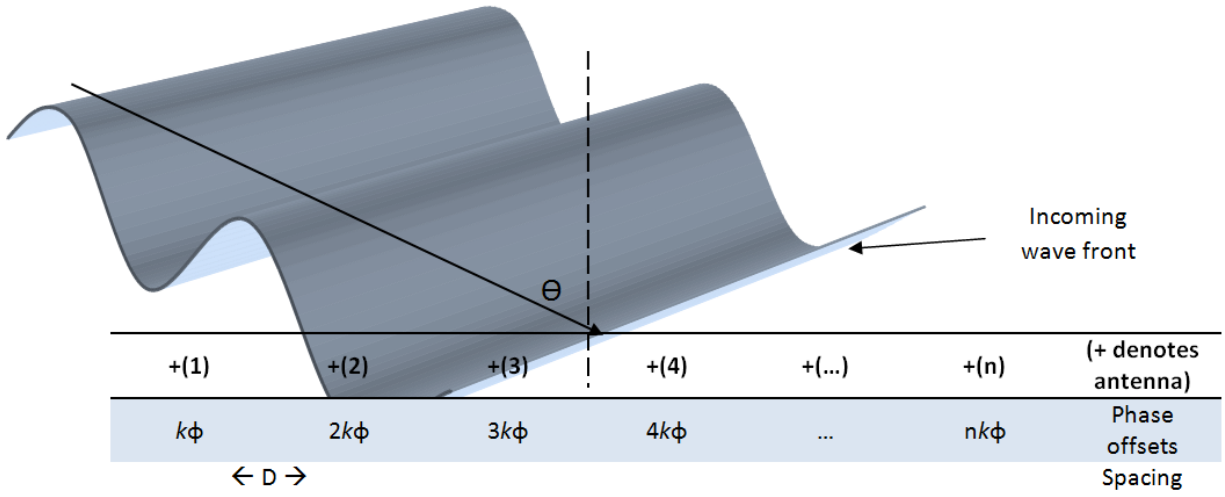


Figure 25: Incoming signal arriving at a receive array. The wave front arrives at different antennas at different times. Thus at any instant in time, the phase of each antenna's acquired signal is related to the spacing between antennas and the incident angle of the transmission.

3 Building a Radar

The purpose of this chapter is to provide our methodology for designing, constructing, and testing our continuous wave radar system. This chapter first discusses the radar system design which covers the hardware and software components of the transmitter and receiver. This section focuses on the receiver design since it is the primary concentration of the project. The next section, time synchronization testing, recaps the timing requirements for the system, explains the methodology for testing synchronization between the receivers in the array, and provides the results of the experiments. The last section, radar processing, describes how the radar principals discussed in the background were implemented into post processing algorithms in Matlab.

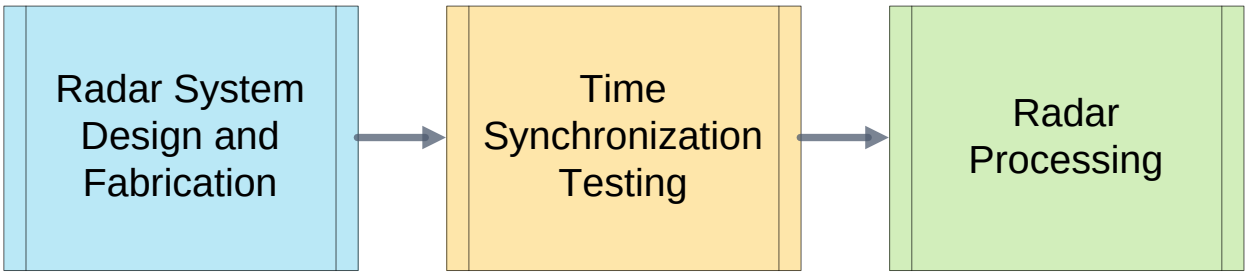


Figure 26: Process for building and testing our radar system.

Figure 26 shows the high level process used for developing this radar system and organizing this chapter. First, the radar is built. Second, the components are synchronized and calibrated. Last, radar processing methods are added to the system.

3.1 Radar System Design

The purpose of this section is to describe how the radar system was designed and constructed. This section discusses the receiver and the transmitter, the two major components of our radar as shown in Figure 27. The receiver segment covers design considerations, an overview of the equipment, and the methods necessary to develop the phased receive array.

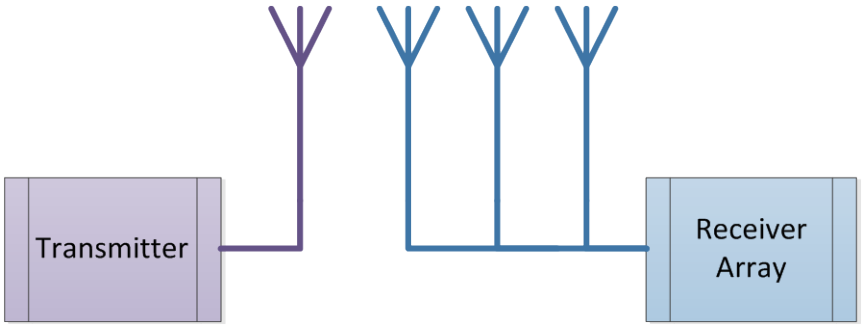


Figure 27: The radar system developed for this project can be divided into two high level components, the transmitter and receiver.

Since the design of a transmitter was out of the scope of this project, a transmitter was provided by Group 33 at MIT Lincoln Lab. The transmitter segment discusses the various components used in Group 33's transmitter setup. It provides the configurations and procedures necessary to incorporate the transmitter into the radar system.

3.1.1 Transmitter

This section discusses the transmitter setup that was provided to us by Group 33. There are two main components in the transmitter setup: a digital waveform generator and an HF power amplifier as shown in Figure 28. A GPS clock was used to synchronize the transmitter with the receiver array. Without timing synchronization the radar processing algorithms do not yield proper results for range and direction. Multiple HF power amplifiers were used to transmit the signal with a greater power (watts). The control computer was responsible for supplying the digital waveform generator the parameters to create a linear frequency modulated (LFM) chirp including the center frequency (MHz), bandwidth (KHz), sample period (μs), and waveform repetition frequency.

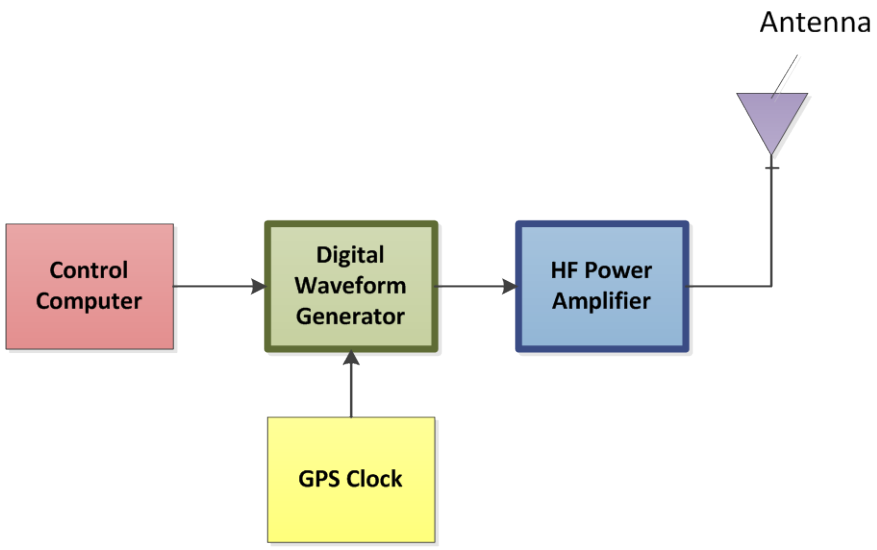


Figure 28: Transmitter Setup. Includes Control computer, waveform generator, HF power amp and GPS synchronization clock

The control computer ran custom written software to control the BAE digital waveform generator. The software called for the input parameters mentioned earlier in the section to create the LFM chirp to be broadcasted. Once the parameters were defined and the transmit sequence initiated, the digital waveform generator sent the chirp it created to the HF Power amplifier which amplified the signal and broadcasted it through a monopole antenna. The HF power amplifier had the ability to transmit up to 50 watts of power. In this setup we used 3 dB of attention on the output signal which reduced the transmit power to only 29 watts (measured).

3.1.2 Receiver

This section discusses the hardware and software components implemented in our receiver array. Figure 29 shows the hardware and software components of the receiver design which are divided into three main steps necessary to build the phased receive array. The first step, hardware connections, explains the hardware functional requirements and signal distribution components such as signal amplifiers and data bus connectors. Second, installation discusses how to configure the software on the computer and the firmware on the receivers. Last, radio programming explains how to network the radios to the host computer and program their internal functionality using GNU Radio.

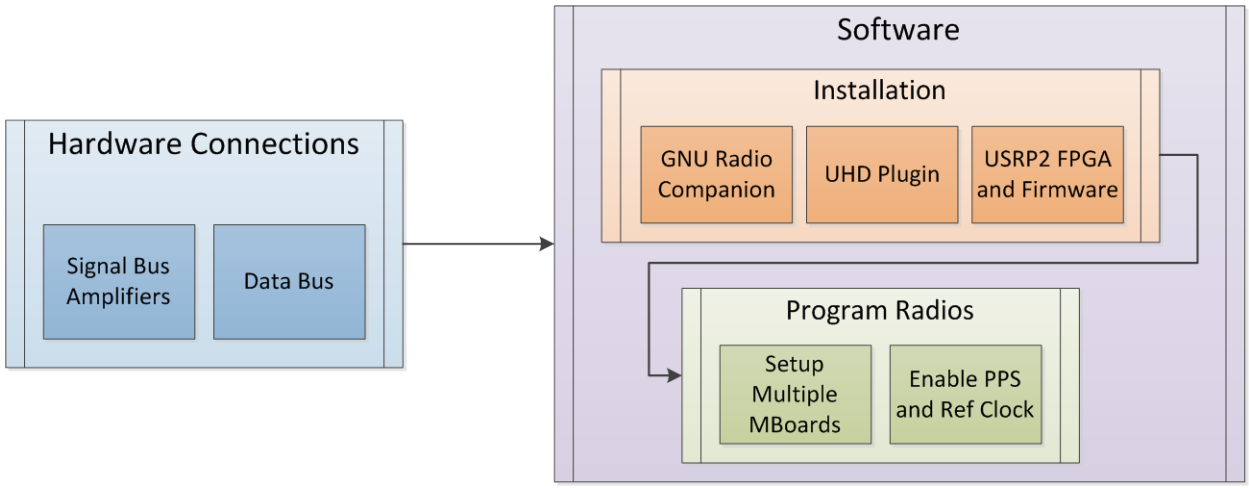


Figure 29: Hardware and software components. The receiver design can be divided into hardware and software components; the software includes installation and radio programming.

3.1.2.1 Hardware

This section presents the design of the signal distribution components used to build the receiver array. The section begins with a high level hardware configuration flow diagram to explain the major components of the system. The following sections outline the requirements for the design based on the hardware specifications.

3.1.2.1.1 Design overview

Figure 30 shows a high level hardware configuration diagram consisting of six blocks. This section only focuses on the four major functional blocks: the GPS clock, the signal amplifiers and splitters, the software defined radios (SDR), and the signal source. Each of these components serves a specific role in our system.

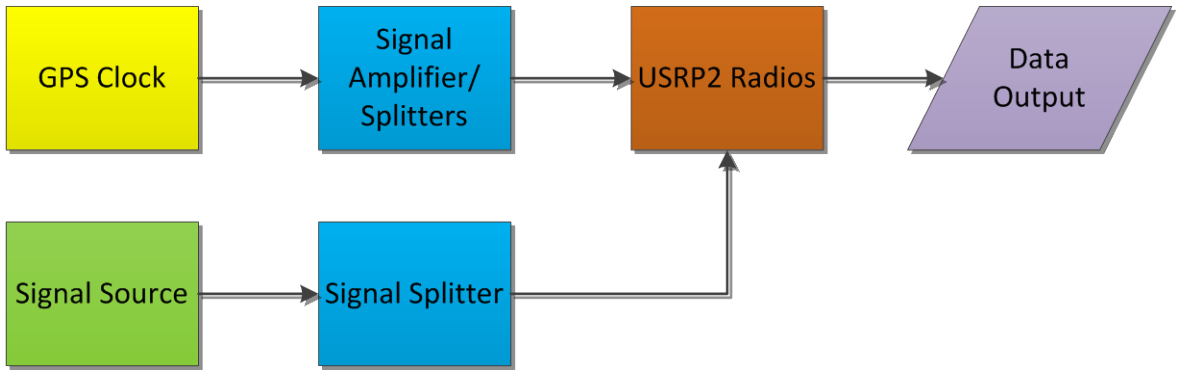
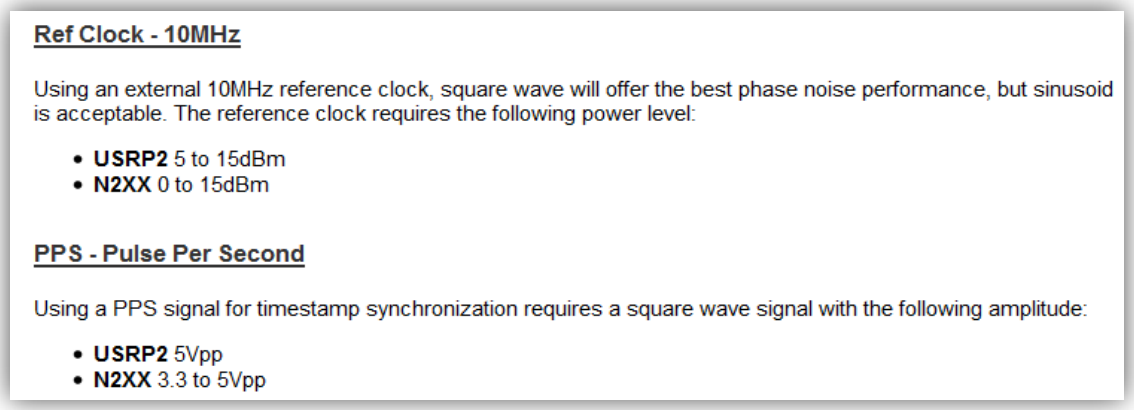


Figure 30: High level hardware configuration: the four major functional blocks in this system are the GPS clock, the signal source, the signal amplifiers and splitters, and the software defined radios.

The GPS clock is essential to our system because it serves as a common time reference for up to six USRP2s in our array. It outputs a 10 MHz reference clock signal and a 1 PPS (pulse per second) signal which are sent to the six SDRs through a powered signal splitter. The purpose of these clock signals is to train each of the USRP2's 100 MHz local oscillators. Synchronizing each of the local oscillators is necessary for the radios to sample data collectively. The signal amplifiers and splitters are used to evenly divide and distribute the 10 MHz clock signal and 1 PPS into 50Ω lines to the SDRs. The amplifiers meet the 5V peak-to-peak voltage requirements for the PPS input and the 10 dBm power requirements for the reference clock input on the USRP2. The signal source refers to the origin of the received data. In our radar receive array, the signal source is an antenna. During the clock synchronization testing, the source is a signal generator.

3.1.2.1.2 Functional Requirements

Synchronizing several USRP2s together requires specific input levels for the reference clock and PPS signals. The requirements for the Ettus USRP2 Software defined radio are shown in Figure 31. All of the data and timing signals used in this project are transmitted via 50Ω SMA cables.



Ref Clock - 10MHz

Using an external 10MHz reference clock, square wave will offer the best phase noise performance, but sinusoid is acceptable. The reference clock requires the following power level:

- **USRP2** 5 to 15dBm
- **N2XX** 0 to 15dBm

PPS - Pulse Per Second

Using a PPS signal for timestamp synchronization requires a square wave signal with the following amplitude:

- **USRP2** 5Vpp
- **N2XX** 3.3 to 5Vpp

Figure 31: Ettus USRP2 Requirements. The input requirements that are shown in this figure are for the reference clock and Pulse Per Second. The Ref. clock requirement is a minimum power level which has to be met in order for the device to function correctly. As for the PPS, a minimum voltage must be meet in order to register as a pulse to the device. (Ettus 2)

The GPS clock, also known as a GPSDO (Global Positioning System Disciplined Oscillator), used for the 10 MHz clock and PPS signals in this project is the Jackson Labs *Fury*. This device was chosen by Lincoln Labs because of its portability and ease of implementation in a future "suitcase radar" application. Figure 32 shows the output specifications for the 10 MHz clock and PPS signals for this model.

Ref	Name	Function	Specification	Pinning
J4	Sine Out	Sine Wave Out	10.0MHz, +7dBm (+-3dBm), 50Ohms, End-Terminate	3-Sine, 2-GND, 1-GND
J6	CMOS Out	TTL/CMOS Out	3.3V/5Vpp 50Ohms, Do Not Terminate	1-Digital Output, 2-GND
J2	1PPS Out	1PPS Output	5V CMOS, 50Ohms, Do Not Terminate	3-1PPS Output, 2-GND, 1-GND

Figure 32: Jackson Labs Fury (GPSDO) Output Specifications. The datasheet excerpt for the Jackson Lab Fury outlines the output power level for the 10 MHz ref out, as well as the pin out of the BNC connector. The PPS signal voltage specs. are also listed.

Comparing the input requirements of the USRP2 SDR to the output specifications of the Jackson Labs Fury GPSDO shows that the two devices are not always compatible. The USRP2 reference clock input level has to be between 5-15 dBm, but the Fury ref clock output is 7 dBm with a +/-3dBm tolerance. If the Fury operates at the minimum output level of 4 dBm it does not meet the USRP2 requirement. In addition the output power level decreases when the reference signal is split between several USRP2s. To address this concern we researched clock signal amplifiers and splitter devices and chose the Pulse Research Lab's PRL-414B 1:4 Fanout 50Ω TTL line driver. This device is designed for clock distribution with the capability to drive four separate output lines at frequencies up to 100 MHz. Using this device ensured that we met the 5 dBm minimum power requirements of the USRP2.

The Jackson Labs Fury provides a 5V CMOS output signal that satisfies the 1 PPS 5 volt peak-to-peak signal requirement of the USRPs as shown in Figure 32. The Pulse Research Lab's PRL-414B 1:4 Fanout 50Ω TTL line driver is qualified for 1 PPS/IRIG-B signal distribution. This device addresses the concern of the PPS output level dropping when connecting up to six USRP2s.

3.1.2.1.3 Construction

The following section outlines the radar hardware setup of six USRP2 SDRs. The necessary hardware that was used in the setup is described in detail as well as the physical connections that were made between the different components.

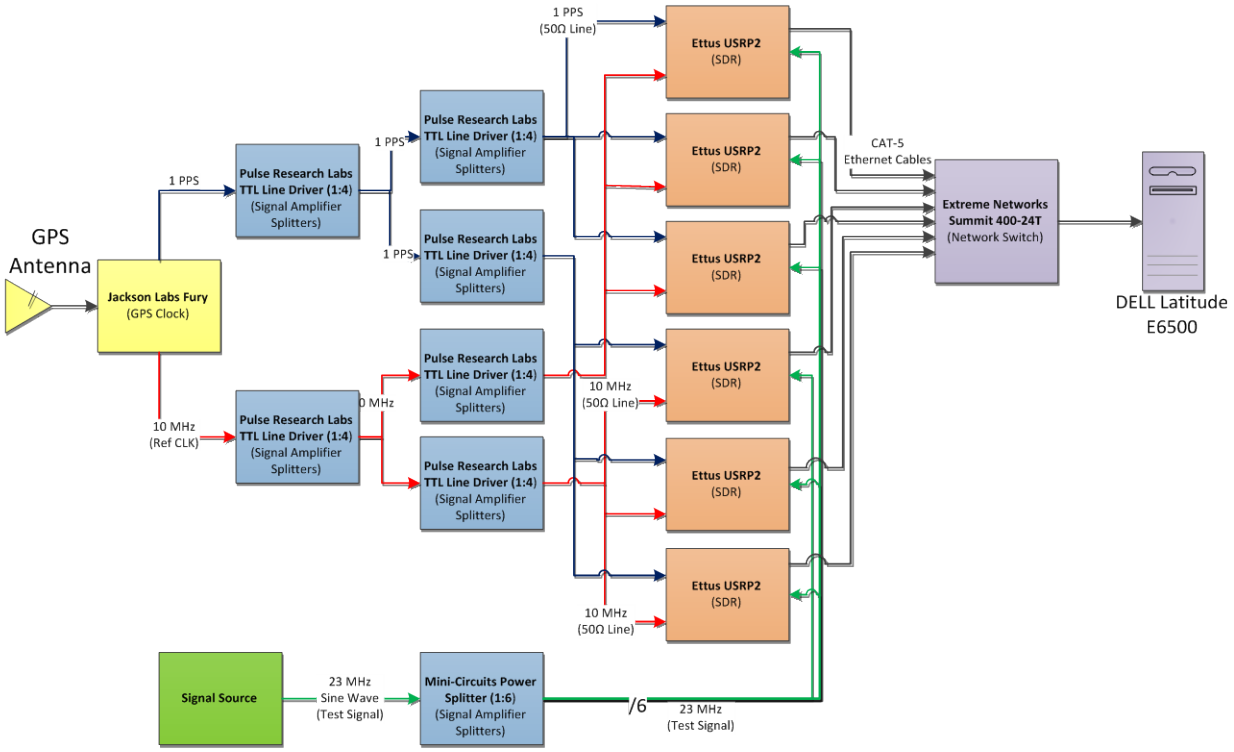


Figure 33: Radar hardware. The flow diagram represents the entire hardware setup of our receive array. The blocks are all color coded, where yellow and green blocks are signal generation sources, blue indicates signal amplification and distribution sources, orange represents the SDRs, and purple represents networking and computer hardware. Note that the signal source block refers to the six antenna array when the system is used for radar processing or the Agilent Technologies E4420B signal generator when the system is used for time synchronization testing.

Figure 33 shows the assembly of our system including the data and timing lines connecting each component. To improve the accuracy of the receiver, the number of variables in the system was kept to a minimum. All of the timing cables (those connecting the 10 MHz and 1 PPS) were measured to the same length (24 inches, the shortest cable length available) and the number of connecting components such as SMA to BNC adapters and elbow joints were kept to a minimum.

The hardware used in this setup is outlined below in Table 4.

Table 4:Receive Array Component List	
Component:	Quantity:
Jackson Labs Fury GPSDO	1
Pulse Research Labs Fanout TTl (1:4) Line Drivers	6
Agilent Signal Generator E4420B	1
Mini-Circuits Power Splitter ZB4PD1-5WT	1
Extreme Networks Summit 400-24T	1
Ettus USRP2 Software Defined Radio	6
SMA [F] - [f] 2 FT Cable	14
BNC [F] - SMA [M] Adaptor	16
SMA Elbow [M] - [F]	12
CAT-5 Ethernet 4 FT Cable	5

As mentioned earlier in the hardware functional requirements section, the Jackson Labs Fury GPSDO was responsible for supplying a 10 MHz reference clock and 1 PPS signals to the USRP2 SDRs to keep them synchronized. To supply the correct signal levels and power to the USRP2 PPS and ref clock inputs, six Pulse Research TTL Line drivers were used to meet the input requirements. The USRP2 outputs were connected to a gigabit enabled network switch. From the network switch the laptop polled the data live from all six USRP2s.

The physical connections were made as follows. Two SMA [F] – [F] 2 foot cables with BNC [F] - SMA [M] Adaptors on the ends of each cable connected the 10 MHz reference clock output of the Fury to the input of one of the TTL line drivers and the other from the PPS output to the input of the other TTL Line driver. From the outputs of the two TTL Line drivers 12 SMA [F] – [F] 2 foot cables connected to the corresponding reference clock and PPS inputs on the six USRP2's. Six CAT-5 4 foot Ethernet cables connected the data output of the USRP2's to the Extreme Networks gigabit enabled network switch. From the Agilent signal generator a SMA [F] – [F] 2 foot cable connected to the mini-circuits 1:4 power splitter. Two of these outputs drove two more 1:4 power splitters yielding a total of eight outputs, of which only six are used. Six SMA [F] – [F] 2 foot cables ran from the output of the power splitters to the

RF1 inputs on the six USRP2 SDRs. A Dell Latitude E6500 Laptop running GNU Radio was connected to the Extreme network switch with a 6 FT Ethernet cable to acquire the data from the USRP2s.

3.1.2.2 Software

The purpose of this section is to present the software components and methods used in our receive array. This section discusses the installation of GNU Radio and the USRP2 firmware followed by an introduction to networking and programming the USRP2 using GNU Radio.

3.1.2.2.1 Installation

As mentioned in the background section, the open source program GNU Radio was used for programming and interfacing with the USRP2. However, the program wasn't installed directly as provided by the GNU Radio repository. In order to add increased functionality to the USRP2 such as enabling external reference clock sources and the debug pins on the motherboard, some of the code was altered before installation.

Instead of extracting a .tar.gz file or installing through the Ubuntu Software Center, we built GNU Radio from the source code because it allowed us to change files before installation. Directions for building the software are beyond the scope of this report but detailed directions can be found on the GNU Radio Wiki site (GNURadio). Once the dependent files were installed and the source code was downloaded, the Universal Hardware Driver (UHD) plugin was prepared for installation. UHD is a firmware version for the USRP2 that is compatible with GNU Radio and provides control for the radio's oscillators and reference clock sources. Synchronization across an array is not possible without the UHD plugin because there would be no control for the reference clock. The UHD source code is provided by Ettus, the company that makes the USRP2. The UHD version used in this project was release 003.001.001, the latest version available when the project began. After downloading the archive, the following changes were made:

First, the UHD archive was added to the GNU Radio source folder created during the initial build (GNURadio). This creates the directory `/gnuradio/uhd`, adds UHD functionality to GNU Radio, and gives the user control over the reference clocks.

Second, the file `/gnuradio/uhd/host/lib/usrp/usrp2/clock_ctrl.cpp` was edited using the text editor Gedit. This file controls the debug clock pins located on the USRP2 motherboard. The pins are turned off by default but changing a few lines of code can turn them on which makes it possible to

characterize the clock drift over long periods of time using an oscilloscope. Table 9 in Appendix 5.2 shows which lines of code were edited.

The next component installed was the USRP2 firmware. This file did not need to be modified in any manner so the prebuilt file was downloaded directly from the Ettus file repository (Ettus). The firmware image was burned to an SD card using the USRP2 card burner application also provided by Ettus. To ensure compatibility with the UHD plugin, the same version (003.001.001) was used for the radios. Once the software on both the host computer and the USRP2s had been installed, the radios were ready for programming.

3.1.2.2.2 SDR Programming

Before using GNU Radio, the network configurations had to be programmed onto the USRP2s. Since all of the radios communicate with the host computer via a gigabit Ethernet connection, each one must have a uniquely assigned IP address. Without a distinct IP address, it is not possible to identify the sender of information packets and hence the source receiver.

The default IP address for the USRP2 and GNU Radio is 192.168.10.2; since the array consisted of six receivers, the remaining five were identified at addresses 192.168.10.3 through 192.168.10.7. The IP address was programmed using the utility provided by Ettus, '`usrp_burn_mb_eeprom`' and the following line of code:

```
sudo ./usrp_burn_mb_eeprom --key=ip-addr --val=192.168.10.x
```

The 'x' in 192.168.10.x indicates the last value of the IP address. The use of 'sudo' was necessary because the script required root privileges to transfer data through the Ethernet port. Once programmed, a pathway that allowed GNU Radio to communicate with the USRP2 existed.

GNU Radio consists of a graphical interface which enables the user to program radios in a flow-chart manner similar to the Matlab product Simulink. This interface makes it easy to quickly prototype a program without much programming knowledge. However, the python scripts it generates can be edited manually providing even more freedom and versatility to the user.

The most important functional block required to program the radios for this project was the UHD: USRP Source. As shown in Figure 34, this component controlled the number of receivers, their IP address, and the settings for synchronization to an external reference clock. The settings used in this project are outlined in Table 5. The receive center frequency was set to 23 MHz, the carrier frequency of the

transmitted chirp. This will be further explained in the radar processing methods section. The sampling rate used after the signal was demodulated to baseband was set to 200 kHz because it was the minimum interpolated sampling frequency supported by GNU Radio. In addition, 200 kHz also satisfied the Nyquist frequency requirement because it was more than twice the highest frequency of the radar chirp which had a 20 kHz bandwidth centered at baseband. The value 'samp_rate' was used, as shown in Figure 34 because it was a global variable initialized by GNU Radio.

Table 5: UHD: USRP Source Configuration

Component	Value
Output Type	Complex (outputs I and Q values)
Ref Clock	External (uses external reference clock)
Sync	Unknown PPS (uses external PPS signal)
Num Mboards (Motherboards)	6
Mbx: Subdevice Spec	:AB (indicates single receive channel (Ettus))
Sampling Rate (samples/sec)	200e3
Center Frequency (Hz)	23e6

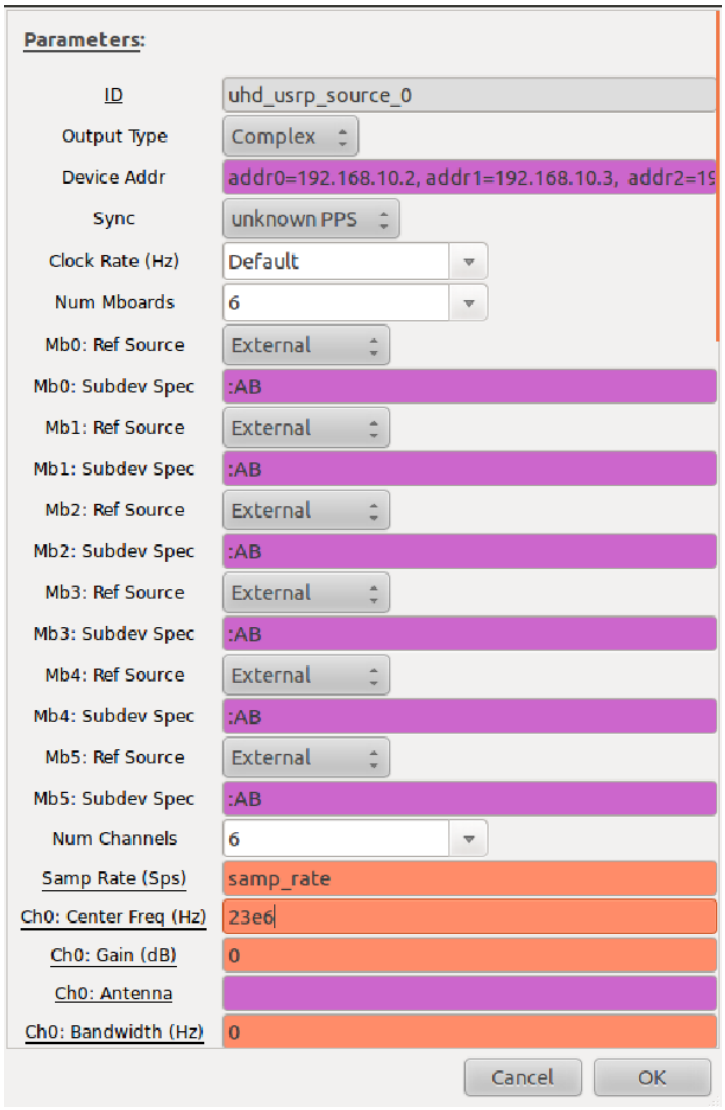


Figure 34: UHD: USRP Source block. This block provides control over how many USRP2s are included in the system (number of motherboards), their IP addresses, the location of the receive daughter cards on the motherboard (Subdevice Spec), the number of output channels, the center frequency, intermediate frequency sampling rate, and synchronization to an external clock reference and PPS signal.

Once the USRP block was configured, each of the six channels was connected to a file sink. This block writes the recorded data to a file which can be accessed later by Matlab. The GNU Radio flow diagram is shown in Figure 35.

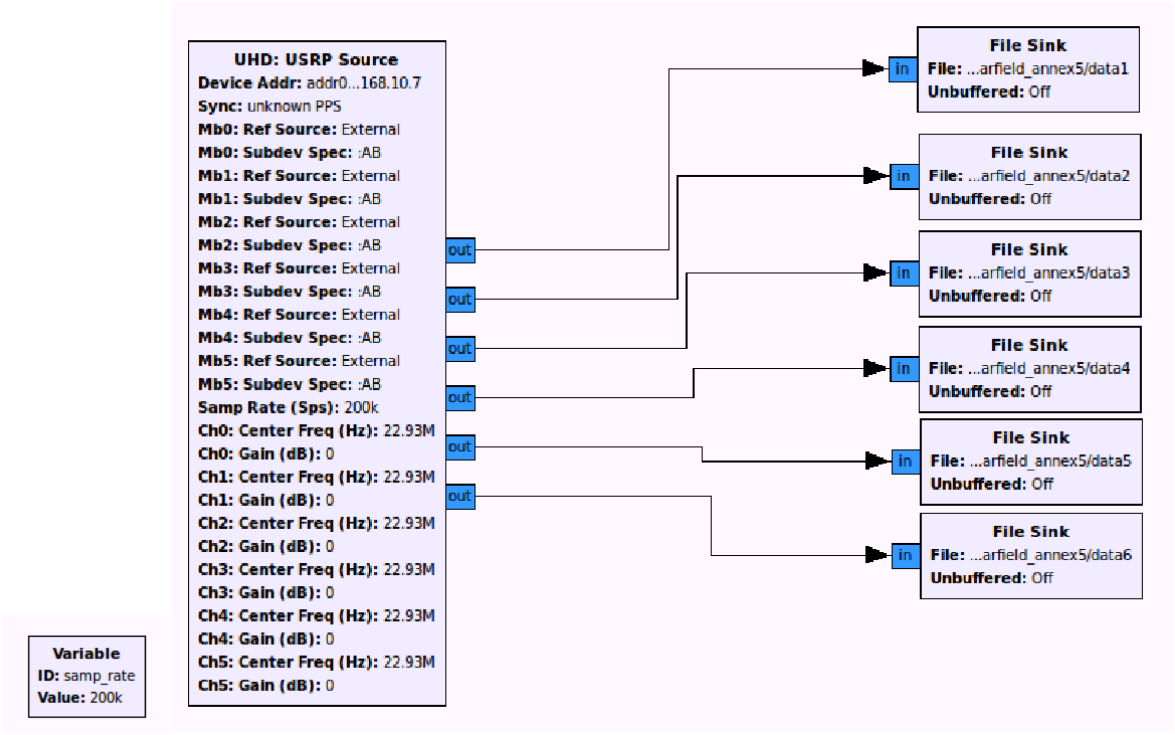


Figure 35: GNU Radio flow diagram. The UHD: USRP Source sets the center frequency, sampling rate, and reads the received data from each USRP2. The raw data are then saved to a file for post processing.

3.1.3 Discussion

This radar was designed with many constraints specified by Lincoln Labs. For example, the array was required to use the USRP2 software defined radio and the Jackson Labs Fury GPSDO. Additionally the size of the array was limited to the six receivers we were able to acquire. With these constraints, we identified the appropriate signal amplifiers and splitters that met the input and output requirements for each component. Lower cost signal splitters could have been designed and built, but due to the time constraints of the project off-the-shelf components were chosen instead. The time synchronization testing section confirms that all timing and data signals were configured correctly to synchronize the system.

3.2 Time Synchronization Testing

The purpose of this section is to discuss the process used to characterize the timing characteristics of the USRP2. Two key goals of these experiments were to identify the level of synchronization between the USRP2s in the receive array and to determine whether the local oscillators would drift away from the reference clock over long periods of time. This section recaps the timing requirements necessary for our radar system, explains the two different methods used to test synchronization across the array, and discusses the results.

3.2.1 Introduction

Recall that each USRP2 in the array had a 100 MHz local oscillator that controlled the timing of the analog to digital converter (ADC). This oscillator was trained by a 10 MHz external reference clock and a 1 Hz PPS signal provided by the Jackson Labs *Fury*. A tolerable offset between each receiver was 5 ns, half the period of the 100 MHz ADC clock. Time offsets less than 5 ns ensured the receivers did not drift by an entire sample period (10 ns) and thus prevented missed samples. Figure 36 depicts what can happen if a receiver samples outside of the 5 ns tolerance window. The furthest blue line to the left depicts a sample that occurred outside of the 5 ns tolerance window. As time progresses, the sampling location (blue line) drifts further from where it should be sampling. The second blue line should have sampled within the 5 ns tolerance window centered at 20 ns. The third blue sample (furthest to the right) depicts a slipped sample; it should have sampled at 30 ns but has actually occurred an entire sample (10 ns) later.

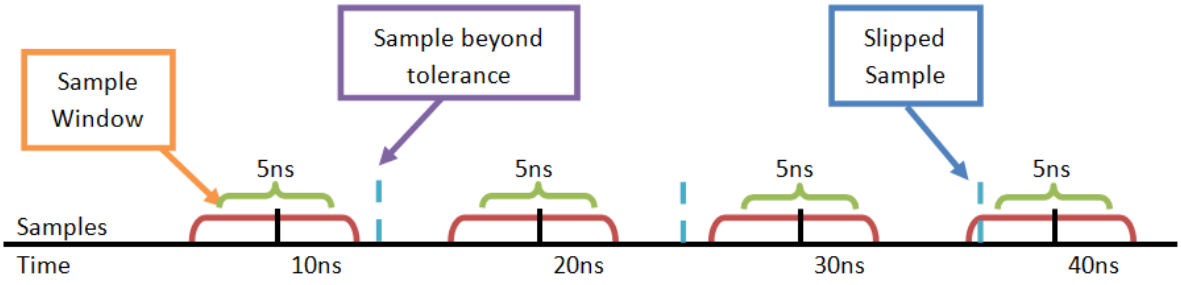


Figure 36: 5 ns drift prevents slipping to the next sample. The furthest blue line to the left indicates a sample that occurred outside of the first 5 ns tolerance window. The second sample is slipped even further, indicated by the blue line between 20 and 30 ns. As time progresses, the offset can drift into the time frame of another sample. The third blue line depicts a slipped sample which should have occurred at the 30 ns sample window but instead has slipped into the next sample window at 40 ns.

In addition to preventing drift, low tolerances were also necessary to achieve low side lobe levels during direction finding. As will be discussed later in the paper, direction finding is achieved by optimizing the magnitude of the vector resulting from the sum of each antenna's component vector for different values of theta theta, the incident angle of the reflected chirp. This sum generates a plot similar to the one shown in Figure 37 with a main peak and side lobes. The peak denotes the direction of the signal so it is imperative to have good side lobe suppression in order for the peak to be distinguished.

In the ideal case where each radio is uniformly weighted, there are no time offsets between receivers, and there is only a single target with no noise or multipath interference, the antennas receive a single sinusoid. The ideally flat wave front of the incoming transmission is represented as a rectangular

window. Thus the direction of the transmission is modeled as a windowed sinusoid; in the frequency domain, this is represented as the sinc function, $\frac{\sin(x)}{x}$. (The math explaining this procedure is explained in the direction finding section.) The side lobes of the direction plot are 13 dB lower than the main beam as shown by the frequency response of the sinc function. However the small time offsets associated with phase-locked timing circuits can introduce variance in the exact sampling time. This variance causes the side lobe level to increase which makes it difficult to discern the peak of a weak signal from the side lobe response of a strong signal – hence complicating the target direction finding problem.

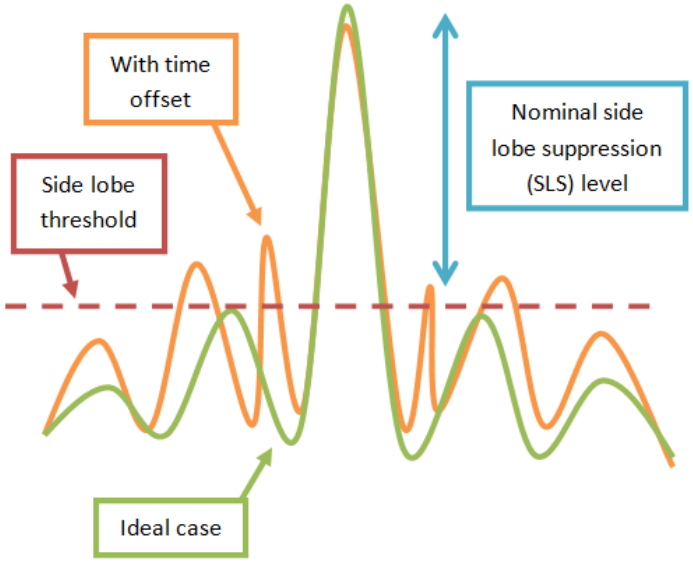


Figure 37: Beam-formed signal side lobes in an ideal case and when there is a time offset.

For this radar system, chirps were transmitted at a 23 MHz carrier frequency which has a period of 43 ns. A 5 ns synchronization tolerance between radios limited the timing error to less than or equal to 15% of the carrier signal period over the 3-30 MHz HF band.

Note that both of the following time synchronization methods were conducted with only four of the six USRP2s. The primary reason for this decision was because of hardware limitations. The oscilloscope we used to compare the ADC clock had only four inputs and each was connected to a different USRP2. To remain consistent with our methods, we only used four USRP2s when determining the time offset using the signal generator.

3.2.2 Signal Generator Testing

The purpose of this test was to determine the level of synchronization by transmitting an identical sinusoid to four USRP2s and comparing their received waveforms. Each of the radios received a sinusoid

from a signal generator but a phase difference between each radio was present because they did not sample at exactly the same time. By knowing each receiver's sampling rate and center frequency, this phase offset was related to a time offset between the two oscillators.

3.2.2.1 Data Collection

For the first synchronization test, we used the same hardware setup established in Section 3.1.2.1.3: Construction, with the exception that only four radios were used instead of six. The Agilent Technologies E4420B signal generator transmitted a 23 MHz sinusoid to the receive input on four USRP2s, as shown in Figure 33. We chose to use 23 MHz because this frequency was the same frequency allocated to Group 33 for radar transmissions. As will be discussed later, the chirps used for our radar processing were transmitted at a carrier frequency of 23 MHz. We used a powered splitter to distribute the signal generator's sine wave to the four USRP2s. The transmitter ran for 60 seconds and each receiver demodulated the sinusoid to baseband where it was sampled at 200 kHz. This procedure is explained in greater detail in the following section. Each radio's received sinusoid was written to a data file on the computer for Matlab post-processing.

3.2.2.2 Data Analysis

The overall processing flow diagram is outlined in Figure 38. First, the received signals from the four receivers were imported into Matlab. Stream processing was used to analyze the data in epochs because the file size quickly increased to about 100 MB after recording for about 60 seconds; the epochs were contiguous and each 0.5 seconds in duration. Next, the phase difference was computed for each epoch between the following pairs: Radio 1 and Radio 2, Radio 1 and Radio 3, and Radio 1 and Radio 4. Radio 1 was used as the reference for all time offsets because we needed a common reference for comparison. The computed time offset was calculated with respect to Radio 1 because there was no other reference clock source that could interface with the computer. Computing the phase difference between pairs of data is one of the major limitations of this method since the computed time offsets are all relative, not absolute. The resulting histogram shows the three offset values, one for each of the pairs of data.

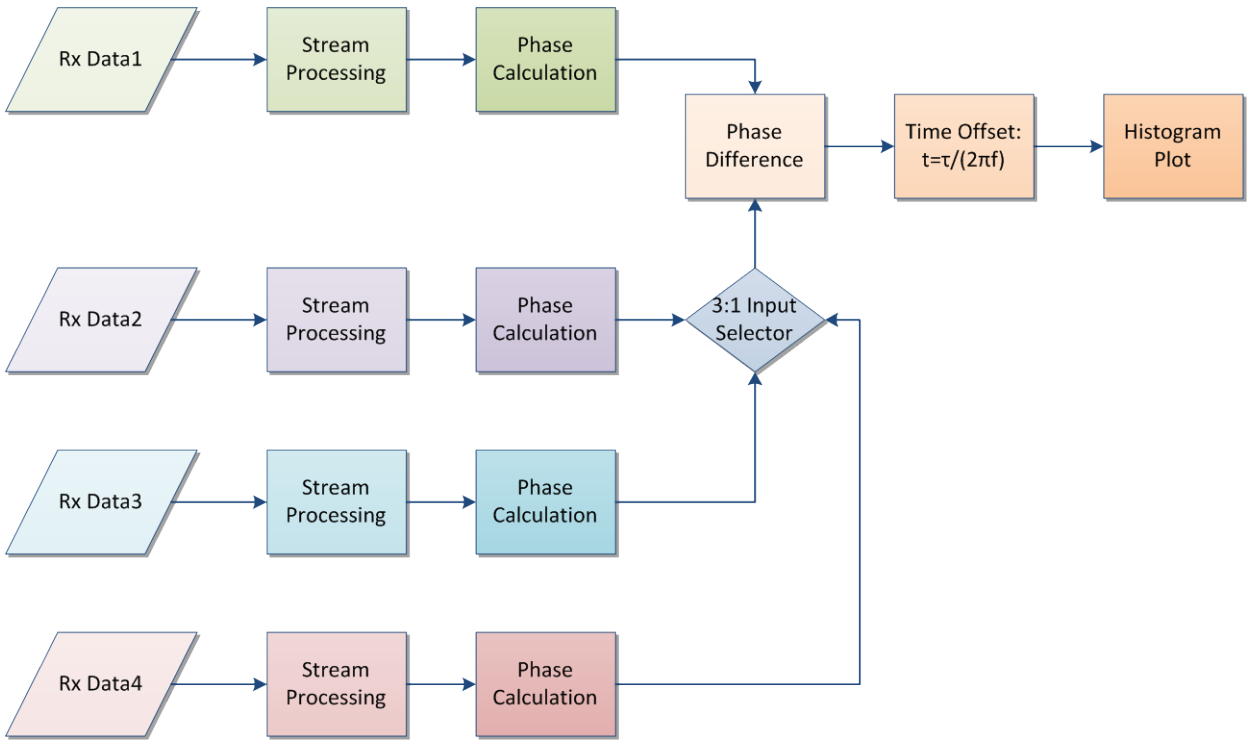


Figure 38: Matlab processing flow diagram. Data is recorded from each of the receivers and imported into Matlab through stream processing. The phase difference between the signals is computed in order to determine the time offset. This is repeated for each signal pair where Data1 is a reference for Data2, Data3, and Data4. The histogram shows the time offset for these three pairs.

Next the phase of each epoch was computed and used to find the phase difference. The following discussion explains how the phase of the demodulated signal is the same as the phase of the transmitted signal by analyzing the demodulation of the in phase (real) component of the transmitted signal. A similar argument applies to the quadrature component.

Let $\cos(2\pi f_c t + \theta)$ represent the sine wave received by the signal generator where t is time, f_c = 23 MHz is the carrier frequency, and Θ is the signal generator's phase offset. The first radio demodulates this signal by multiplying it by $\cos(2\pi f_c t + \phi_1)$ where ϕ_1 is the phase of the first receiver. This assumes that the receiver can attain the same frequency as the transmitter but the phase is not aligned with the signal generator. Demodulation yields a double frequency term, $\frac{1}{2}\cos(2\pi f_c t + \theta + \phi_1)$ and a baseband term, $\frac{1}{2}\cos(\theta - \phi_1)$. The double frequency term is removed when passed through a low-pass filter so only the baseband signal remains. Note that the phase of the first receiver's baseband signal is $\theta - \phi_1$.

Now let the second radio demodulate the same signal, $\cos(2\pi f_c t + \theta)$, by multiplying it by $\cos(2\pi f_c t + \phi_2)$ where ϕ_2 is the phase of the second receiver. Demodulation yields a similar double frequency term

and a baseband term, $\frac{1}{2}\cos(\theta - \phi_2)$ which has a phase of $\theta - \phi_2$. The phase difference between the two demodulated signals is $(\theta - \phi_2) - (\theta - \phi_1) = \phi_1 - \phi_2 = \varphi$, the same phase difference between the two received signals before demodulation. This proof shows that the phase difference of two demodulated sinusoids centered at baseband is the same as the phase difference of the same signals before demodulation (assuming a perfect mixing frequency but inaccurate mixing phase). Since φ is measured in radians, the time offset between the signals can be computed by

$$t_{\text{offset}} = \frac{\varphi}{2\pi f_c}$$

where f_c is the carrier frequency of the transmitted signal in Hz.

3.2.2.3 Results

The histogram in Figure 39 shows the time offset of the three receivers with respect to the first. The negative time offset indicates that the reference (radio 1) is lagging with respect to the three radios shown. The greatest time offset between the radios does not exceed 1.3 ns, well within the 5 ns margin established as one of our deliverables. The range about the mean time offset for each data set is less than 0.16 ns. Table 6 shows the time offset mean, standard deviation, and spread for each of the data sets shown in the histogram.

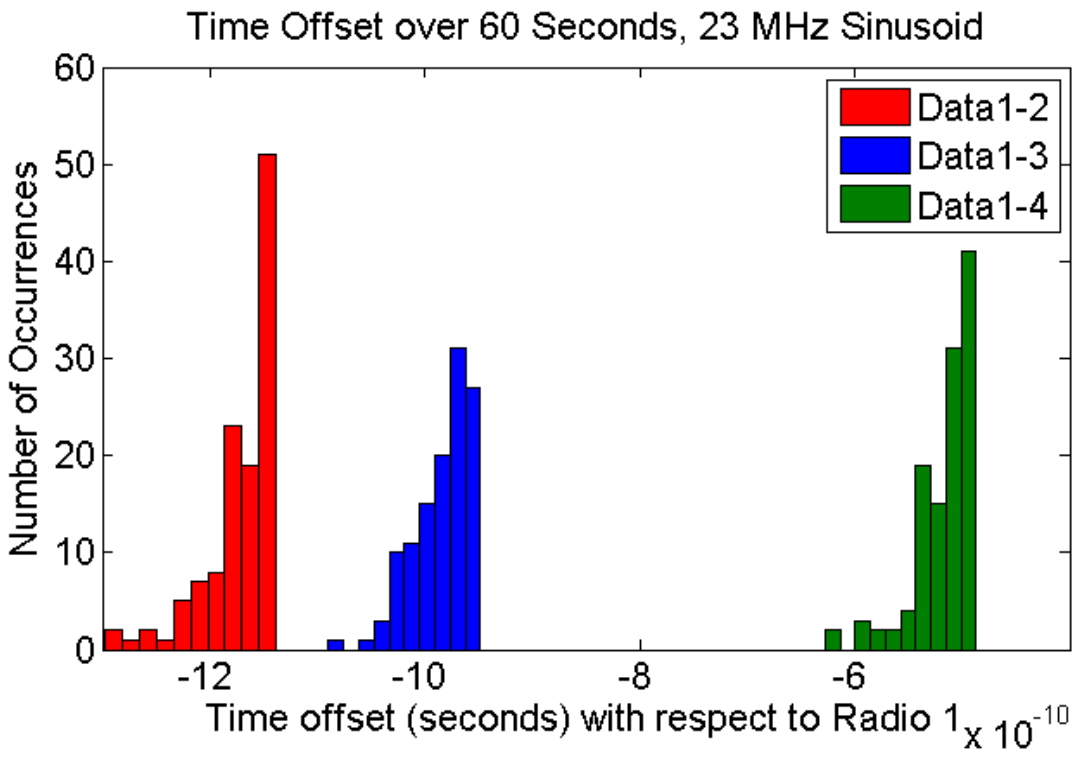


Figure 39: Time offset histogram for 4 radios over 60 seconds. Data1-2 shows the time offset between data sets 1 and 2, Data1-3 shows the time offset between data sets 1 and 3, and Data1-4 shows the time offset between data sets 1 and 4. The maximum time offset does not exceed 1.3 ns.

Table 6: Time Offset Characteristics

	Data 1-2	Data 1-3	Data 1-4
Mean	-1.172 ns	-0.9841 ns	-0.5172 ns
Standard Deviation	0.03159 ns	0.0263 ns	0.0272 ns
Minimum	-1.139 ns	-0.9483 ns	-0.4871 ns
Maximum	-1.297 ns	-1.090 ns	-0.6278 ns
Spread	0.1590 ns	0.1412 ns	0.1407 ns

Although the mean time offset is minimized in this experiment because of the short (24 inch) connecting cables, large time offsets can be tolerated as long as the offset between radios is consistent across the entire array. For example, when these receivers are connected to antenna inputs (rather than a signal generator), cabling might be 10s of meters in length. As long as the lengths of the cables connecting each receiver to the signal source are equal, the time offset of each radio relative to each other remains consistent regardless of cable length. It is the relative cable length difference that affects the time offset.

3.2.3 Clock Debug Pin Testing

The second method used to test the clock synchronization was conducted using the USRP2 clock debug pins and an oscilloscope. We performed two tests using the oscilloscope to monitor the USRP2's ADC signal. The ADC clock signal can be sent to the clock debug pins located on the USRP2 motherboard if GNU Radio is configured as described in Section 3.1.2.2: Software. The purpose of the first experiment was to determine whether the radios could consistently trigger off the 1 PPS signal, keep their oscillators locked to the reference source, and to measure the time offset between the reference clock and each receiver over short time durations (≤ 60 s). The purpose of the second experiment was to determine whether the clocks would drift away from the reference after several hours. Both of these tests helped characterize the level of synchronization by using an oscilloscope to plot the ADC clock waveform from the radios. This test provided an alternative approach to using a function generator because we could look directly at the clock waveform generated by the hardware, not characteristics of received signals.

3.2.3.1 Methods

The oscilloscope, an Agilent Infiniium, has four input channels and an auxiliary trigger reference input. For the first test, we connected three of the probes to debug pins on three USRP2s. The clock debug pin is located on port 2 of J503, as shown on the USRP2 schematic in Figure 40. This connection provided the ADC clock signal output on each USRP2. The fourth probe was connected to the 1 PPS output. This signal served as a reference and the oscilloscope triggered on its rising edge. Although the scope could have triggered off the 1 PPS through the auxiliary trigger input, the signal would not display on the screen. For this test, we chose to connect one of the probes to the PPS in order to accurately measure the time offset of each radio from a common reference.

This test ran for 30 minutes and the scope persistence was enabled. The persistence setting is similar to the "hold" function in Matlab; when enabled, the screen does not refresh and each received waveform is plotted on top of the previous one. With this configuration, drift in the clock signal would be noticeable if the received waveforms did not line up with each other. The test setup is shown in Figure 41.

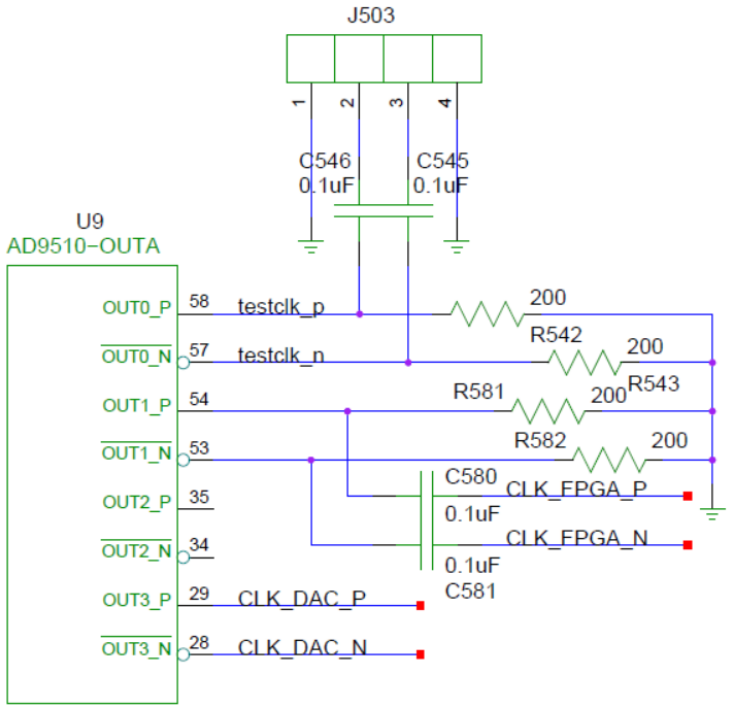


Figure 40: Clock debug pin schematic. Pin 2 of J503 is the clock signal and Pin 1 is ground (Ettus).

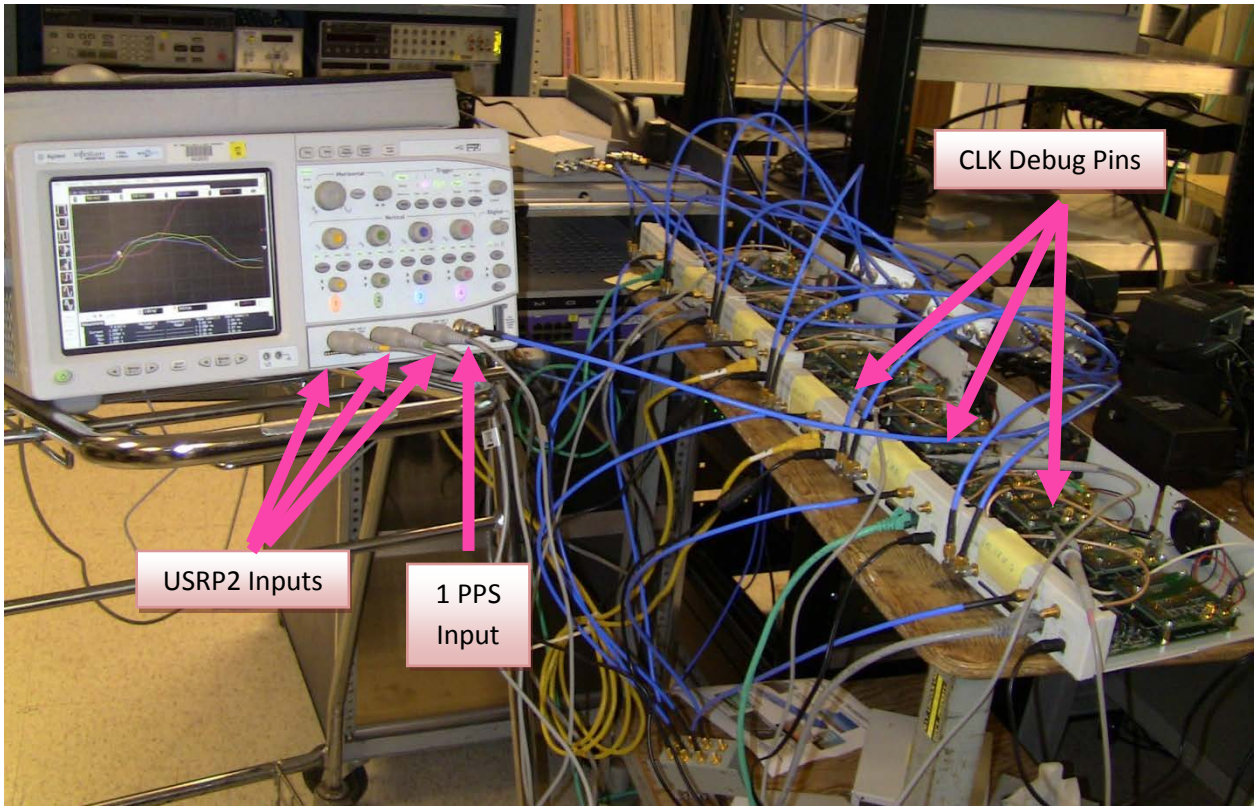


Figure 41: Photograph taken of the oscilloscope test. Channels 1, 2, and 3 are connected the USRP2 debug pins. Channel 4 is connected to the 1 PPS reference signal.

The second test was very similar to the first. One major difference was that all four channels were connected to the USRP2. The 1 PPS signal was connected to the oscilloscope's auxiliary trigger input because the purpose of this test was not to determine the time offset of the radios but whether any of their local oscillators drifted after running for several hours. This test spanned 15 hours, much longer than the previous one which was only 30 minutes. Once the probes were connected and the trigger set to 1.76 V (midway between the low and high voltage of the PPS signal), the display mode was set to "persistent" in order to visualize any clock drift.

3.2.3.2 Results

The clock waveform analysis produced similar results to the signal generator tests. Figure 42 shows a 30 minute persistence plot of the 1 PPS and the clock signals from three radios. Channels 1 – 3 show the 100 MHz ADC clock signal from three different receivers (yellow, green, and purple). Channel 4 shows the rising edge of the 1 Hz PPS signal (pink). Each of the receivers triggered their local oscillators on the rising edge of the 1 PPS signal. Since one trace was generated every second, there are approximately 1800 overlapping traces. This figure shows that each of the radios triggered correctly because the clock offset for any receiver relative to the 1 PPS did not exceed 2 ns, which was within the 5 ns tolerance.

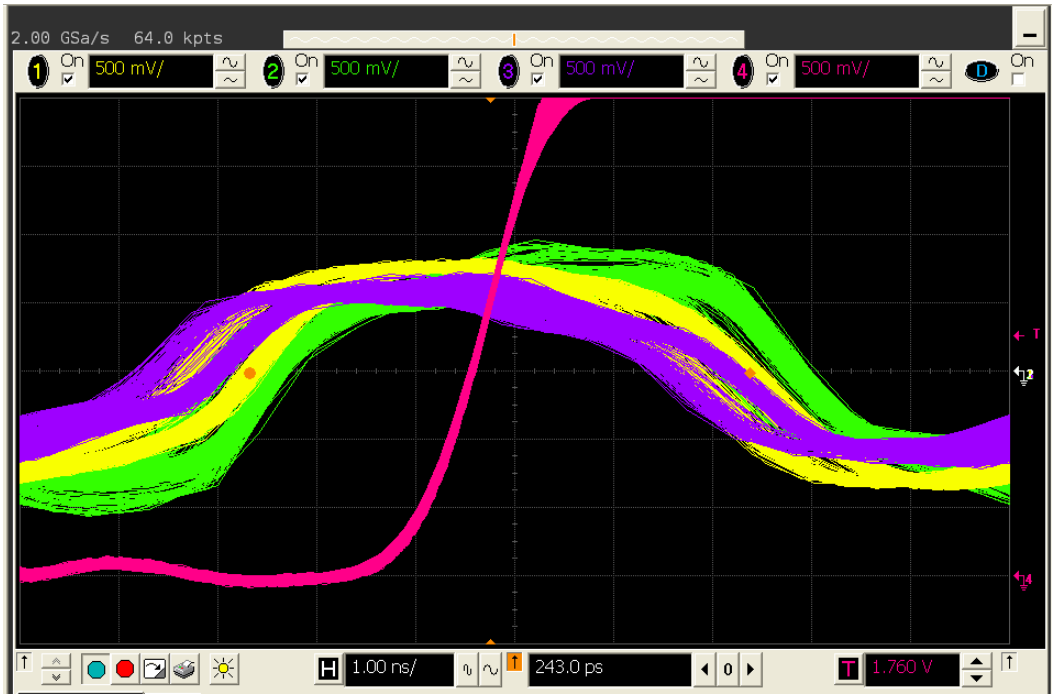


Figure 42: 30 minute persistence plot of the ADC clock triggering on the 1 PPS signal. The 1 PPS trigger is connected to channel 4 (pink); the other three channels (channels 1-3) are connected to three USRP2s. After running for 30 minutes, the maximum time offset is less than 2 ns. The x-axis shows time with 1 ns/division and the y-axis shows the signal voltage with 500 mV/division.

Figure 43 shows a 15 hour persistence plot of the ADC clock signal from four receivers. A new trace was drawn every second so the screen shows approximately 54,000 overlapping traces. This figure shows that even after 15 hours, the clocks remained locked to the reference and the drift between samples did not exceed 2 ns.

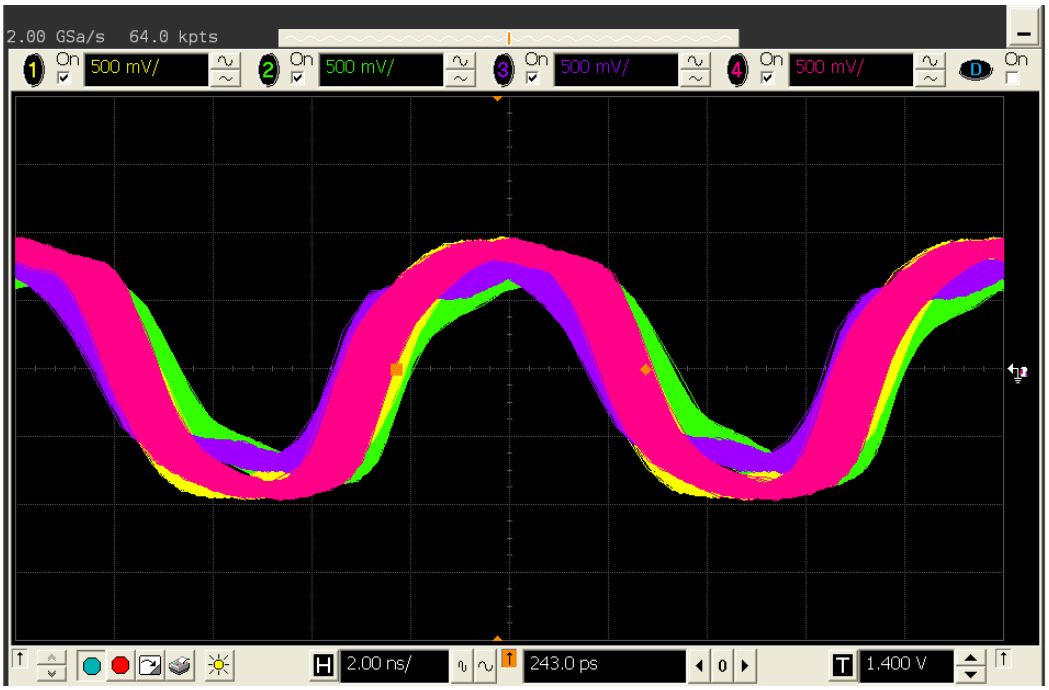


Figure 43: 15 hour persistence plot of four ADC clock signals. For this test, all 4 inputs were connected to different USRP2s and the PPS reference was connected to the auxiliary trigger input. After running for 15 hours, there is no drift greater than 2 ns between the oscillators. The x-axis shows time with 2 ns/division and the y-axis shows the signal voltage with 500 mV/division.

Figure 44 provides a comparison as to what would be expected if only three of the four clocks were synchronized but one was drifting. In this example, the USRP2s connected to channels 1, 2, and 3 all had a reference clock signal; the USRP2 connected to channel 4 however did not. The radios connected to channels 1-3 remained synchronized because their reference signal was not affected. Because the persistence was enabled, channel 4 appeared all over the screen even after running for just 30 seconds (approximately 30 traces). This plot shows that the external reference clock and PPS signals are critical for synchronizing and training the local oscillators.

Figure 44: Example with only three of four radios synchronized. In this example, Radios connected to channels 1, 2, and 3 had a reference clock input; the radio connected to channel 4 did not. The x-axis shows time with 2 ns/division and the y-axis shows the signal voltage with 500 mV/division.

Figure 45 shows what the plot looked like after 30 seconds (30 traces) when none of the oscillators were synchronized to a common reference. In this example, none of the four radios had an external reference clock signal. Because there is no reference, the ADC clock of each radio must rely on its own local oscillator. Since no two oscillators have exactly the same frequency, the radios' clock signals are incoherent. This result is portrayed in the following persistence plot.

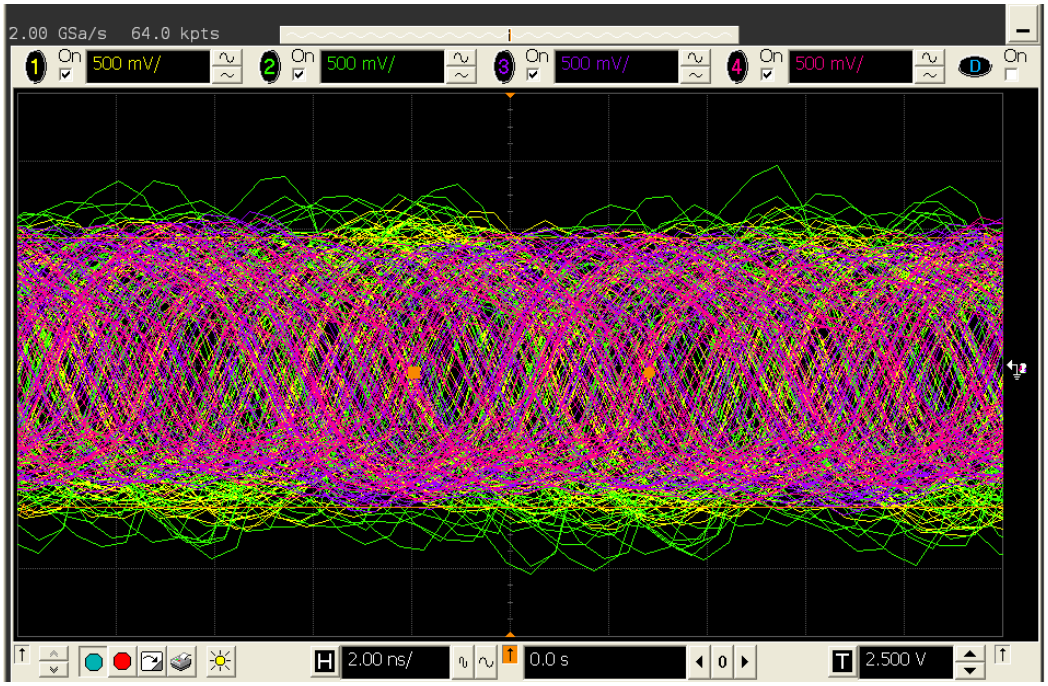


Figure 45: Plot depicting a scenario when none of the radios are synchronized. In this example, none of the four radios had a reference clock signal connected. The x-axis shows time with 2 ns/division and the y-axis shows the signal voltage with 500 mV/division.

3.2.4 Discussion

Time synchronization testing was conducted to measure the timing offsets of the local oscillators in the receiver array. Two different methods were used to determine the level of synchronization. One compared the phase offsets of a sine wave that was transmitted using a signal generator. The second method used an oscilloscope to plot the ADC clock waveform generated by each radio. As stated in the introduction one of the project deliverables was to have the radios synchronized within 5 ns of each other. The histogram results showed that the time offset did not exceed 1.3 ns, which is well within the 5 ns margin. The mean time offset between all four radios was -0.891 ns with an average standard deviation of 0.028 ns. The minimum and maximum time offsets were -1.287 ns and -1.507 ns, respectively.

The second test which directly recorded the ADC clock waveforms generated by each radio also showed the USRP2s did not drift by more than 2 ns. After running both of these tests the data are conclusive that the receiver array was synchronized within the project deliverable of 5 ns. Another test that could be performed in the future is measuring the drift of the 1 PPS signal, this would indicate whether the GPSDO was contributing to the overall clock drift in the system. The possibility of a drifting PPS clock

could explain why the variance of the second test was so much greater than the variance of the first. In the first test, the clock drift was measured relative to each of the other radios. If the PPS clock started drifting, each of the local oscillators would also start to drift. The time offset between radios relative to the first remains constant. But when referenced to an external clock independent of each radio's local oscillator, as is the case in the second test, the drift is observed and hence characterized by an increase in variance.

It is possible that the results could have been improved if a more precise GPSDO was used instead of the Jackson Labs unit. Also, further research and investigation into the phase locked loop in the radio's FPGA could lead to software modifications to boost performance. Making these two changes could slightly improve the synchronization between radios.

3.3 Radar processing

The purpose of this section is to describe how the radar processing techniques were implemented in our radar system. This section recaps the radar processing methods mentioned in the background, explains how they were implemented in Matlab, discusses the methods used to test the radar receiver, and provides the results achieved with both simulated and recorded data. Note that all six of the USRP2s were used for our radar system, as shown in Figure 33, not just the four which were used for time synchronization testing.

3.3.1 Matlab Implementation

All of the radar processing methods for this project's receive system were conducted in Matlab. Received data from the USRP2s was recorded onto the computer using GNU Radio. The data from each of the six radios was imported into Matlab to create a six channel array. As shown in Figure 46, the radar processing in Matlab was broken into four steps: range processing, Doppler processing, direction finding, and calibration. Calibration is depicted as a loop because it compares the acquired results (empirical) with expected results (calculated) to make adjustments as necessary. Upon completion of these four steps, the output plots reveal the range, speed, and direction of targets that may be present in the recorded data. The following sub-sections describe how each of these steps works.

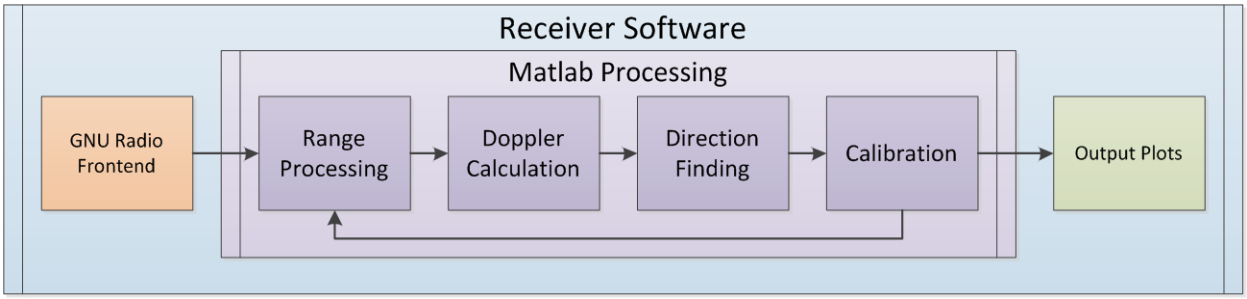


Figure 46: Radar processing software flow diagram

3.3.1.1 Data Preparation

Before radar processing can begin in Matlab, the data exported from the USRP2 (saved on the computer) must first be rearranged for more efficient processing. GNU Radio saves the recorded data from each channel as one long vector of repeating chirps. However, Matlab processing is more efficient if these chirps are stacked next to each other in a matrix, as shown in Figure 47. The columns of the matrix consist of the individual frequency sweeps (chirps) and the rows consist of the range cells, or elements of the matrix that correlate to a specific distance from the receiver. These configurations will be further explained in the following sections.

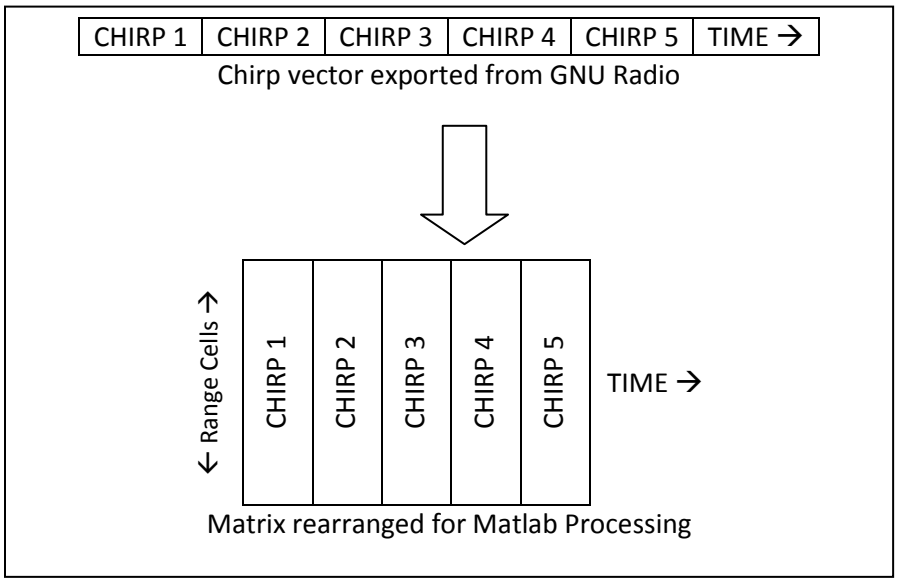


Figure 47: Vector rearrangement. The data vector exported from GNU Radio is rearranged into a matrix to optimize the processing conducted in Matlab. The number of columns in the matrix is equal to the number of chirps and the number of rows is equal to the number of samples in each chirp.

When reshaping the matrix, it is not necessary to know when each chirp starts and stops. Figure 47 shows a general case of how the chirps are rearranged. However, chirps can overlap between adjacent columns because calibration with a known target can correct any offsets. After proper calibration, which will be discussed later in this section, radar processing conducted on the matrix shown in Figure 48 will have the same results as processing conducted on the matrix in Figure 47. In Figure 47, Chirp 2 is in Column 2. However, radar processing will still work even if Column 2 is composed of the last 397 samples of Chirp 1 and the first 103 samples of Chirp 2, for example.

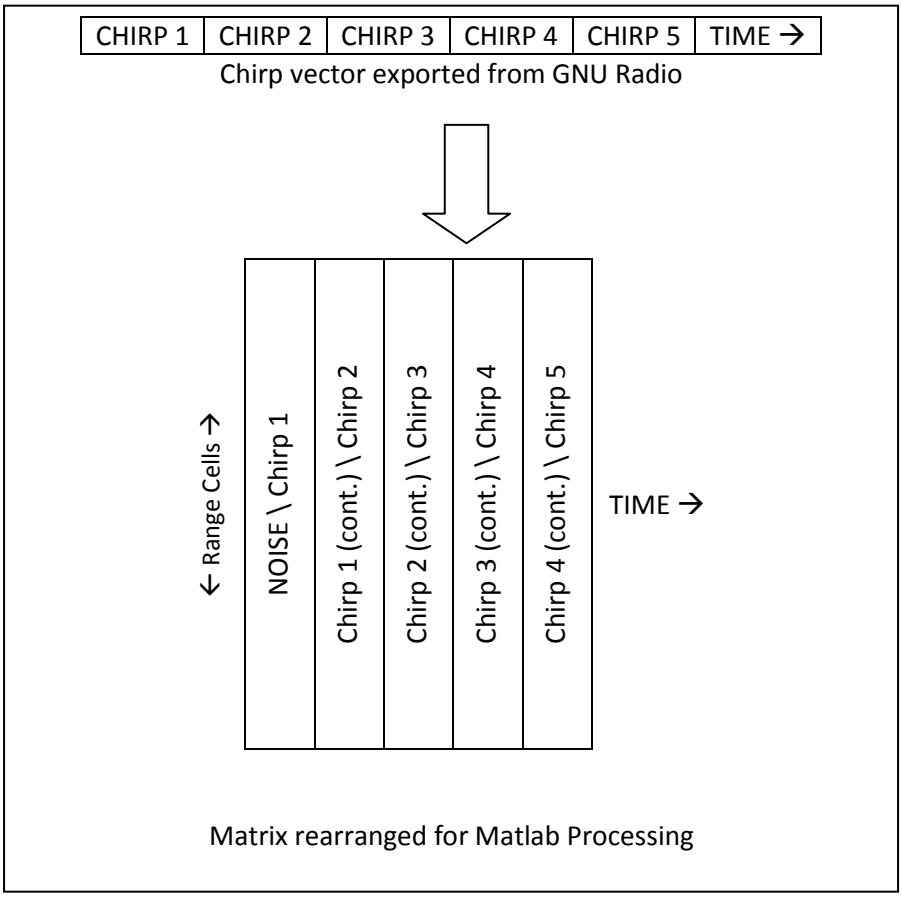


Figure 48: Offset matrix

3.3.1.2 Range Processing

To determine the range in Matlab, we must identify the time delay between the transmitted and the received signals. The time delay can only be determined if the transmitter and receivers are synchronized. For this project, our transmitter and receivers were synchronized to a GPS clock, the same method used for Group 33's arrays. If the clocks are not synchronized, the time delay can still be

computed but requires calibration with a known target. This method will be further explained in the Calibration section.

Computing the time delay can be achieved by replicating the transmitted chirp (Figure 13) and correlating it with the received signal. Autocorrelation of the chirp creates a signal whose magnitude exhibits a very distinct peak (see Figure 49), a desired characteristic which can help identify a target obscured in a noisy signal. When the prototype transmit chirp is correlated against each received chirp (each column of the data matrix), this peak creates a ridge across a row of the matrix, as shown in Figure 51. This ridge correlates to a specific range cell which corresponds to the distance of the target from the receiver.

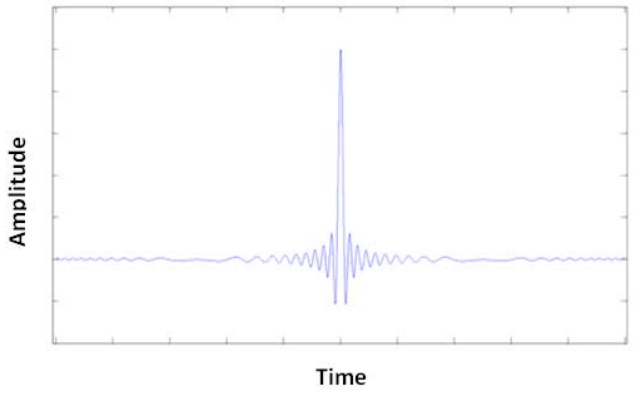


Figure 49: Chirp autocorrelation

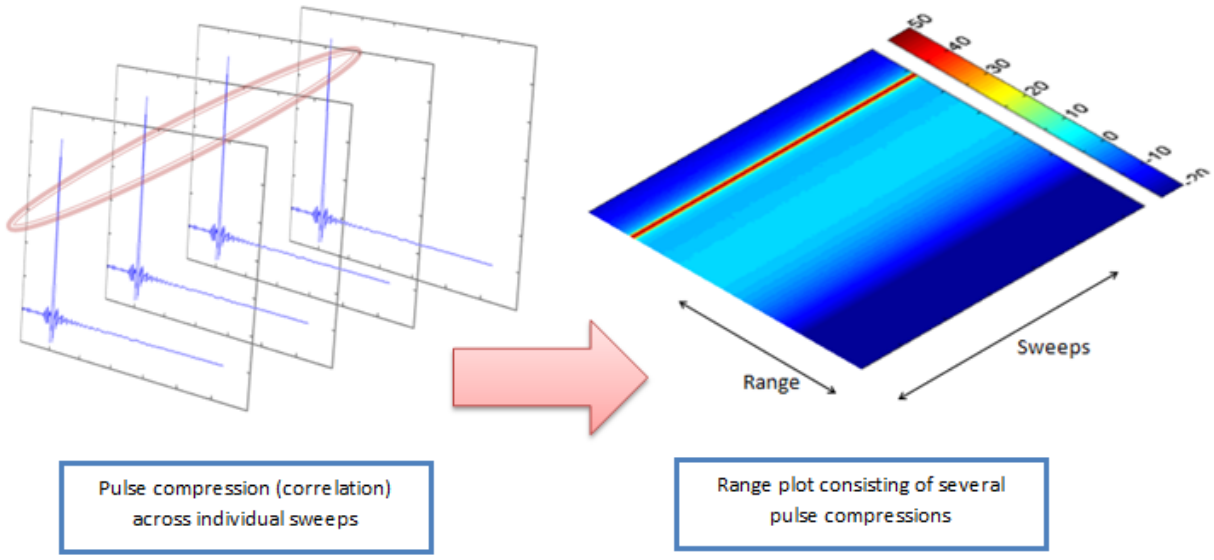


Figure 50: Creating a Range plot from repeated pulse compression. After stacking the pulse compressed chirps, the individual peaks create a ridge (shown in red) representing the target's range cell. Range processing requires the transmitter and receiver to be synchronized to a common reference. For the arrays used by Group 33 as well as our receive array, a GPS clock served as our reference.

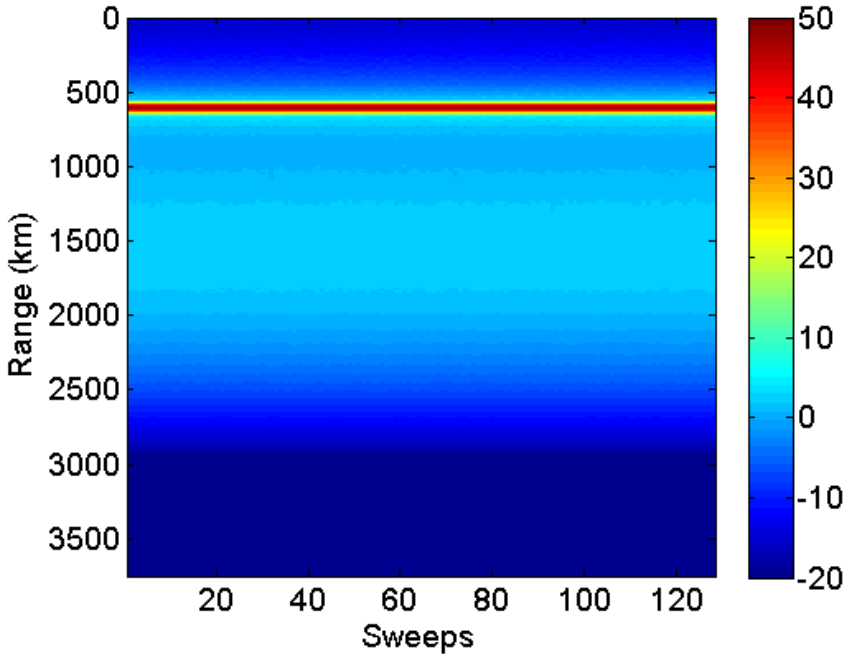


Figure 51: Range of a target. This Range plot generated by data simulated in Matlab shows a target located approximately 600 km from the receiver. The ridge shown (in red) is formed by the correlation of the chirp with each column of the matrix.

A range cell sets an upper and lower boundary for the range of a target. The range cell resolution can be determined by dividing the maximum unambiguous range by the number of samples per sweep. For example, the Matlab simulated data used to generate the range plot in Figure 51 had a simulated waveform repetition frequency of 40 Hz (40 chirps transmitted every second) and 625 samples per chirp. The maximum unambiguous range is about 3750 km so the range cell resolution is 6 km. The first range cell for this radar is 0 – 6 km; if the peak created by correlation appears within this range cell, it means the target is 0 – 6 km away from the receiver.

3.3.1.3 Doppler Processing

To determine a target's rate of movement using Matlab, we need to determine the Doppler shift frequency which is the phase progression from sweep to sweep. The phase between chirps changes periodically with frequency *f*. This frequency can be found by taking the Fourier transform across each range cell (across each row of the Range plot shown in Figure 51). Because the Fourier transform of a periodic function generates a delta function, δ(F), at each harmonic, a peak will occur along the range ridge at a particular Doppler cell that corresponds to the target's speed. A Doppler cell is similar to a range cell except the boundaries are a measure of radial velocity. The resulting Doppler-Range plot reveals a target at the intersection of the Doppler cell and the range cell, as shown in Figure 52.

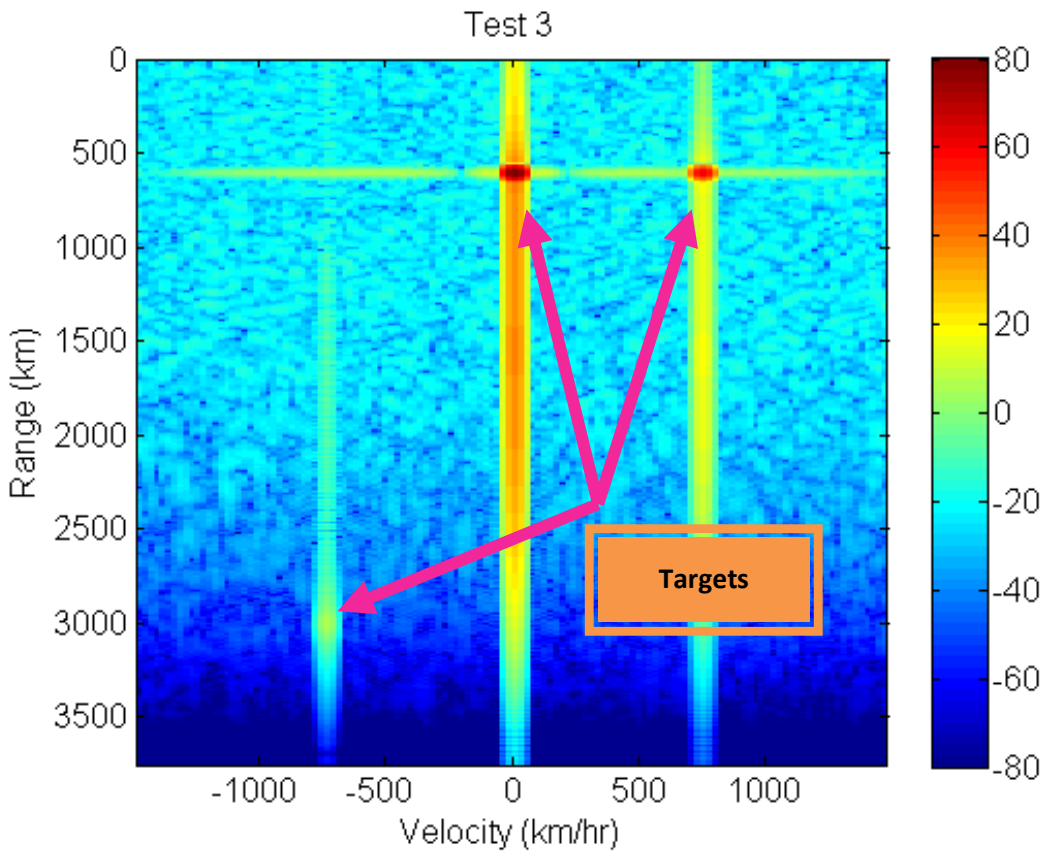


Figure 52: Range-Doppler plot for data simulated in Matlab. Targets are indicated by high intensity points and are identified by the arrows in the figure. The three targets are located at approximately 3000 km moving away from the receiver at 750 km/hr, 600 km moving at 0 km/ hr (stationary), and 600 km moving towards the receiver at 750 km/hr.

The maximum observable (unambiguous) velocity, V_{max} can be related to the waveform repetition frequency (WRF), F_W and the center frequency, F_C of the chirp by

$$V_{max} = \frac{c}{F_C} * \frac{F_W}{2}$$

where c is the speed of light (Skolnik). The derivation of the formula can be found in Appendix 5.1: Doppler Frequency Shift. As the WRF increases, so does the maximum observable velocity. The faster an object moves, the greater the phase shifts between samples. Because the WRF is sampling the waveform generated by these phase shifts, it must remain above the Nyquist frequency to avoid aliasing. Thus, a higher WRF is necessary to view fast targets. The data set used to generate Figure 52 is the same one used for Figure 51; with a WRF of 40 Hz and simulated center frequency of 18 MHz, the maximum observable velocity is approximately 1,500 km/hr.

3.3.1.4 Direction Finding

Range and Doppler processing are conducted on the demodulated chirps acquired from each antenna in the array. The data set from each of the six antennas produces its own Range-Doppler plot which can be used to distinguish targets. If each of the six Range-Doppler plots is rearranged into a three-dimensional matrix, as shown in Figure 53, the individual targets appear at approximately the same range and Doppler cells in each of the plots. For example, if a target appears at range cell 17 and Doppler cell 57 of the first plot, the same target appears at approximately range cell 17 and Doppler cell 57 in all six plots. Direction finding requires a target vector containing complex samples from each of these six cells. Note that Range-Doppler plots only show the magnitude of the complex signals located at each cell but direction finding uses the complex value consisting of both magnitude and phase.

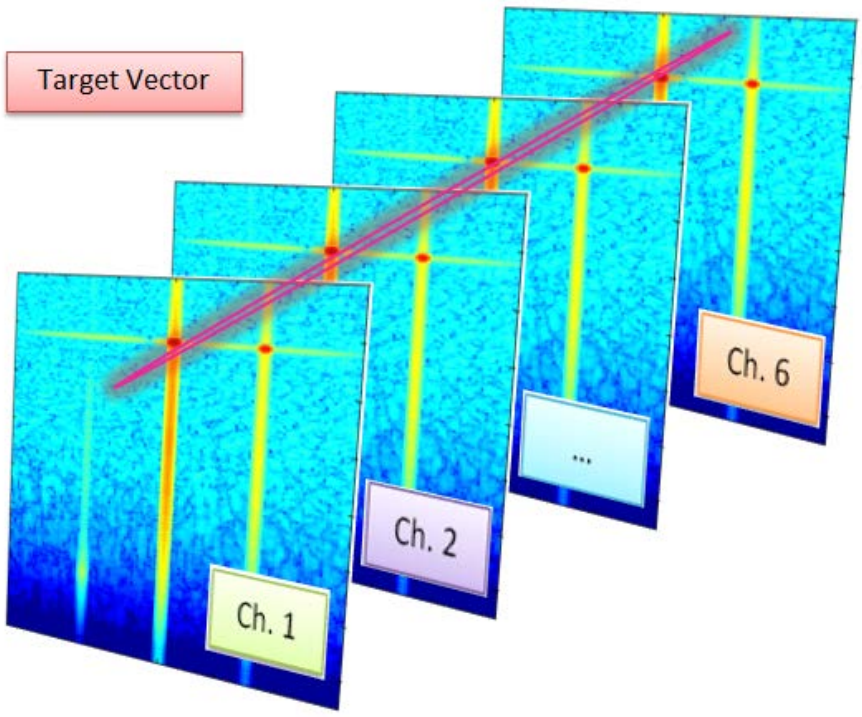


Figure 53: The target vector is acquired from the same range and Doppler cells of each channel. Because there are six channels in the array, the target vector is comprised of six complex values.

Direction can be identified by computing the phase difference of these six points with respect to the first. In this system, the six antennas are equally spaced so the phase difference between adjacent antennas is ideally the same for the entire array. E.g. the phase difference between the samples acquired from antennas 1 and 2 is approximately the difference between antennas 5 and 6. Similarly, the phase difference between the samples acquired from antennas 1 and 3 is approximately twice the

difference between antennas 1 and 2. So if we use the first antenna as a reference, we can determine the expected phase at each n^{th} element by

$$\Phi(n) = \frac{2\pi}{\lambda} d(n-1)\sin\theta, n = 1, 2, \dots, 6$$

where ϕ(n) is the phase of the received sample from the n^{th} antenna with respect to antenna 1, λ is the wavelength of the carrier frequency, *d* is the distance between antennas, and theta (Θ) is the incident angle of the incoming signal, as shown in Figure 25. Note that the expression $d(n-1)$ represents the total distance of the n^{th} antenna from antenna 1. I.E. the distance between antennas 1 and 4 is $d(4-1) = 3d$.

Each of the six complex values that comprise the target vector has both magnitude and direction; the target vector is essentially comprised of six vectors, one from each Range-Doppler plot. When adding each of these points, the greatest magnitude is achieved when they all have the same direction. Each of these points has magnitude *M* (which correlates to the strength of the signal) and angle ϕ (which corresponds to the signal's phase). If each complex point is of the form $Me^{j\phi}$, then the magnitude can be calculated by multiplying by $e^{-j\phi}$, a vector with magnitude equal to one and with a phase that is the negative of that of the original point. For the rest of this section, we will refer to this vector as the "zeroing vector" because its product has an angle of zero.

The method of direction finding used in this project is best explained in the following example. Suppose the data sample acquired from the second antenna (the second point in the target vector) is $M_2 e^{j\phi_2}$; the zeroing vector is $e^{-j\phi_2}$. Now, let's express phi (ϕ_2) in terms of the angle of the incoming signal, theta (Θ). The zeroing vector now looks like $e^{\frac{2\pi}{\lambda}d\,(2-1)\sin\theta}$ where λ is the signal wavelength, *d* is the distance between the first and second antennas, and Θ is the angle of the incoming signal. However, theta is unknown so we create a theta vector defined from -90 degrees to 90 degrees in increments of one degree ($\theta = -90°:90°$). When $e^{\frac{2\pi}{\lambda}d\,(2-1)\sin\theta}$ is expressed for all theta values, the result is a 181 point vector of candidate zeroing vectors, one for each incoming angle from -90 degrees to 90 degrees. The correct zeroing vector, when multiplied by the data sample $M_2 e^{j\phi_2}$, yields the largest magnitude. The index of this zeroing vector is the index of theta which is the angle of the incoming signal.

This process is repeated for all points in the target vector, as outlined in Figure 54. First, a candidate zeroing vector matrix is defined for Θ from -90° to 90°. Next, this is multiplied by the acquired data

vector to find the value of theta that maximizes the sum of the magnitude from the *n* channels. A relative maximum appears at the value theta that generates the best zeroing vector for all channels in the array. This theta value is the incident angle of the target.

Target vector (1 x 6) created from the 6 Range-Doppler plots

Candidate zeroing matrix (6 x 181)

$$\begin{bmatrix} M_1 e^{j\phi_1} \\ M_2 e^{j\phi_2} \\ \vdots \\ M_6 e^{j\phi_6} \end{bmatrix}^T * \begin{bmatrix} e^{-j\alpha(1,-90°)} & e^{-j\alpha(1,-89°)} & \dots & e^{-j\alpha(1,90°)} \\ e^{-j\alpha(2,-90°)} & & & \vdots \\ \vdots & & & e^{-j\alpha(5,90°)} \\ e^{-j\alpha(6,-90°)} & \dots & e^{-j\alpha(1,89°)} & e^{-j\alpha(6,90°)} \end{bmatrix}$$

$$\alpha(n,\theta) = \frac{2\pi}{\lambda} d(n-1)\sin\theta\,, \theta = -90°{:}\,90°, n = 1{:}\,6$$

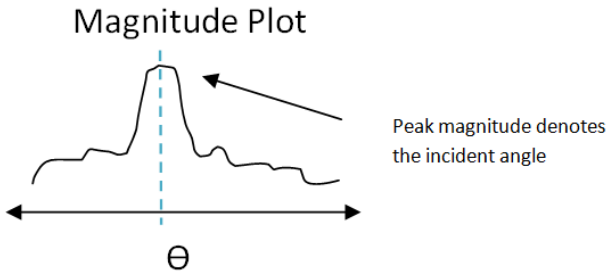


Figure 54: Direction finding process: the received vector is multiplied by a control matrix that sweeps Θ values from -90°:90°. The matrix rows are summed and plot against Θ. A peak will occur at the Θ value that maximizes the magnitude.

3.3.1.5 Calibration

Calibration is necessary to adjust for timing offsets between the transmitter and receiver as well as factors that affect the direction finding capabilities such as unequal cable lengths or uneven antenna gain. Calibration requires the use of a test target with known range and direction relative to the receiver. For this project, the transmitter was used because it met these criteria. Calibration is a type of feedback loop in that it compares the known (calculated) range and direction of a target to the empirical range and direction of the same target. Adjustments are then applied to the incoming data to correct any discrepancies.

Range is calibrated by shifting the range cell of the transmitter to the correct range cell. For this project, the transmitter is very close to the receiver (< 100 m) so it is expected to appear in the first range cell. The transmitter as it appears on the Range-Doppler plot serves as a reference and all other features can be measured relative to the transmitter in order to gain an accurate reading. For example, suppose the

transmitter appears at range cell four and an aircraft appears at range cell nine. The aircraft is five range cells away from the receiver.

Direction is calibrated by measuring the phase of the received test signal, determining the difference relative to the known phase, and adding this offset to the phase of each subsequent sample. For example, suppose the test signal is located at an angle of 10° but the phase of the received signals indicates it is at 17°. The expected phase for a 7° difference at each element *n* is subtracted from each sample of the n^{th} channel. Thus, the resulting phase ϕ for all subsequent channels is:

$$\Phi(n) = \frac{2\pi}{\lambda} d(n-1)\,(\sin\theta - \sin 7°)$$

3.3.2 Methods and Results

This section describes the experiments conducted to test our radar processing scripts and make improvements. We conducted initial tests with simulated data generated in Matlab to test the functionality of the range and Doppler processing components. Once our scripts produced results similar to those generated by our sponsors, we had confidence to test our radar processing with data recorded from our USRP2 receive array.

The data generated in Matlab simulated a single channel (one antenna) and was used to test the range and Doppler processing functions. Because multiple channels were necessary to determine direction, we could not fully test our direction finding scripts without actual data. After recording data from our linear array of six USRP2s, we could test our direction finding functions, determine the functionality of our receive array, and optimize each of our processing scripts. In each of these experiments, processing was conducted with knowledge of the center frequency of the transmitted signal, the chirp bandwidth, and the waveform repetition frequency (WRF).

3.3.2.1 Simulated Data

Simulated data were first used to test the range and Doppler functionality of the receiver. Our processing scripts were tested with three scenarios, each increasing in difficulty. The first data set had only one stationary target and tested our range processing. As shown in Figure 55, there is no noise and the target is clearly identified by the relative power difference. Because this target appears in the center of the x-axis, it means that it is stationary. If the target was moving towards the receiver, there would be a positive Doppler shift and it would appear on the right. If the target was moving away from the receiver, there would be a negative Doppler shift and it would appear on the left.

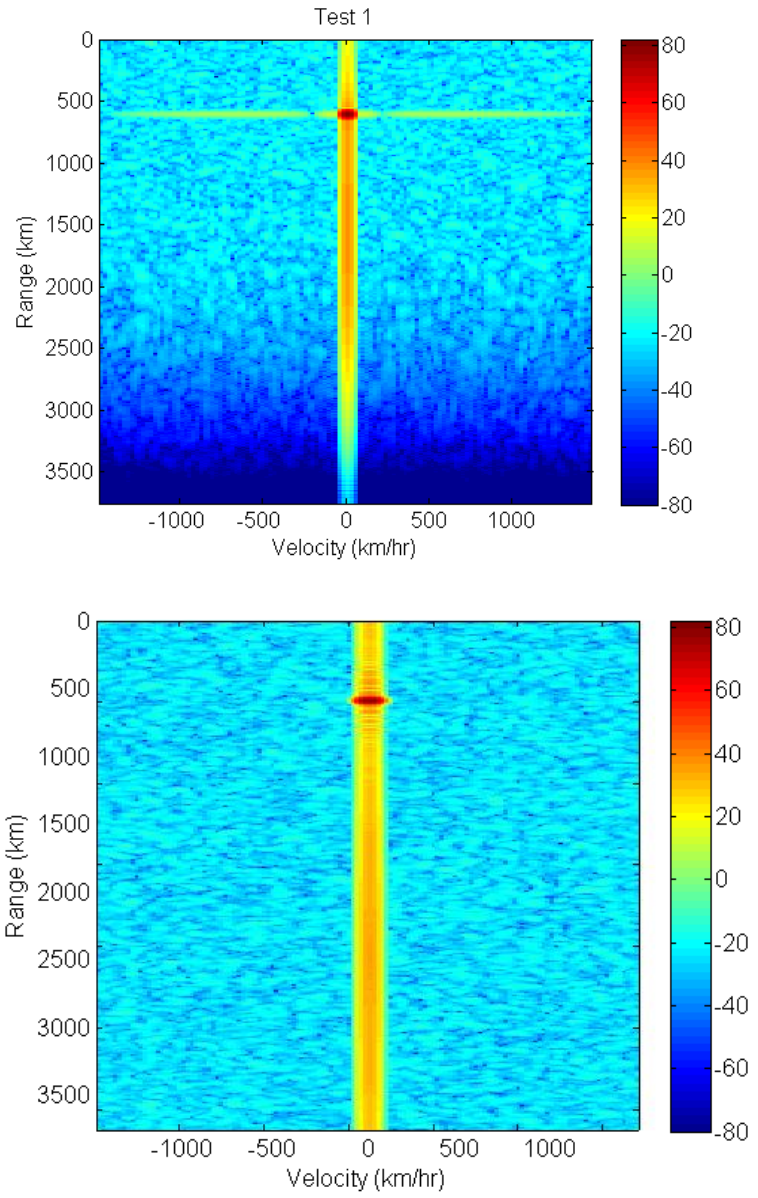


Figure 55: Simulated data test 1; single stationary target. Top plot shows the Range-Doppler plot created using our radar processing scripts. Bottom plot shows the Range-Doppler plot created using the same data but our sponsor's processing. Both show a target located at 600 km moving at 0 km/hr. A positive velocity indicates the target is moving towards the receiver while a negative velocity indicates it is moving away.

Once the range processing script was working, we tested our Doppler processing with the second data set which had a single moving target. The addition of the Doppler shift is indicated by the off-center target, as shown in Figure 56. Finally, the third data set included multiple targets each at different ranges and speeds. The resulting plots are shown in Figure 57.

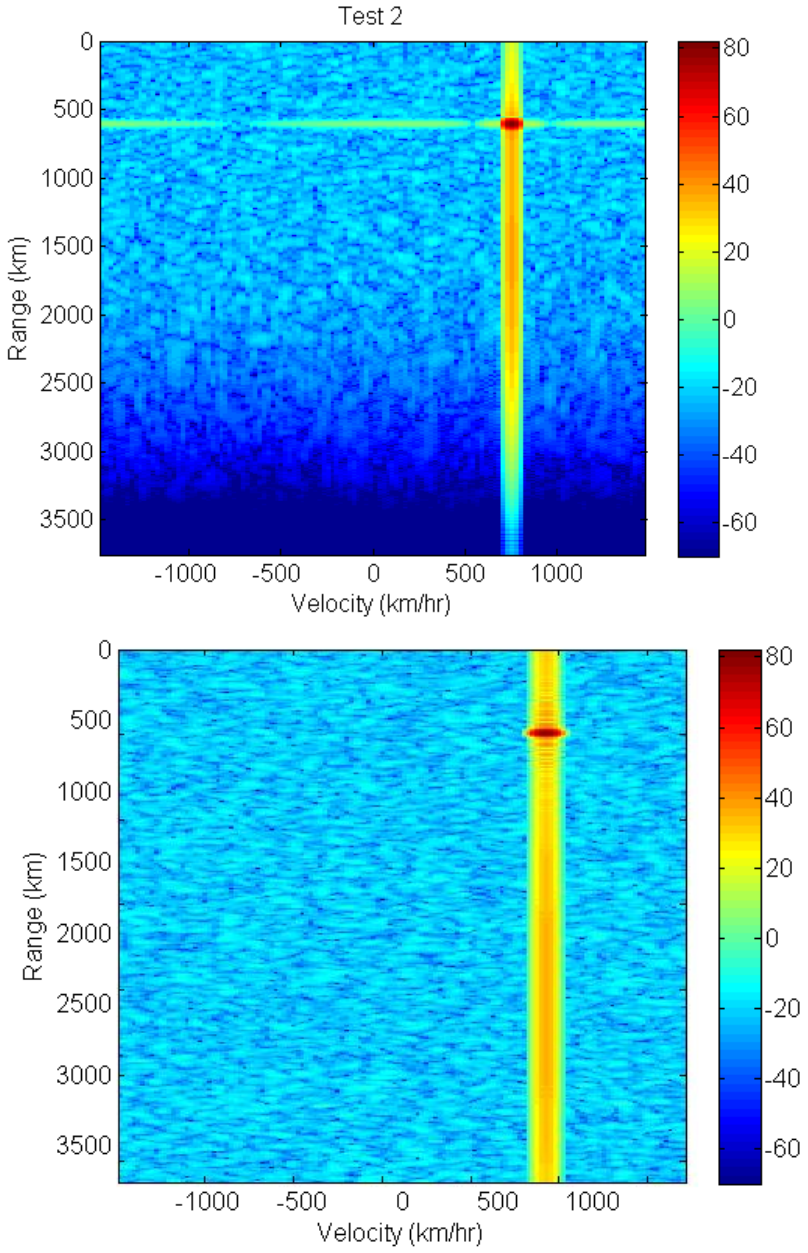


Figure 56: Simulated data test 2: addition of a Doppler shift to a single moving target. Top plot shows the Range-Doppler plot created using our radar processing scripts. Bottom plot shows the Range-Doppler plot created using the same data but our sponsor's processing. Both show a target located at 600 km moving at 7500 km/hr away from the receiver.

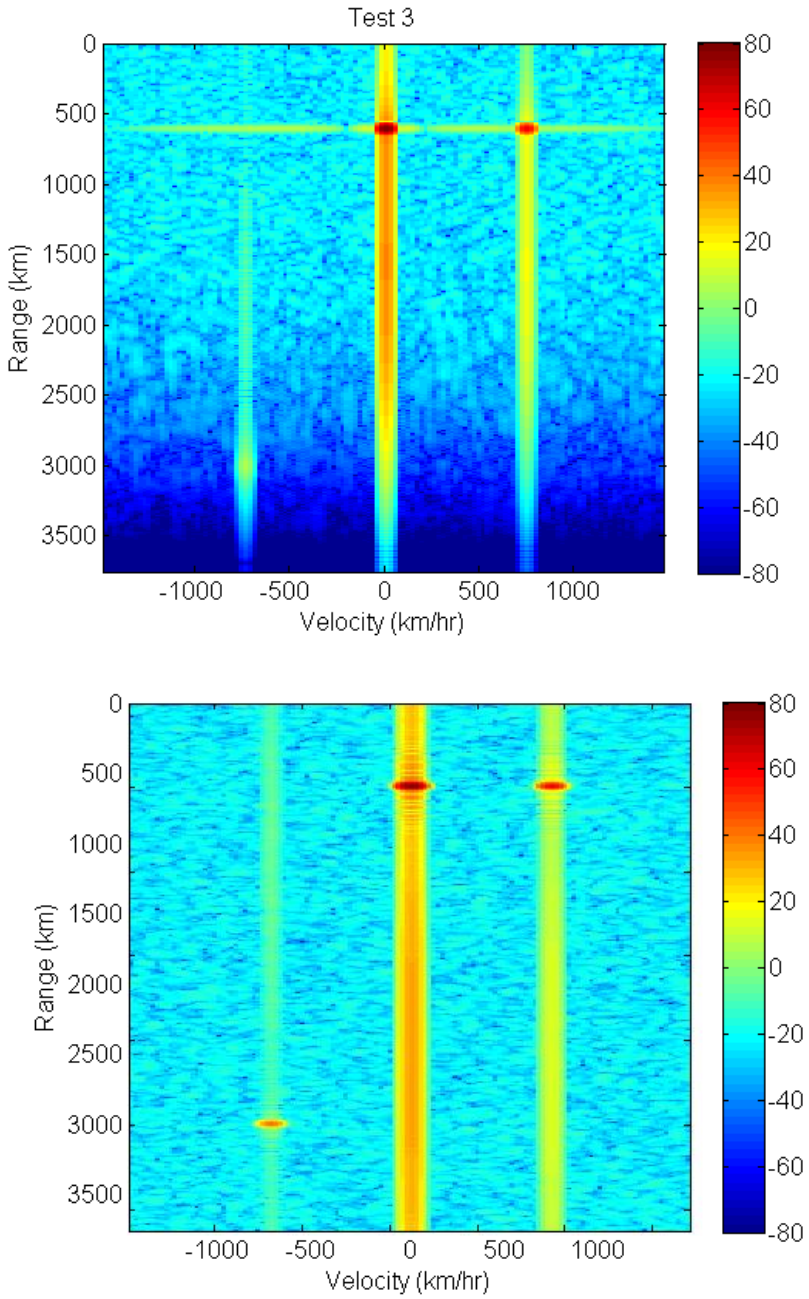


Figure 57: Simulated data test 3 - multi target. This test shows three targets. There are two targets at 600 km; one is moving toward the receiver at 750 km/hr and the other is stationary. There is also a distant target at 3000 km moving away from the receiver at 750 km/hr. Note that the returned power from this target is much weaker because it is five times as far away. Top plot shows the Range-Doppler plot created using our radar processing scripts. Bottom plot shows the Range-Doppler plot created using the same data but our sponsor's processing.

3.3.2.2 USRP2 Recorded Data

The other tests we conducted were with data recorded from our own USRP2 receive array. The purpose of these experiments was to test the functionality of our entire radar system, both hardware and software. The previous test helped to support that our range and Doppler processing scripts were

correct, however it did not provide any insight into our direction finding or calibration scripts. However we could test the script with our own recorded data because we knew the direction of targets we acquired, namely our transmitter which served as our calibration target.

3.3.2.2.1 Set Up

Before we could begin the tests, we had to set up the array. Six receive antennas were placed in a linear array, as shown in Figure 58. The antennas were equally spaced 6.5 meters from each other, the maximum distance for receiving a 23 MHz signal without spatial aliasing (the wavelength λ of a 23 MHz signal is 13 meters). The transmit antenna was located on a 15 m tower and had an output power of 70 watts.

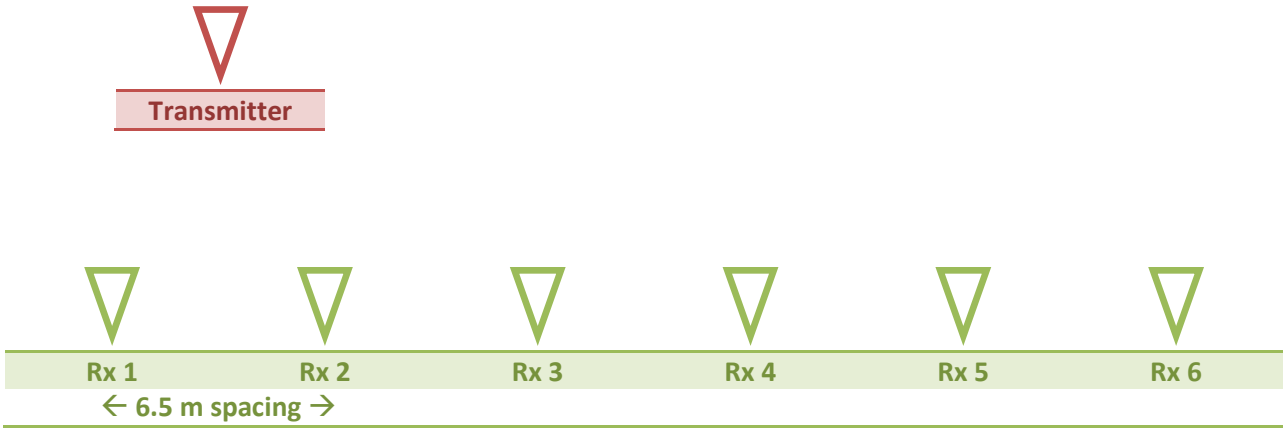


Figure 58: Transmitter and receiver array configuration. Each antenna was equally spaced 6.5 meters away from each other.

3.3.2.2.2 Direction finding from a fixed transmitter

We used our transmitter as a test signal for our receive array to determine whether our direction finding scripts were correct. Although we did not have the flexibility to move the transmitter or the receiver to test direction finding, we could change the incident angle of the transmission relative to the array by only receiving on specific antennas because the transmitter was close (< 100 m) to the receiver. The process is depicted in Figure 59. Our receive array consists of six radios however not all six are necessary for direction finding. To test our direction finding scripts, we divided the array into three overlapping four-element arrays. The first array consists of antennas 1 through 4, the second contains antennas 2 through 5, and the third contains antennas 3 through 6.

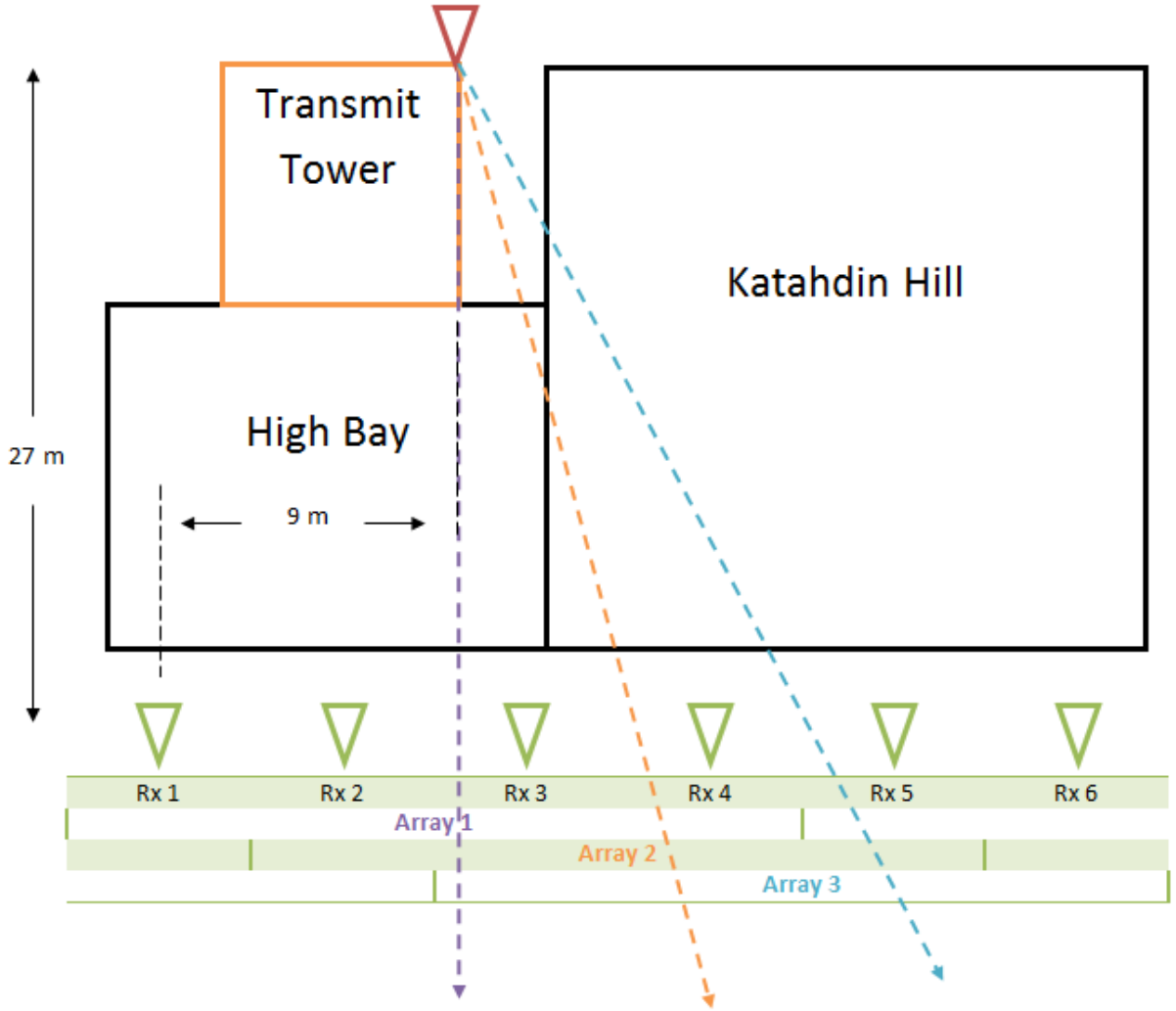


Figure 59: Direction finding testing method. The 6 antennas are divided into four smaller receive arrays, each shifted to the left by 6.5 meters. The incident angle of the transmitted signal is different for each array and can be used to test if the direction finding script is functioning properly.

Each of these arrays is shifted over by 6.5 meters, the length between two adjacent antennas. As a result, the center of the array, which is the reference point used to measure the incident angle, also shifts by 6.5 meters. Using simple geometry, we calculated the expected incident angle at the midpoint of each of these arrays, as shown in Table 7. The direction finding ("theta") plot for each array is shown in Figure 60 - Figure 62. The vector magnitude sum is plot against values of theta. The direction of the signal is denoted by the local maximum (peak) of each plot.

Table 7: Direction finding angles: calculated vs. empirical

Array	Calculated Incident Angle	Empirical Incident Angle	Error
1 (Antennas 1-4)	17°	18°	1°
2 (Antennas 2-5)	28°	31°	3°
3 (Antennas 3-6)	37°	46°	9°

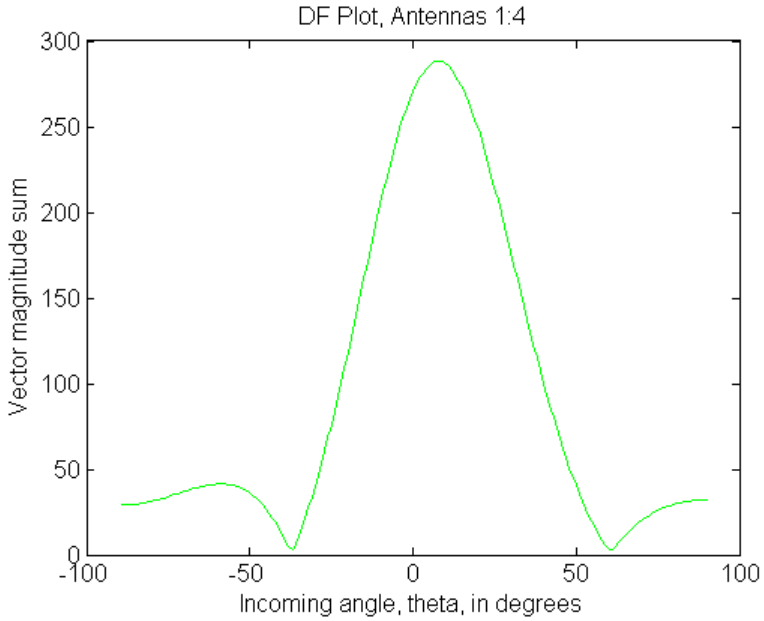


Figure 60: Direction finding plot for the first array.

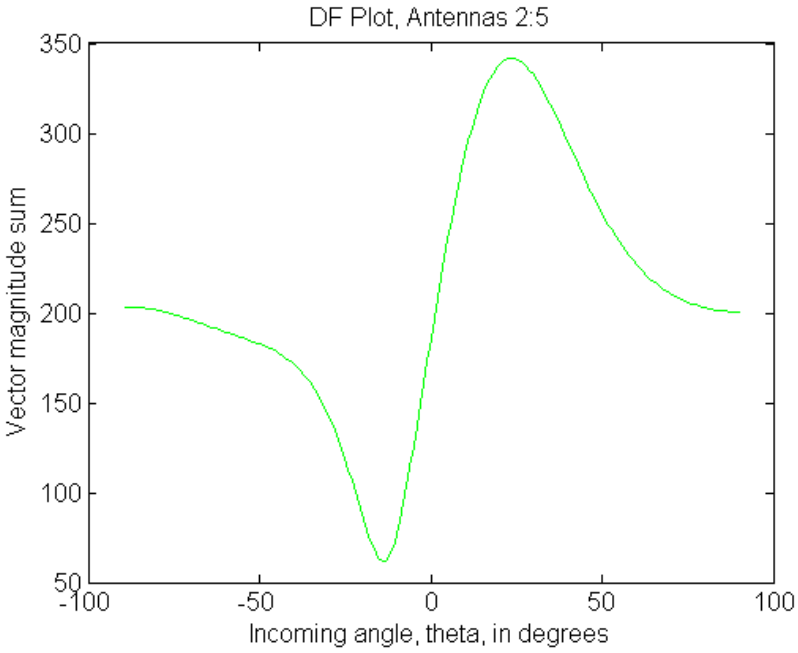


Figure 61: Direction finding plot for the second array.

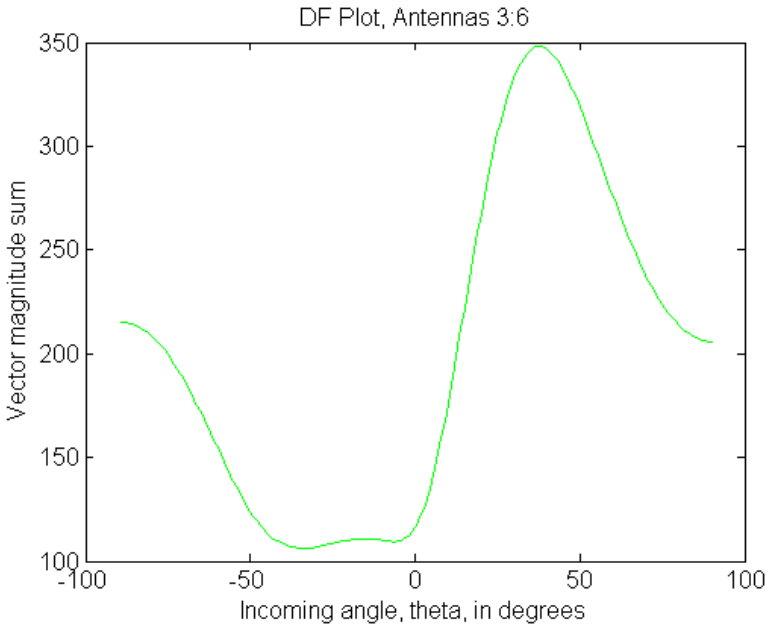


Figure 62: Direction finding plot for the third array.

Note that the error increased as the array was "shifted" through the six antennas. One of the limitations of this method was that only four of the six antennas in the system could be calibrated. Initially only antennas 1-4 were properly calibrated. So for the second array (antennas 2-5), only three of the four antennas were calibrated and for the third array (antennas 3-6), only half of the antennas were calibrated.

Another factor contributing to the error is the close distance between the transmitter and receiver array. Our method of direction finding assumes the receive array is far away from the transmitter (in the transmitting antenna's far field). The far field boundary distance *R* from the transmitter can be estimated by $R = \frac{D^2}{\lambda}$ where D is the array length (approximately 32.5 m) and λ is the carrier wavelength (about 13 m) (Skolnik). Thus R can be approximated to be 81.25 m, about three times the distance between the transmitter and receiver array in this experiment. This short distance most likely affected the direction calculations and contributed to the error.

Despite the errors, we were able observe a consistent trend across the three arrays that was consistent with the expected results. This experiment helps to support that our direction finding scripts were operating correctly.

3.3.2.2.3 Direction finding from a moving transmitter

Group 33 owns a truck with a mobile transmit station. The same transmitter and antenna used in the previous experiment were mounted on the truck shown in Figure 63. Unlike the transmit tower, the transmitter on the truck could be placed at different locations. In addition, the transmitter could operate while the truck was moving which allowed us to test the Doppler processing and direction finding at the same time.

Figure 63: Group 33's mobile transmit station. The BAE transmitter and antenna are the same used in the previous test.

For the first test, we placed the truck in the far field nearly centered with the array. Because it was difficult to place the truck directly on the array normal line (line perpendicular to the array), the truck was off centered by about five degrees. The purpose of this test was to demonstrate direction finding

under nearly ideal conditions: a stationary target in the far field of the array. In addition, we used the data collected from this experiment to calibrate the array for the next experiment when the truck was moving. The direction plot of the stationary transmitter is shown below in Figure 64.

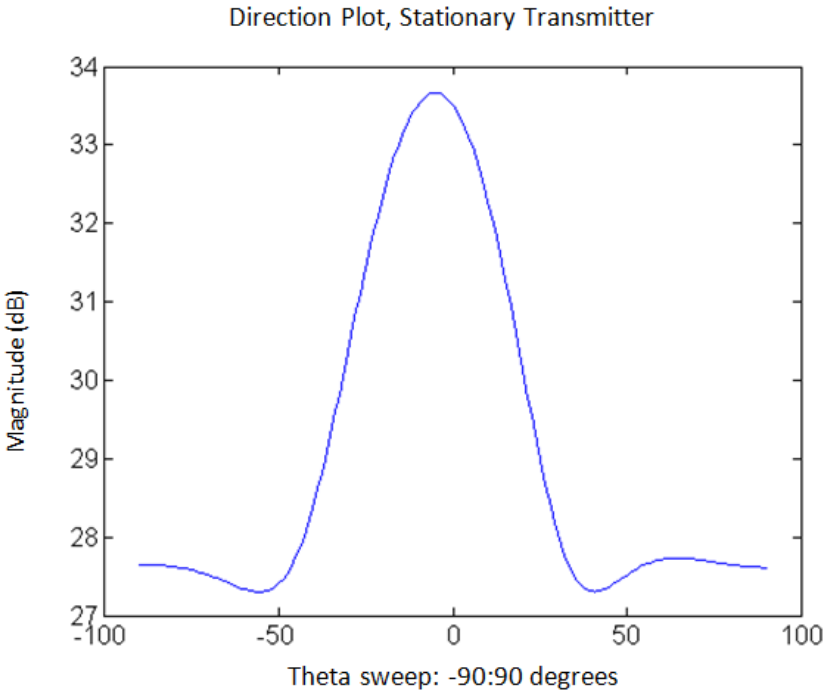


Figure 64: Direction plot of a stationary far field target located 5° off center from the midpoint of the array.

This figure shows that under ideal conditions our receive array can correctly locate a target with side lobe suppression of about 6 dB.

For the second test, the truck drove in the path depicted below in Figure 65. The following direction plots (Figure 66) were taken at one second intervals beginning when the truck started moving on the route. The strongest peak of each plot represents the angle of the transmitter, as shown. The other peaks are results of multipath reflections as well as side lobes. Each figure shows the sum of the magnitude of each antenna's signal on a dB scale.

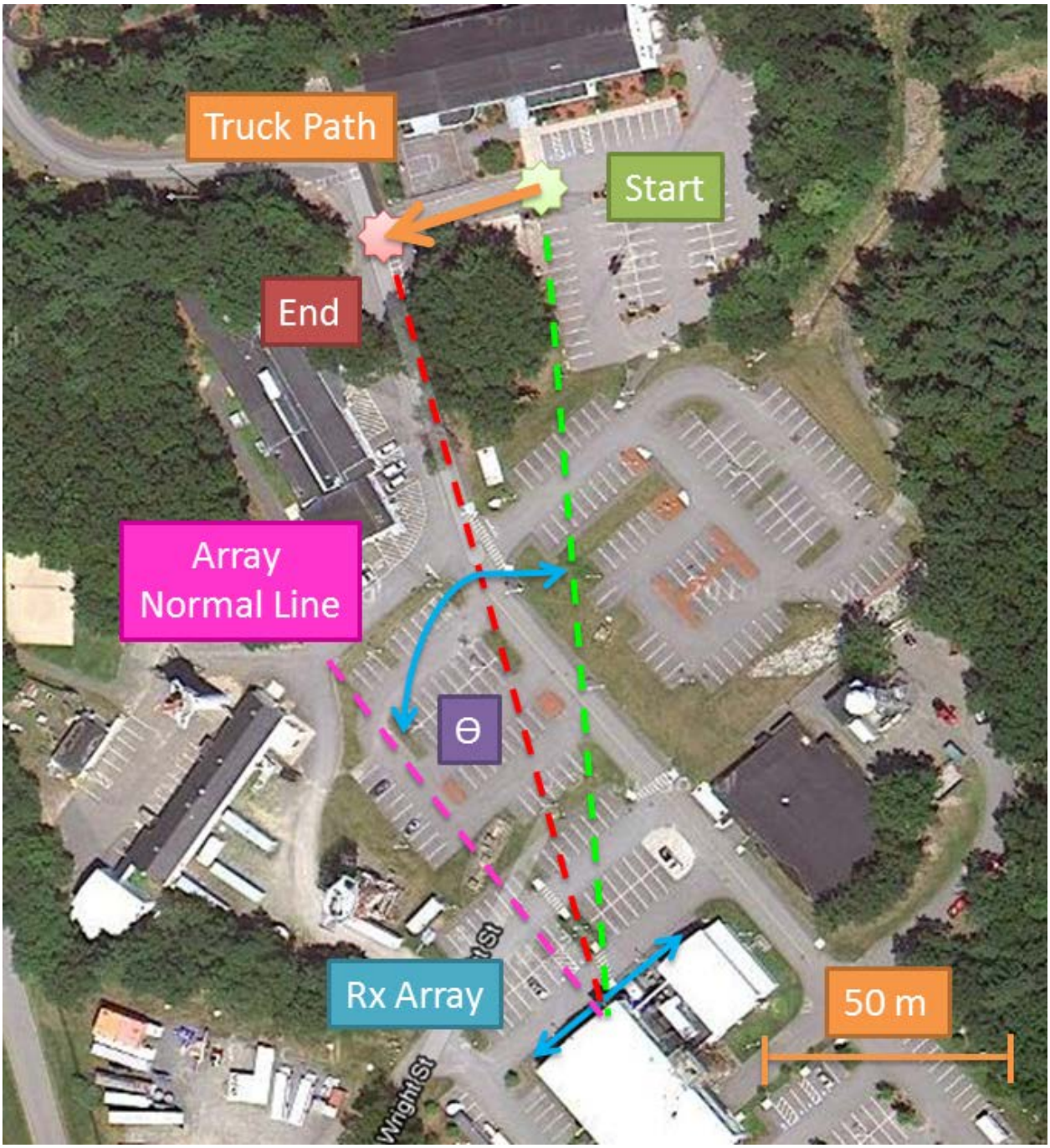


Figure 65: Path of Group 33's mobile transmit truck. Google Maps view of Katahdin Hill vicinity of Lincoln Laboratory facility in Lincoln, MA.

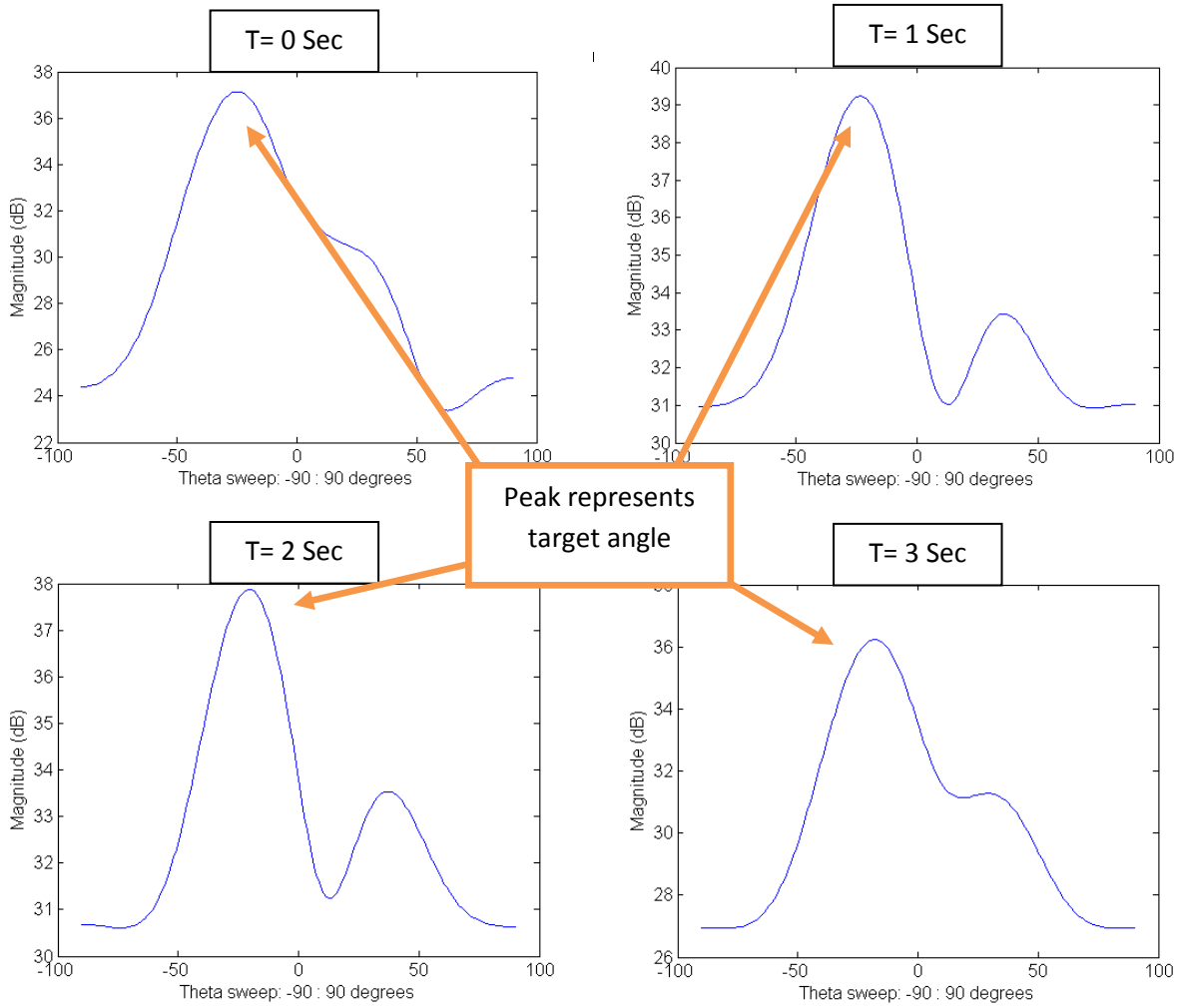


Figure 66: Direction of the moving transmitter. The peak of each plot denotes the angle of the incoming transmission at one second intervals. The figures are plot on a dB scale.

Figure 67 shows on overlap of the direction plots collected at 0.5 second intervals over three seconds. The peak of each plot is normalized to 0 dB in order to more clearly depict the change in direction. Over the three seconds, the angle changes about eight degrees, approximately 2.6 degrees per second. This equates to 27 km/hr, the approximate speed of the truck during the test.

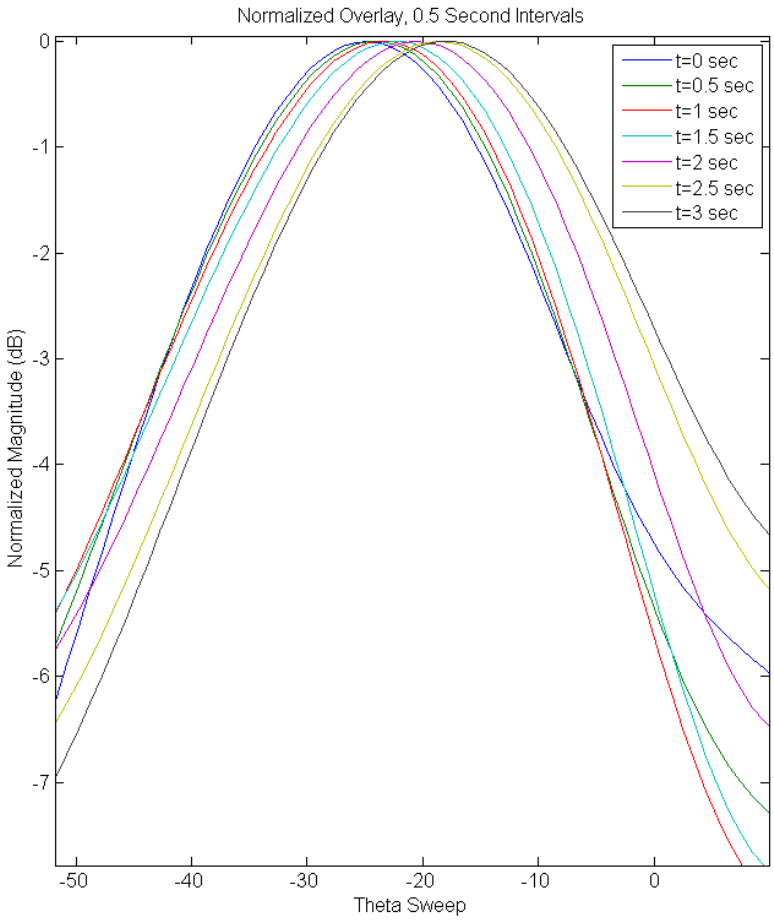


Figure 67: Normalized overlay of three seconds. To compare the peak over the three seconds (taken at 0.5 second intervals), each sample is normalized to the maximum value of the magnitude and plotted on a dB scale. The angle changes from -25° to -17° at an angular velocity of about 2.6°/sec.

The empirical and actual angles of the truck relative to the array normal line are shown below in Table 8. The error is approximately nine degrees and remains consistent over the three seconds. This error could be associated with poor calibration; improved calibration scripts could provide an offset that corrects this discrepancy. Errors arising from this test are most likely not due to the close proximity of the transmitter since it is in the far field of the receive array. However, during the test several vehicles crossed in front of our array including a school bus. Large metallic objects can reflect electromagnetic signals so it is possible one of these vehicles altered the incident angle of the transmitter. Despite the errors, this test shows that our receive array was able to identify a moving target and produce results with a trend that corresponds to the target's movement.

Table 8: Direction of the moving truck			
Time	Empirical Angle	Actual Angle (estimated)	Error
T=0 sec	25°	35°	10°
T=1 sec	23°	32°	9°
T=2 sec	20°	29°	9°
T=3 sec	17°	26°	9°

The operation of our receive array is further demonstrated with the following Range-Doppler plots which show the speed of the truck during this time interval. Note that this plot indicates the speed is only about 10 km/hr, not 27 km/hr. The Range-Doppler plot only shows the velocity component perpendicular to the receive array, also known as the radial velocity. Because the truck also has a velocity component parallel to the array, the Range-Doppler plot cannot accurately depict the absolute velocity.

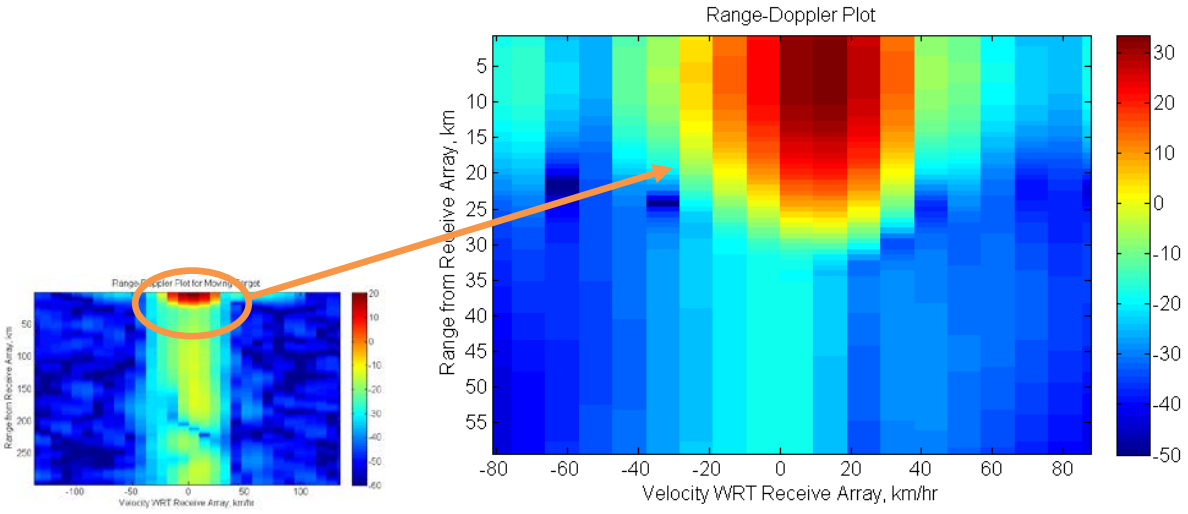


Figure 68: This Range-Doppler plot shows the transmitter moving at about 10 km/h. Note that this speed only represents the radial velocity not the absolute velocity of the truck.

In comparison a Range-Doppler plot of a stationary target is shown below in Figure 69. The peak of the signal appears slightly off-center because of the low Doppler cell resolution and because there is no range bin centered at 0 km/hr, just -10 to 0 km/hr and 0 to 10 km/hr.

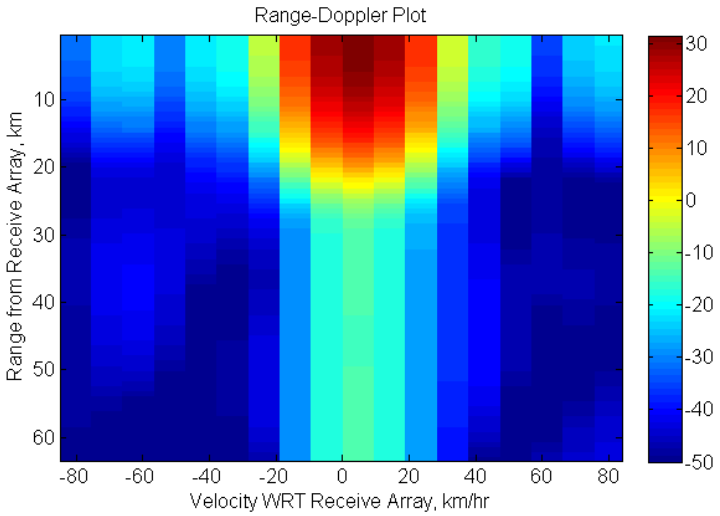


Figure 69: Range-Doppler plot of a stationary target

3.3.3 Discussion

The simulated and recorded data showed that our range and Doppler processing functions were operational. The plots created using simulated data showed that our range and Doppler processing scripts could operate with similar performance as our sponsor's scripts. Recording data from our own array helped to support that our Doppler scripts were functioning correctly. We were not able to fully test our range scripts because the computer controlling the transmitter at Lincoln Labs on the day of the test was not able to lock to the GPS reference clock. For this reason, the range of each data set was manually calibrated because we knew that the strongest signal was coming from the transmitter which was located in the first range cell. Had both the transmitter and receiver been locked to the GPS reference clock, this additional calibration would not have been necessary.

The preliminary direction results acquired when using the transmit tower showed a trend that correlated to the expected results. Although there were errors, these were most likely associated with the close proximity of the transmitter and inaccurate calibration, as described previously. The second test, which used a signal transmitted from the parked truck, showed that it was possible to correctly determine the direction when the source is located in the far field. The final test, which used data recorded as the truck was driving through the parking lot, showed that our processing could identify the moving target and, despite errors possibly due to calibration and multipath interference, distinguish different incident angles which corresponded to the truck's movement over a period of time.

4 Conclusion

The purpose of this project was to develop the framework for an inexpensive phased receive radar array using software defined radios. The objective was to assess timing synchronization between software defined radios in an array and to deliver a working prototype that could determine the direction, range, and movement of a target during post-processing. Each of the project's five deliverables was met:

1. Timing: Each of the radios was synchronized to within 2 ns of each other, less than the 5 ns objective declared at the beginning of the project.
2. Calibration: A Matlab script identified phase offsets that are out of the user's control (e.g. offsets due to varying cable lengths or materials, differing clock oscillators, antenna differences, *etc.*).
3. Data processing: GNU Radio was used to record (in real time) the received signals from six antennas and save the raw data for post processing. The scripts written in Matlab generated Range-Doppler plots to show the distance and velocity of targets and direction plots to show the azimuth of the incoming transmissions.
4. Direction finding: Our receive array determined the azimuth of a target with an error not exceeding 15 degrees, less than the 30 degree objective.
5. Range finding: Our processing scripts met our objective by correctly identifying the range of targets in simulated data generated in Matlab to within several kilometers.

Six USRP2s were synchronized using a 10 MHz external reference clock and a 1 PPs signal. Using a test transmission signal and the USRP2 clock debug pins, we verified that the each radio in the array was synchronized to within 2 ns of each other using two different methods. Achieving this level of synchronization showed that is possible to construct a phased receive array using the USRP2. Existing radar processing techniques for determining a target's range, radial velocity, and direction were implemented in Matlab. Initial tests were conducted with simulated data to test the functionality and accuracy of our code. Once verified, we proceeded to use data recorded from our own receive array. Despite errors due to the limitations in the size of the array, power output of the transmitter, and lack of time to more fully develop the calibration procedures, we were able to show that it is possible to use the USRP2 as part of an operational phased receive array capable of determining a target's range, velocity, and direction.

4.1 Time Synchronization

This project showed that an array of USRP2s can be synchronized assuming the external reference clock source can meet the input power specifications of each radio. As the number of receivers in the array increases, amplification becomes necessary because the distributed power output available to each radio decreases. Provided the specific configuration described in this paper, we were able to achieve synchronization of less than 2 ns across the entire array and a spread about the mean sample time of each radio of less than 0.16 ns.

In comparison, the receivers currently used by Group 33 can achieve a synchronization of less than 2 ns; the clock drift of the radios is dependent on the accuracy and drift of the GPSDO.

When researching options for the signal splitters and amplifiers, we identified products capable of distributing the reference clock and PPS signals to up to 22 devices. Using these commercially available off-the-shelf (COTS) items could make it possible to expand the current system and add several more receivers to the array. However, as the size of the radar system increases, it is necessary to filter the clock signals to remove excess noise which can lead to clock jitter.

Future work on this topic may include research on the implementation of the internal phase locked loop in the USRP2. It is possible that clock synchronization performance could be improved by modifying the FPGA or UHD firmware. Another possible method for improving synchronization could be to bypass the phased locked loop altogether by distributing a common 100 MHz clock signal to each receiver in the system. The receive array used by Group 33 is synchronized in a similar manner where a single clock controls every radio.

4.2 Radar Processing

All of the radar processing techniques implemented in this project were preexisting and well defined. Developing improved processing algorithms was beyond the scope of this project. The purpose of implementing these techniques was to demonstrate an operational radar with range, Doppler, and direction processing capabilities using the USRP2. The results from the tests conducted with simulated and actual data show that the USRP2 is a viable software defined radio option for radar applications. The accuracy and resolution of our system was highly dependent on the hardware setup of the overall radar system. Limited to six antennas we had an effective direction finding resolution of 20 degrees. Factors such as array size, limited transmit power, and receive signal amplification negatively affected our results. However, this proof of concept has demonstrated that the USRP2 can be used to construct a

functional receive array.

Fabricating a larger line array would improve the angular resolution of the system and hence the direction finding accuracy. Adding a second dimension would allow the system to determine both the azimuth angle and elevation angle of a target. Relocating the transmitter further from the receive array would make it possible to use more powerful HF amplifiers. Increasing the transmit power would improve the radar's range and signal to noise ratio (SNR). Lastly, adding a front end amplifier to the USRP2 could also improve the array's performance.

4.3 Remarks

The experiments conducted during this project characterized the time synchronization specifications across multiple USRP2s. The USRP2 was then used to build a phased receive array and collect data used to determine the range, velocity, and direction of a target. This project shows that an inexpensive phased receive array can be developed using the USRP2.

5 References

Adamy, David. *Introduction to Electronic Warfare Modeling and Simulation*. Boston, MA: Artech House, 2003. 12-21. Print.

"Building GNU Radio on Ubuntu Linux." *Gnuradio.org*. GNU Radio, 29 Aug. 2011. Web. 9 Sept. 2011. <http://gnuradio.org/redmine/projects/gnuradio/wiki/UbuntuInstall>.

Ettus. "UHD Start - Ettus Research LLC." *Ettus Apps*. Ettus, 6 Sept. 2011. Web. 11 Sept. 2011. <http://ettus-apps.sourcerepo.com/redmine/ettus/projects/uhd/wiki>.

Hall, P. S., and R. G. Lee. "Principles of Radar Operation." *Land Warfare: Brassey's New Battlefield Weapons Systems and Technology Series* 9.1 (1991): 15-21. Print.

Jenkins, Herndon H. *Small-aperture Radio Direction-finding*. Boston: Artech House, 1991. Print.

Skolnik, Merrill Ivan. *Introduction to Radar Systems*. 3rd ed. New York: McGraw-Hill, 2001. 339+. Print.

Toomay, J. C., and Paul J. Hannen. *Radar Principles for the Non-specialist*. Raleigh, NC: SciTech Pub., 2004. Print.

Ziemer, Rodger E. ., and William H. . Tranter. "Multiplexing." *Principles of Communication*. 5th ed. Hoboken (N.J.): John Wiley & Sons, 2002. 176-83. Print.

6 Appendix

6.1 Doppler Frequency Shift

Doppler frequency shift can be used to determine the velocity of a moving target by the following derivation:

The total phase change, ϕ, of a two-way propagation path of the transmitted signal is defined by

$$\phi = 2\pi * \frac{2R}{\lambda} = \frac{4\pi R}{\lambda}$$

where λ is the carrier wavelength and R is the range of the target. Finding the change in phase over time yields the angular frequency:

$$\omega_d = \frac{d\phi}{dx} = \frac{4\pi}{\lambda}\frac{dR}{dx} = \frac{4\pi v_r}{\lambda} = 2\pi f_d$$

where v_r is the radial velocity and f_d is the Doppler frequency shift. Frequency f_d can be expressed as

$$f_d = \frac{2v_r}{\lambda} = \frac{2f_c v_r}{c}$$

where f_c is the frequency of the radar transmission and c is the speed of light.

6.2 GNU Radio Source Code Modifications

Table 9: Modifications made to `/gnuradio/uhd/host/lib/usrp/usrp2/clock_ctrl.cpp` In order to enable the clock debug pins

Line Number	Modification	Purpose
30	`static const bool enb_test_clk = true; // change 'false' to 'true'`	Enables the test clock
69	`this->enable_external_ref(true); // change 'false' to 'true'`	Enables the reference pin
236	`ad9510_regs_t::POWER_DOWN_LVPECL_OUT0_NORMAL; // change OUT0_SAFE_PD to 'OUT0_NORMAL'`	Turns debug pin output 'on' as default setting
254	`ad9510_regs_t::CHARGE_PUMP_MODE_NORMAL ; // change CHARGE_PUMP_MODE_3STATE to 'CHARGE_PUMP_MODE_NORMAL'`	Turns debug pin output 'on' as default setting

Once the source code has been adjusted for time synchronization and debugging, the UHD module was built and installed followed by the rest of GNU Radio. Following the directions from the GNU Radio Wiki, this was achieved with the following commands:

```
sudo make install
sudo ldconfig
```

The line ‘`ldconfig`’ is important for updating the links and cache to the updated libraries. Without it, GNU Radio might not be able to locate the necessary files for the UHD plugin.