RETRODIRECTIVE PHASED ARRAY ANTENNA FOR NANOSATELLITES

By

Justin W. Long, B.S.

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electrical Engineering

University of Alaska Fairbanks

December 2019

APPROVED:

Dr. Denise Thorsen, Committee Chair
Dr. Obadiah Kegege, Committee Member
Dr. Joseph Hawkins, Committee Member
Dr. Charles Mayer, Committee Member
Dr. Richard Wies, Chair
Department of Electrical Engineering
Dr. William Schnabel, Dean
College of Engineering and Mines
Dr. Michael Castellini, Dean
Graduate School

Abstract

This thesis presents a S-band phased array antenna for CubeSat applications. Existing state-of-the-art high gain antenna systems are not well suited to the majority of CubeSats, those that fall within the 1U (10 cm x 10 cm x 10 cm) to 3U (10 cm x 10 cm x 30 cm) size ranges and in Low Earth Orbit (LEO). The system presented in this thesis is designed specifically to meet the needs of those satellites. This system is designed to fit on the 1U face (10 cm x 10 cm) of a CubeSat and requires no deployables. The use of beamforming and retrodirective algorithms reduces the pointing requirements of the antenna, easing the strict requirements that high gain antennas typically force on a CubeSat mission. Additionally, this design minimizes volume and uses low cost Commercial-off-the-Shelf (COTS) parts.

This thesis discusses the theoretical background of phased array theory and retrodirective algorithms. Analysis are presented that show the characteristics and advantages of retrodirective phased antenna systems. Preliminary trade studies and design analyses show the feasibility and expected performance of a system utilizing existing COTS parts. The preliminary analysis shows that an antenna system can be achieved with ≥8.5 dBi of gain, 27 dB of transmitted signal gain, 20% Power Added Efficiency (PAE) within a 1 W to 2 W power output, and an 80° effective beamwidth.

Simulation results show an example antenna array that achieves 8.14 dBi of gain and an 82° effective beamwidth. Testing results on a prototype of the front-end electronics show that with minimal calibration, the beamforming and scanning error can be reduced to 5°. The power consumption and signal gain of the electronics is also verified through testing.

The CubeSat Communications Platform, a CubeSat mission funded through the Air Force Research Laboratory is in Phase A design to demonstrate this antenna system, along with other experimental payloads. This thesis includes a discussion of interface control, mission requirements, operations, and a recommended experiment sequence to test and verify the antenna system on-orbit.

Table of Contents

Page

List of Figures

Page

List of Tables

Page

List of Appendices

Symbols

Notation	Description
AF	Array factor
D	Directivity
$ERROR_{max}$	Maximum allowable antenna steering error
G	Gain
L	Patch Antenna Length
L	Loss
P	Power
S	Array spacing
T_{scan}	Period between scans
W	Patch Antenna Width
α	Steered phase difference between the n+1th and the nth element
β	Propogation Constant
ϵ_e	Effective dielectric constant
ϵ_r	Dielectric constant
η_0	Free space imedance, 376.730Ω
λ	Wavelength
ω_{sat}	Rotational velocity of spacecraft relative to ground station
ϕ	Zenith angle
ψ	Phase difference
θ	Azimuth angle
ζ	Signal phase difference between the n+1th and the nth element
c	Speed of light in vacuum
f_0	Design frequency
h	Dielectric thickness

Notation	Description
t_{SW}	RDA switching speed
t_{comm}	Spacecraft to RDA command time
t_{ground}	Ground system interrogator response time
t_{int}	Interrogator duration
t_{prop}	Propogation time from spacecraft to ground
t_{scan}	RDA scan duration
t_{setup}	Spacecraft setup and wait time before commanding scan
t	Copper thickness

Acronyms and Abbreviations

Notation	Description
ADCS	Attitude Determination and Control System
AF	Array Factor
AFRL	Air Force Research Laboratory
AR	Axial Ratio
ASGP	Alaska Space Grant Program
CCP	CubeSat Communications Platform
CDH	Command and Data Handling
COTS	Commercial-off-the-Shelf
DFN	Dual Flat No-Leads
DOA	Direction of Arrival
EIRP	Effective Isotropically Radiated Power
EPS	Electric Power System
HFSS	High Frequency Structure Simulator
HPA	High Power Amplifier
HPBW	Half-Power Beamwidth
IL	Insertion Loss
JAXA	Japan Aerospace Exploration Agency
LEO	Low Earth Orbit
MMIC	Monolithic Microwave Integrated Circuit
NASA	National Aeronautics and Space Administration
NEN	Near Earth Network
P-POD	Poly-Picosatellite Orbital Deployer
PAE	Power Added Efficiency
PCB	Printed Circuit Board

Notation	Description
QFN	Quadrature Flat No-Leads
RDA	Retrodirective Antenna
RF	Radio Frequency
RMSA	Rectangular Microstrip Antenna
RX	Receive
SAR	Synthetic Aperture Radar
SPDT	Single-Pole, Double-Throw
SSEP	Space Systems Engineering Program
TR	Transmit/Receive
TX	Transmit
UAF	University of Alaska Fairbanks
UHF	Ultra-High Frequency
UNP	University Nanosatellite Program

Glossary

amplitude imbalance

The off-nominal amplitude difference between two channels. That is, the additional amplitude difference that exists between two channels beyond the nominal expected amplitude difference.

antenna array

A configuration of multiple antenna elements that have been combined to modify the beam pattern

array factor

The beam pattern of an array of isotropic elements. The term "factor" indicates that in real arrays, one must multiply the array factor by the element pattern to obtain the full array pattern.

array pattern

The effective beam pattern of an array antenna.

axial ratio

The ratio of the major to the minor axis of a polarization ellipse.

azimuth angle

The angle from the X-axis towards the Y-axis.

command and data handling

A subsystem of a spacecraft. The Command and Data Handling unit is the central brain of the spacecraft, processing command and telemetry and making core spacecraft decisions.

commercial-off-the-shelf

Products that are commercially available, and are not custom designed or fabricated to suit a missions needs.

CubeSat

A type of nanosatellite that conforms to a set of standards developed by the California Polytechnic State University. See the CubeSat design specification [The CubeSat Program, 2014].

direction of arrival

The direction, epxressed as a zenith and azimuth angle, from which the interrogating signal approaches.

effective isotropically radiated power

the total power in watts that would have to be radiated by an isotropic antenna to give the same radiation intensity as the actual source at a distant receiver located in the direction of the antenna's strongest beam (main lobe).

half power beamwidth

The angle between the half-power (-3 dB) points of the main lobe of an antenna, when referenced to the peak effective radiated power of the main lobe.

insertion loss

The drop in signal power resulting from a device being inserted into a transmission path. Insertion Loss is typically measured in decibels, and is the decibel value of the S_{21} parameter.

isotropic

Radiating equally in all directions.

Ka-band

The band of the electromagnetic spectrum from 27 GHz to 40 GHz. This band is commonly used for satellite downlink, especially on deep space missions.

nanosatellite

An artificial satellite with a mass between 1kg and 10kg. All 1U to 6U CubeSats fall within this classification of satellite.

phase imbalance

The off-nominal phase difference between two channels. That is, the additional phase difference that exists between two channels beyond the nominal expected phase difference.

phased array

An array antenna antenna with the capability to separately control the excitation signal at antenna elements. Such an array has the capabilities of beamforming and scanning.

power added efficiency

A measure of the effiency of power added in an amplifier system. Defined as the (output power - input power) / (DC power).

retrodirective

The capability of an antenna to transmit a signal back towards an incoming signal. This could be as simple a corner antenna, in which the signal bounces off both walls of a corner, or as complex as a scanning phased array.

retrodirective antenna

An antenna that is capable of returning s signal in the direction of an interrogating signal.

S-band

The band of the electromagnetic spectrum from 2 GHz to 4 GHz. This band includes several allocations for satellite downlink and uplink. NASA has a downlink allocation at 2200 to

2290 MHz, and an uplink allocation from 2025-2110 MHz. Amatuer CubeSats commonly use a designated 2.4 GHz and for downlink.

ultra high frequency

The band of the electromagnetic spectrum from 0.3 GHz to 1 GHz. This band is commonly used for satellite uplink and health status downlink. It generally provides much lower bandwidth and data rates than higher frequency bands, but is considered robust and reliable.

X-band

The band of the electromagnetic spectrum from 8 GHz to 12 GHz. This band is commonly used for satellite downlink, and deep space missions often use it for uplink as well.

zenith angle

The angle from the Z-axis (antenna boresight) to the XY-plane.

Acknowledgements

I dedicate this thesis to my father, Richard Cyril Long

and in loving memory of my mother, Karen Woodruff Long

Special thanks to my advisor, Dr. Denise Thorsen. Despite my occasional obstinacy, she has mentored me with patience for the last six years. She challenged and inspired me to be a better engineer, and I am proud to credit her for whatever accomplishments I may achieve.

Special thanks to Dr. Obadiah Kegege, my research collaborator at NASA. Dr. Kegege's initiative and support repeatedly brought new opportunities and collaborations, including my new career at NASA.

Thanks to the UAF Space Systems Engineering Program for providing space, equipment, and expertise.

Thanks to the Alaska Space Grant Program for providing funding support in this project.

Thanks to NASA GSFC Flight Microwave and Telecommunications Systems branch for providing mentoring, equipment, and work space. Special thanks to Mae Huang, Cornelis Du Toit, and Victor Marrero-Fontanez for their support and guidance.

This work was supported in part by a NASA Space Technology Research Fellowship (80NSSC170185).

Chapter 1

Introduction and Background

This thesis discusses the design, prototyping, and testing of a RDA Array for nanosatellite applications. This first chapter will introduce the the theoretical background of the technologies, discuss the existing state-of-the-art, identify gaps in that state-of-the-art, and introduce the solution that is explored in this thesis.

1.1 Background of Nanosatellite Communications

The Nanosatellite Database maintained by Kulu [2019] shows that as of July 10, 2019 there were a total of 1186 nanosatellites launched, with plans to launch 438 in 2019 alone. Of the 1186 launched, 1088 are CubeSats. The rising popularity of CubeSats is largely due to the design standard maintained by California Polytechnic State University [The CubeSat Program, 2014]. This standard allows for CubeSats to be packed together and launched from a Poly-Picosatellite Orbital Deployer (P-POD). The P-POD is placed in otherwise unused space on rockets carrying larger and more expensive spacecraft, allowing for CubeSats to acquire reduced-cost launches. The CubeSat standard also allows for the development of standardized parts and Commercial-off-the-Shelf (COTS) products for CubeSats . The end result is a reduction in cost and development time, and increased access to space. Figure 1.1 shows past and planned nanosatellite launches by year, illustrating the rising popularity of CubeSats.

1.1.1 CubeSat Communications

As CubeSats rise in popularity, they have begun to explore new applications that demand high performance and efficient miniaturized satellite technologies. Specifically, there are three trends that motivate this project:

1. CubeSats are exploring applications that require high data throughput, such as earth observation remote sensing.

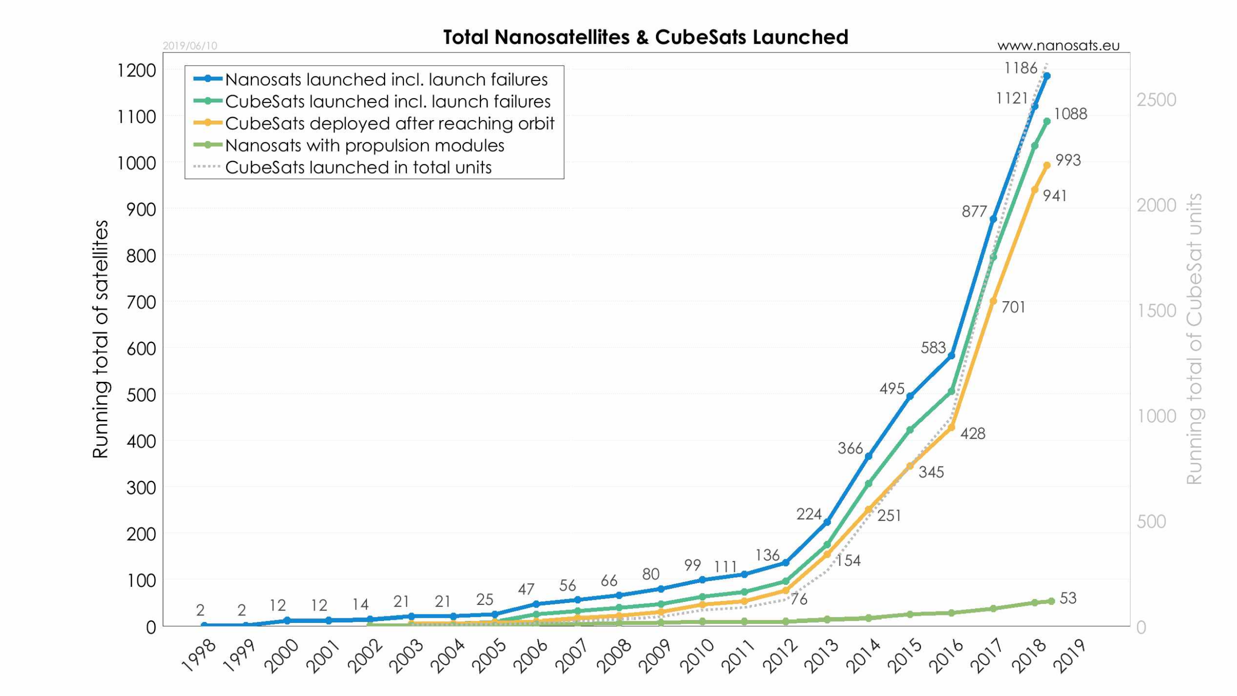


Figure 1.1: Nanosatellite launches by year [Kulu, 2019].

2. CubeSats are exploring applications that require versatile communications abilities, such as constellations and relays. Figure 1.2 shows several major CubeSat constellations currently in orbit or planned for launch.

3. CubeSats are starting to run into trouble regarding spectrum management, especially in the crowded Ultra-High Frequency (UHF) and X-band. Figure 1.3 shows CubeSats by freuqency band.

An example for demand of high performance communication systems is Planet Labs, who have launched over 350 3U earth observing CubeSats. Devaraj et al. [2019] discusses how they meet the communication challenges of such a large constellation, by using six data channel over a 300 MHz bandwidth at X-band and a deployable and narrow beamwidth helical X-band antenna. Another example, Japan Aerospace Exploration Agency (JAXA) plans to launch a constellation of 20 Synthetic Aperture Radar (SAR) satellites [Saito et al., 2019], and GHGsat Inc. shows plans to launch a constellation of 10 greenhouse gas observing satellites [Ligori et al., 2019].

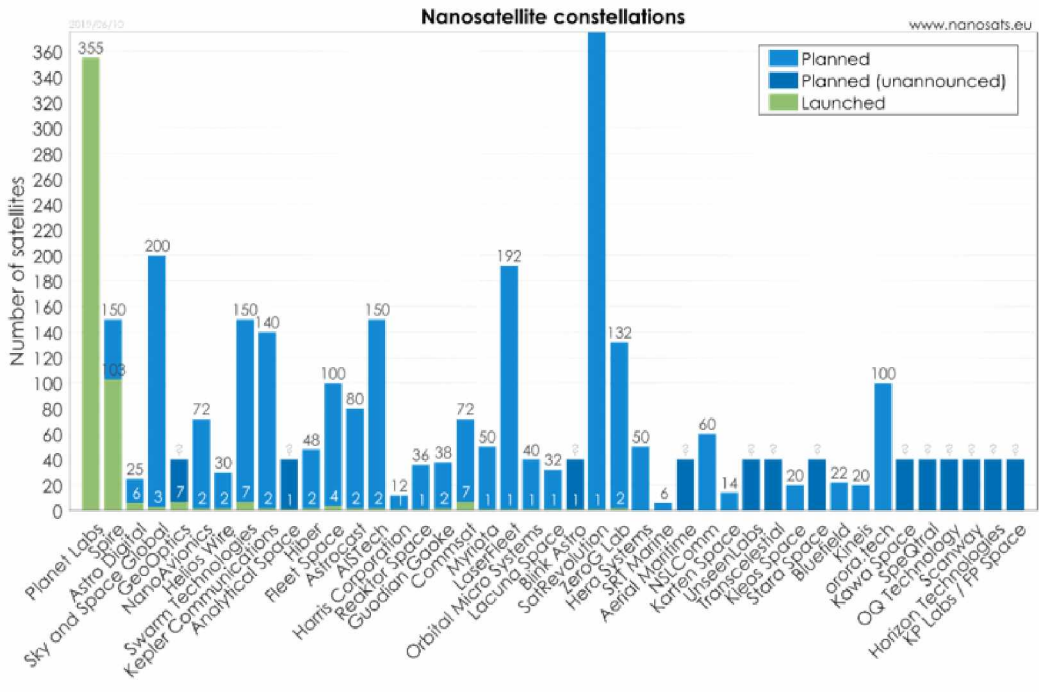


Figure 1.2: Nanosatellite constellations [Kulu, 2019].

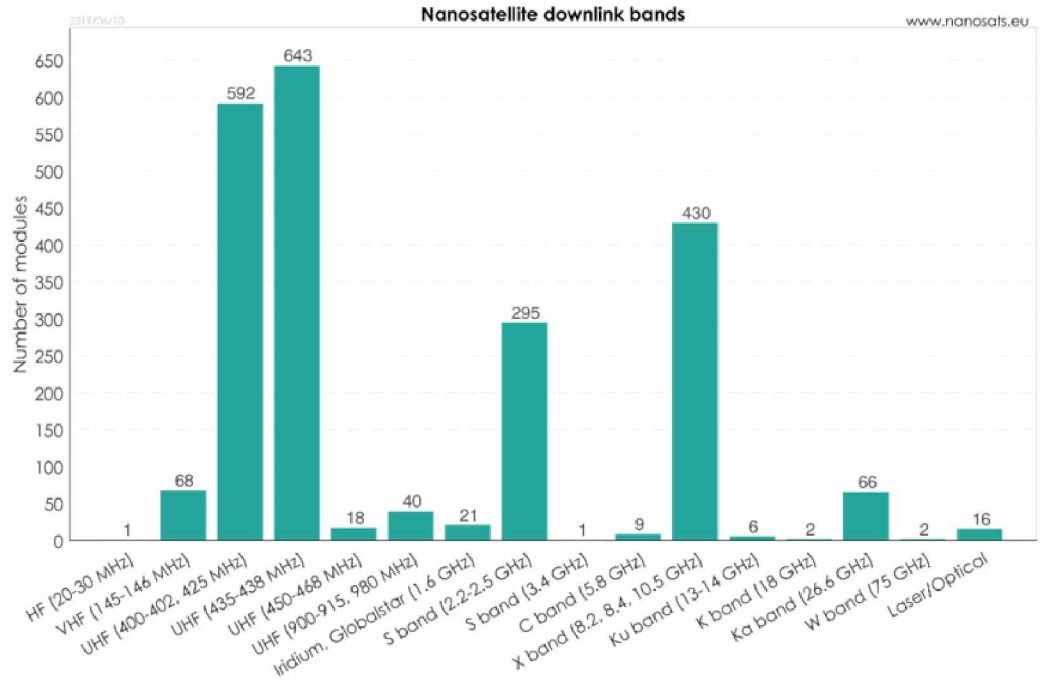


Figure 1.3: Nanosatellites by frequency band [Kulu, 2019].

1.1.2 Communication Link Budget

The most direct way of improving information throughput is to increase the received power. The received power of a basic communication link budget is described in decibel form by Equation (1.1).

$$P_{RX} = P_{TX} + G_{TX} - L_{TX} - L_{FS} + G_{RX} - L_{RX} - L_M \tag{1.1}$$

The power received P_{RX} is a function of the satellite transmit power P_{TX}, gain of the transmit antenna G_{TX}, total losses in the transmitter L_{TX}, free space losses L_{FS}, gain of the receive antenna G_{RX}, total losses of the receiver L_{RX} and other miscellaneous losses L_M. The power received at the ground can be increased by increasing the CubeSats transmit power, increasing the gain of the transmit antenna, decreasing the losses in the transmitter, minimizing the free space losses, and/or minimizing the other miscellaneous losses.

This thesis focuses on improving the transmit antenna gain G_{TX} by proposing a new antenna system for CubeSats. The following sections discuss recent work in the field of CubeSat antennas, and introduce the design solution presented within this thesis.

1.1.3 CubeSat State-of-the-Art

Historically, CubeSats have relied on low-data rate UHF communication systems. These systems are well suited to the CubeSat platform thanks to their simplicity, high reliability, and versatility. The link budget is easily closed with an omnidirectional dipole antenna and less than 1 Watt of transmit power.

Missions that require more data throughput raise the downlink frequency to S-band or X-band and utilize high gain directional antennas. The issue is that these directional antennas enforce pointing requirements that are difficult or costly to meet. They also often require deployables, which significantly increase the cost and complexity of CubeSat missions.

This thesis proposes a new CubeSat antenna to improve transmitter gain, while avoiding the strict constraints often placed by other high gain antennas, such as the high accuracy pointing or

deployables. The design being proposed is a phased array with retrodirective steering capabilities, known as a Retrodirective Antenna (RDA). Klein et al. [2014] showed that a four element phased array fit to the 1U face of a CubeSat can provide 13 dBi of gain at 2.4 GHz. Additionally, the design is steerable: the antenna can electronically steer its beam by controlling the phase of each element. This avoids the strict pointing requirements imposed on the satellite in other high gain antenna designs. The end result is an antenna that is feasible for CubeSats as small as 1U in orbits as low as Low Earth Orbit (LEO).

1.2 Antenna Array Theory

An antenna beam pattern can be modified by combining signals from multiple antenna elements. This configuration is known as an antenna array. The simplest analysis of antenna arrays is the case of two isotropic elements, or antennas that radiate and receive equally in all directions. Array analysis is performed by summing the phasors representing the amplitude and phase of each element.

Consider the case of two isotropic receivers, A and B, separated by a spacing S (in wavelengths). Because of the spacing between the receivers, they may receive an incoming signal at different times and therefore different phases. The phase difference between the elements depends on the Direction of Arrival (DOA) of the incoming signal, and the spacing between the receivers. Refer to the three cases shown in Figure 1.4.

In case (a) the signal reaches both elements at the same time, and each element receives the signal at the same phase. In this case, the signal received by the two elements combine constructively and there is a factor of 2 increase in signal strength (or 3 dB improvement). In case (b) the signal reaches receiver B, then travels the full distance S to reach receiver A. In this case, the phase shift is equal to the spacing S. For example, if S is half the wavelength of the signal (0.5λ), then the phase difference is 0.5λ, or 180°. 180° phase difference results in the signal being completely out-of-phase; the received signals combine destructively and cancel each other out.

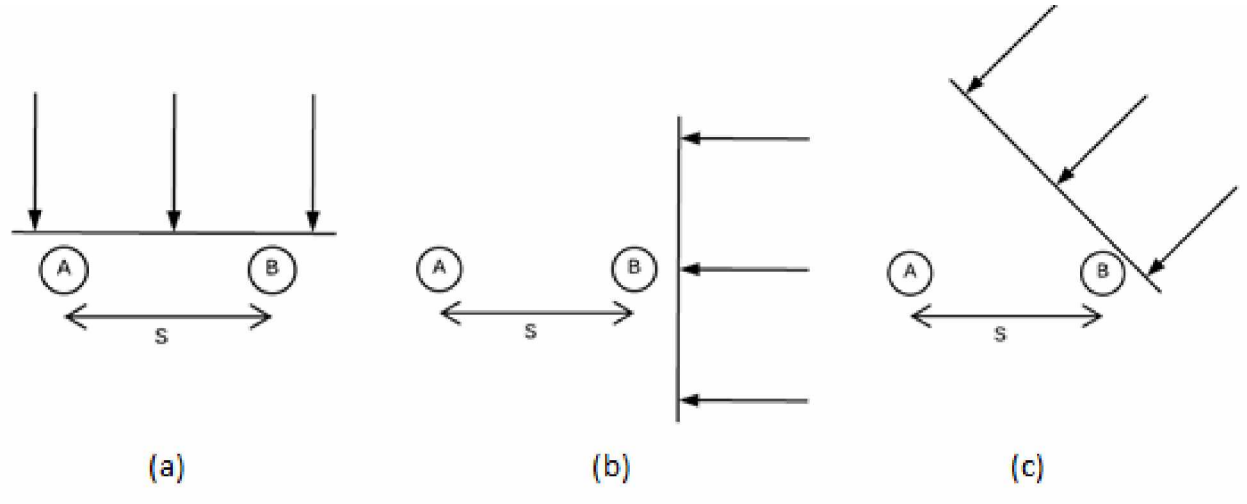


Figure 1.4: Isotropic elements receiving an incoming signal.

In case (c) the phase difference between the elements depends on both the DOA and the spacing. Figure 1.5 shows the unknown and desired parameters in case (a), and the solved parameters in case (b). The phase difference ψ between the elements is equal to the distance traveled by the signal wavefront to reach element A after it has already reached element B; this distance is marked as D in the figure. Through simple geometry it is can be shown that the angle opposite of side D is equal to the zenith angle ϕ and the phase difference between signals, ψ, can be solved. The solution is shown in Equation (1.2).

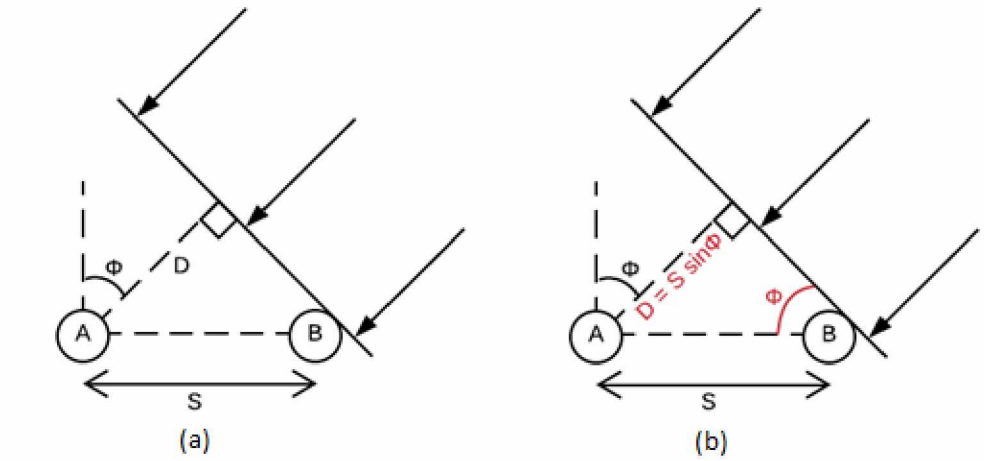


Figure 1.5: Isotropic elements receiving an incoming signal at an arbitrary angle.

$$\psi = S\,sin(\phi) \tag{1.2}$$

Knowing the phase difference between the elements, the signal at each element can be written as a phasor, and the two phasors can be summed together. This is shown in Equation (1.3).

$$AF = 1 + 1e^{-j2\pi\psi} = 1 + 1e^{-j2\pi S\,sin(\phi)} \tag{1.3}$$

1.2.1 Array Factor

The result shown in Equation (1.3) is known as the Array Factor (AF). The AF is the beam pattern of an array of isotropic elements. The term "factor" indicates that in real arrays, one must multiply the AF by the element pattern to obtain the full array pattern. The **array pattern** represents the effective beam pattern of the array antenna. Equation (1.4) shows the array pattern, $F(\phi, \theta)$ as the product of the AF and the element beam pattern, $f(\phi, \theta)$. For the two dimensional array, the azimuth angle, or θ, has been included. A description of the coordinate system can be found in Appendix A.

$$F(\phi, \theta) = AF(\phi, \theta)\, f(\phi, \theta) \tag{1.4}$$

1.2.2 Phased Arrays

Array antennas are popular because of the ability to shape the beam pattern by controlling the excitation signal at each element, and the ability to scan the beam through space by dynamically adjusting the phases electronically. An array with this capability is known as a phased array. Rather than needing to mechanically slew the antenna, a phased array is electronically steered inside the feed network. This is advantageous in several ways: the mechanical pointing requirements of the system are reduced, the electronic steering is much faster than any mechanical system, and some phased arrays are capable of multiple simultaneous main beams.

1.3 Array Geometries

This section explores array geometries, beginning with the linear array, and building up to a derivation of the array factor for the rectangular planar array. The rectangular planar array is the chosen array geometry for this thesis.

1.3.1 Linear Array

From the two element array, the next simplest array is the linear array, shown in Figure 1.6. The simple geometry of the linear array allows for an easy analysis and shows several important trends in phased arrays [Stutzman and Thiele, 2012].

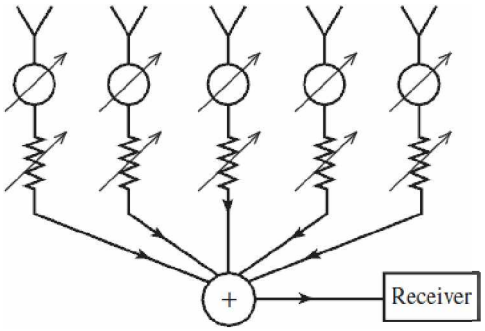


Figure 1.6: Linear phased array diagram [Stutzman and Thiele, 2012].

The AF for the linear array is derived using the same approach shown in Section 1.2, taking the sum of the signal phasors at each antenna element. The phase difference from the first element to the second element is ψ_1, and the phase difference from the first element to the third element is ψ_2, and so on, so that the AF for the linear array becomes Equation (1.5), where I is the excitation current of each element.

$$AF = I_0 + I_1 e^{j\psi_1} + I_2 e^{j\psi_2} + \cdots \tag{1.5}$$

If the linear array is uniformly spaced, then the geometry shown in Section 1.2 can be applied, and Equation (1.5) becomes Equation (1.6).

$$AF = I_0 + I_1 e^{jS\,sin(\phi)} + I_2 e^{j2S\,sin(\phi)} + \cdots = \sum_{n=0}^{N-1} I_n e^{jnS\,sin(\phi)} \tag{1.6}$$

Stutzman and Thiele [2012] perform further mathematical analysis of Equation (1.6), showing that the element spacing S determines how many periods of the array factor appear in the visible region, $-\pi/2 < \phi < \pi/2$. $S = 0.5\lambda$ spacing results in exactly one period of the AF. Larger spacing $S > 0.5\lambda$ results in more than one period, meaning that the sinusoidal array factor repeats itself within the visible region. Smaller spacing $S < 0.5\lambda$ results in fractions of a period.

A special case is the uniformly excited and uniformly spaced linear array, where the current amplitude of every element is a constant I_0. In this case, the AF simplifies to Equation (1.7). Analyzing this equation shows several important trends for the value of N:

1. As N increases, the main lobe narrows.
2. As N increases, the number of sidelobes increase, for a total of $N - 2$ side lobes.
3. The width of the main lobe is $4\pi/N$ and the width the sidelobes are $2\pi/N$.
4. The amplitude of the side lobes decreases as N increases.

$$AF = \frac{I_0\,sin\left(\frac{N\psi}{2}\right)}{sin\left(\frac{\psi}{2}\right)} \tag{1.7}$$

1.3.2 Rectangular Planar Array

The geometry of choice for this thesis is a planar array, which can be easily integrated into the CubeSat form factor. Stutzman and Thiele [2012] generalize Equation (1.6) for a two dimensional planar array, giving Equation (1.8) where (ϕ_0, θ_0) is the main beam pointing direction. The coordinates x'_{mn} and y'_{mn} correspond to the x and y coordinates of the mn^{th} element. Refer to Appendix A for a complete description of the coordinate system.

$$
\begin{aligned}
AF &= \sum_{m=1}^{M}\sum_{n=1}^{N} I_{mn} e^{j\alpha_{mn}} e^{j\zeta_{mn}} \\
\zeta_{mn} &= \beta\hat{r}\cdot\hat{r}'_{mn} = \beta[x'_{mn}sin(\phi)cos(\theta) + y'_{mn}sin(\phi)sin(\theta)] \\
\alpha_{mn} &= -\beta[x'_{mn}sin(\phi_0)cos(\theta_0) + y'_{mn}sin(\phi_0)sin(\theta_0)]
\end{aligned}
\tag{1.8}
$$

If all rows and columns in the array have identical current distrubution, then Equation (1.8) is seperable and can be further simplified to Equation (1.9).

$$
\begin{aligned}
AF &= \sum_{m=1}^{M} I_{xm} e^{j\alpha_{xm}} e^{j\zeta_{xm}} \sum_{n=1}^{N} I_{yn} e^{j\alpha_{yn}} e^{j\zeta_{yn}} \\
\zeta_{xm} &= \beta x'_m sin(\phi)cos(\theta) \\
\zeta_{yn} &= \beta y'_n sin(\phi)sin(\theta) \\
\alpha_{xm} &= -\beta x'_m sin(\phi_0)cos(\theta_0) \\
\alpha_{yn} &= -\beta y'_n sin(\phi_0)sin(\theta_0)
\end{aligned}
\tag{1.9}
$$

Equation 1.10 shows the maximum directivity for a planar array, where A_p is the physical area of the full array.

$$
D_{Max} = \frac{4\pi}{\lambda^2} A_p \tag{1.10}
$$

1.4 Phased Array Antennas

Arraying antennas has the potential to greatly improve information throughput by increasing the gain of the transmitting and/or receiving antennas, G_{TX} and G_{RX}. However, misalignment of phase due to AF can have the opposite effect. Phased arrays offer the first step in avoiding phase misalignment.

By introducing an intentional phase shift to certain elements of an array, a phased array can be created. Phased arrays can be used to scan for interrogating signals, perform beamforming, and facilitate multiple access technology. The basic concept is to phase shift the signal coming in at each element so that the signal combines in-phase to create the strongest signal possible. Alternatively, the signals can be shifted so that they are combined out-of-phase, negating unwanted signal.

1.4.1 Beamforming

Utilizing the array factor Equation (1.8), a phased array system can be designed to provide the unique advantages of beamforming. Beamforming is the process of controlling the phase shift of each element to "form" the antenna beam pattern as desired; a signal coming from any direction can be made to combine in-phase or out-of-phase. Desired signals can be constructively combined, and unwanted signals can be destructively combined. This process is also called steering.

To receive maximum power, the signal must combine in-phase at each antenna element. The necessary phase shift to achieve this for a particular ϕ in a two element array was already shown in Equation (1.2). That equation is rewritten in Equation (1.11), but denoting the phase shift as ψ_{lobe} to indicate that the phase shift is intentionally added for beamforming purposes.

$$\psi_{lobe} = S\,sin(\phi) \tag{1.11}$$

To cancel a signal, the signal must combine out-of-phase at each antenna element. To achieve this at a particular ϕ, a 180° (or π) phase shift is added to Equation 1.11. The resulting equation for the phase shift ψ_{null} is shown in Equation 1.12.

$$\psi_{null} = S\,sin(\phi) + \pi \tag{1.12}$$

An important note to make from Equation 1.4 is that the array pattern is a function of the pattern of the original element. This means that as a planar array is steered away from broadside, the gain decreases and the beam widens. Stutzman and Thiele [2012] states that as a phased array is steered, the peak of the total array pattern follows the element pattern shape. Stutzman and Thiele [2012] also provides an accompanying figure that shows an example of this trend, see Figure 1.7.

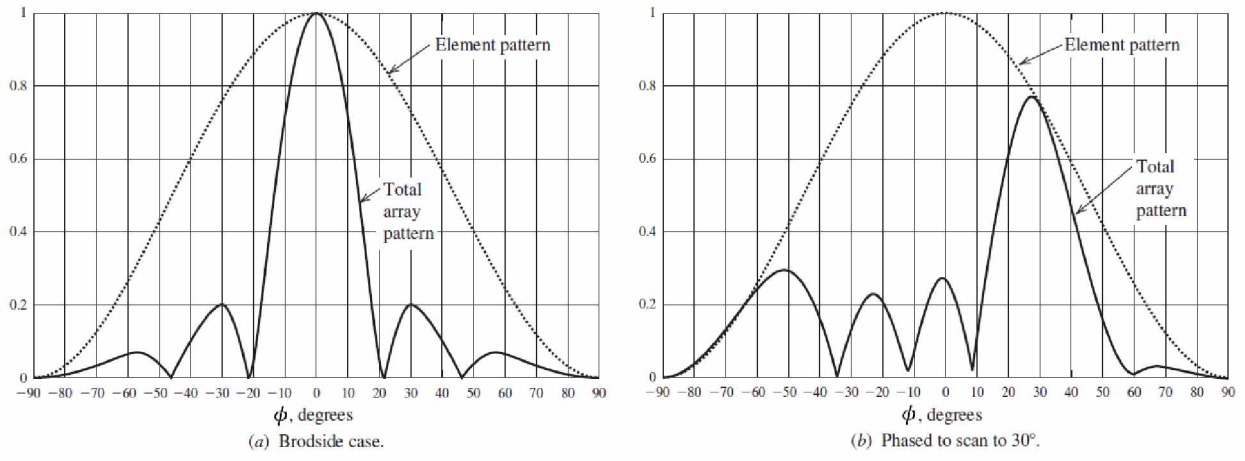


Figure 1.7: A linear phased array steered to (a) broadside, (b) 30° [Stutzman and Thiele, 2012].

1.4.2 Feed Network

The most significant complexities in a phased array arise from the feed network. Feed networks are typically parallel, series, or parallel-series fed. Space feeds are less common, using a feed horn to illuminate small pickup antenna elements before introducing the phase shift, and is not practical for CubeSat applications. Figure 1.8 shows the four common feed network architectures.

Stutzman and Thiele [2012] showed that the physical construction of the feed network is typically one of two designs, brick or tile. Brick construction places all of the feed network hardware for one element onto a single module. Tile construction layers the feed network so that like components are on the same layer.

1.5 Retrodirective Antenna Arrays

While electronically steered phased arrays are low profile and fast scanning, they require prior knowledge of the target's location. A Retrodirective Antenna (RDA) offers the capability of steering the antenna beam towards a target without any knowledge of the angular position of the target. The design proposed in this thesis implements retrodirective capabilities and explores various retrodirective algorithms. A brief introduction to retrodirectivity is given here, and plots highlighting the benefits of retrodirectivity are shown in Section 2.1.

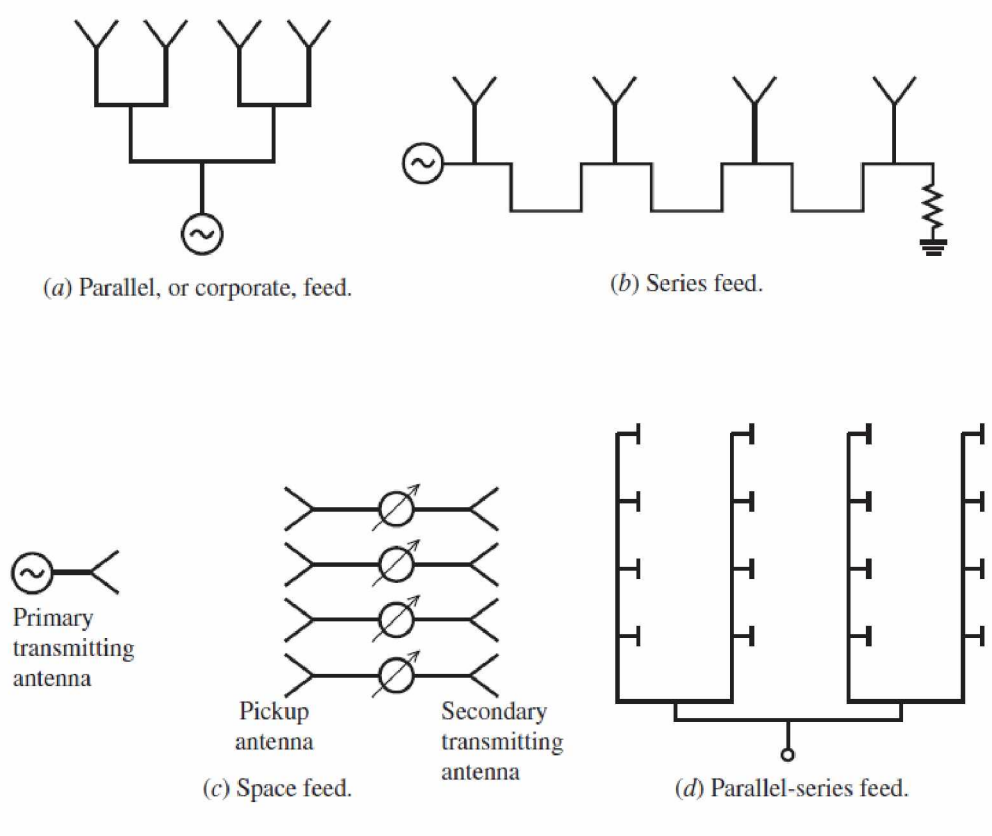


Figure 1.8: Types of array feed networks [Stutzman and Thiele, 2012].

1.5.1 Passive Reflectors

Two common RDAs using passive reflectors are the corner reflector and the Van Atta array. Kraus [1940] discusses the simplest example of a retrodirective antenna, the corner reflector, shown in Figure 1.9. Incoming signals bounce off one wall of the corner, then the other wall, and end traveling back towards the interrogating source. This is the definition of retrodirectivity, where a signal is transmitted back in the direction of an interrogating source.

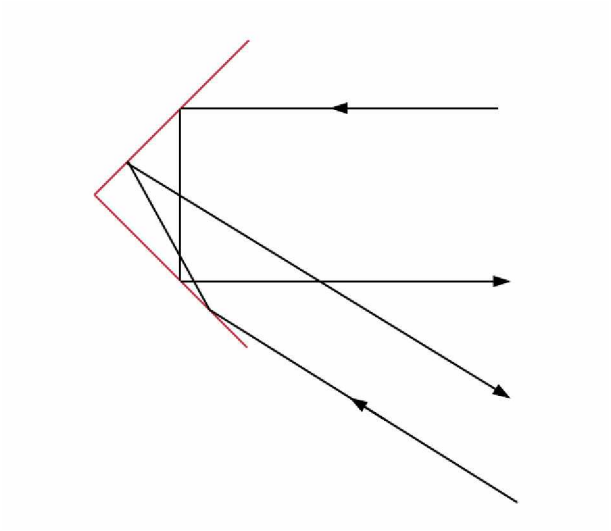


Figure 1.9: Retrodirectivity of a corner reflector.

Sharp [1958] introduced the Van Atta array, which operates similarly to the corner reflector in that it reflects an incoming signal back in the direction of arrival. But rather than physically bouncing the signal, the Van Atta array operates by interconnecting opposite antenna elements through equal length transmission lines. The interrogating signal is received at the first element, phase shifted 180° through the transmission line, and retransmitted out the opposite element. This occurs in every element pair, effectively retransmitting the signal back in the original direction. The operation of the Van Atta array is shown in Figure 1.10.

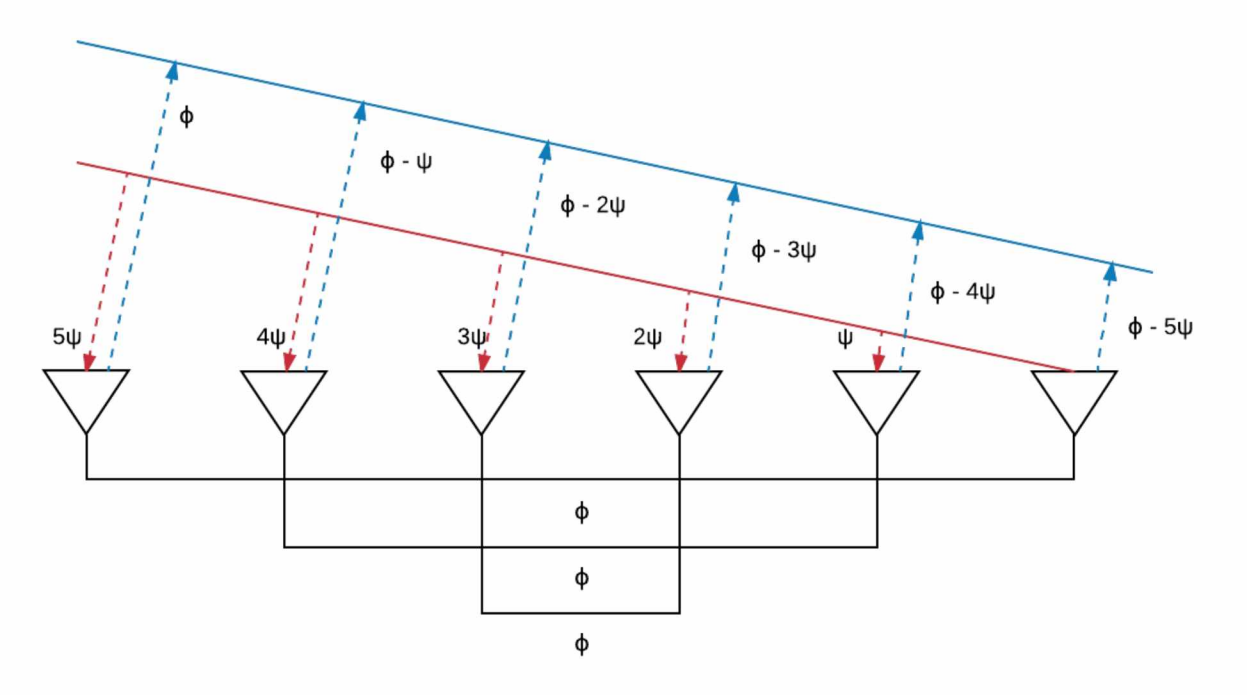


Figure 1.10: Operation of a Van Atta reflector array.

1.5.2 Scanning Arrays

The retrodirective arrays discussed so far are just passive reflectors. To create a more robust and useful duplex communication system, the array must be capable of determining the direction of an incoming signal. The process of finding the DOA of an incoming signal is known as scanning.

RDA arrays for digital communication systems became feasible with phase conjugation techniques. The architecture shown in Figure 1.11 was proposed by Shiroma et al. [2006], and shows a simple realization of a full-duplex RDA. The phase of the interrogating signal is measured in phase detectors in the detecting array, which gives the DOA. The beam of the transmitted signal is then directed back using phase shifters in the transmitting array. The drawback of this design is that the system is not capable of differentiating multiple interrogators.

Akagi et al. [2008] proposed a design for a power scanning RDA. The receive array uses phase shifters to scan the entire range of the array, and uses a power detecting circuit connected to a microcontroller to record the received power throughout the steering range. The microcontroller

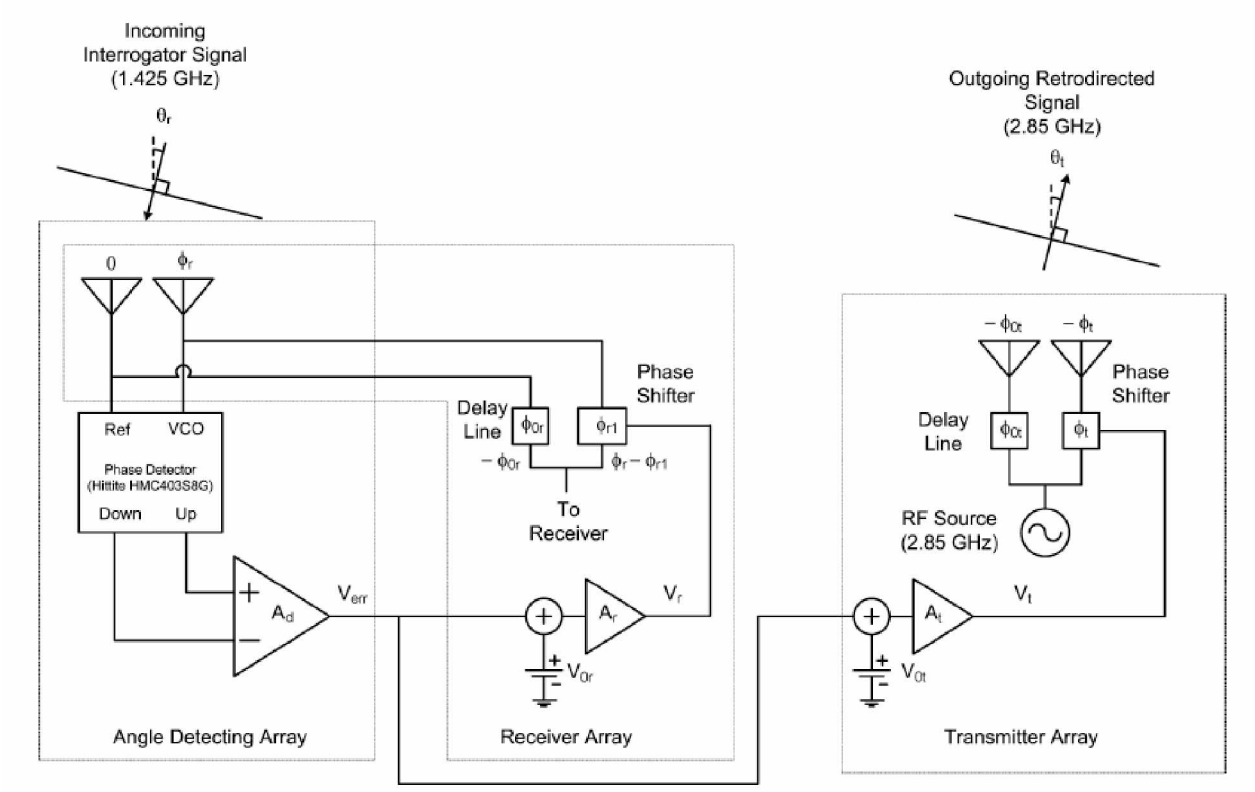


Figure 1.11: Architecture of a phase detecting RDA proposed by Shiroma et al. [2006].

then controls the phase shifters to steer the beam towards the highest power level, effectively aiming at the strongest interrogating signal. The drawback of this design is that the system must be switched between scanning and transmitting/receiving, so is not capable of constant receiving and transmitting. Compared to the phase detecting design, the advantage of the power scanning design is that it can be used to communicate with multiple interrogators, and power detecting circuitry is simpler and more common than phase detecting circuitry. Figure 1.12 shows the architecture proposed by Akagi et al. [2008].

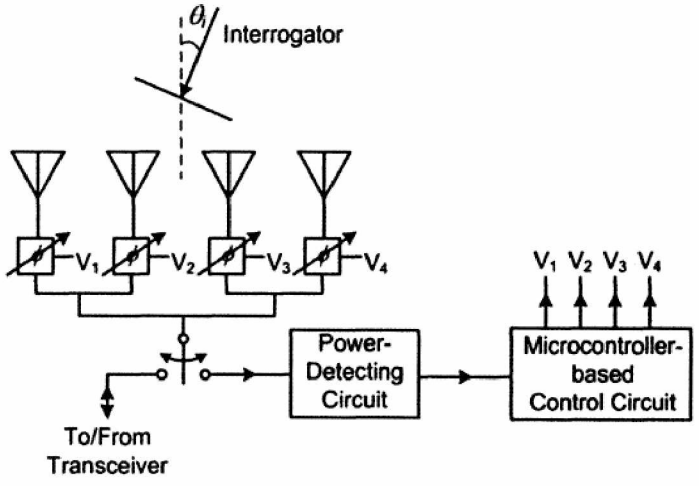


Figure 1.12: Architecture of a power scanning RDA proposed by Akagi et al. [2008].

Iwami et al. [2010] introduces the null scanning technique which improves location resolution over the power scanning technique. The null scanning design uses a similar concept to the power scanning design, but scans for power minimums rather than power maximums. By finding the power minimum, the array is searching for the location at which the interrogating signal is being received totally out of phase. As shown previously in Equation 1.12, adding a 180° phase shift at this location gives the direction of the interrogating signal. This is advantageous because the null has faster rate of change than the maximum power peak, resulting in more precise scanning. Additionally, the architecture shown by Iwami et al. [2010] uses separate scanning and phase shifting arrays so that constant transmission and reception is possible.

1.6 UAF CubeSat Initiatives

This thesis is part of an ongoing initiative at the University of Alaska Fairbanks (UAF) to develop an advanced CubeSat communication system. This program is managed through the UAF Space Systems Engineering Program (SSEP), with funding from the Alaska Space Grant Program (ASGP). This CubeSat communication initiative includes the work by Sielicki et al. [2013] to develop coding and modulation protocols. It also includes the work by Klein et al. [2014] to develop a phased array antenna to improve antenna pointing. Yet another thesis project involved in the program is the design of a software defined radio especially for CubeSats that are capable of implementing these technologies.

A realizable next step in the program is an investigation and design of a retrodirective antenna array. A retrodirective array can improve information throughput by further optimizing the loss and gain term to maximize the power received at the ground station.

During the development of this thesis project, SSEP received funding through the Air Force Research Laboratory (AFRL) University Nanosatellite Program (UNP) to develop a CubeSat that demonstrates the technologies developed as part of this advanced communication program. The RDA is one of the two primary experimental payloads for this CubeSat mission, known as the CubeSat Communications Platform (CCP). Because the RDA now has a preliminary mission to be designed for, the development of the RDA and CCP became intertwined efforts. For this reason, much of the work performed on the CCP mission relative to the RDA is discussed in Chapter 6.

1.7 Thesis Overview

Chapter 1: Introduction and Background introduces the theoretical background of the technologies being discussed in this project and explores the state-of-the-art for in CubeSat communication systems. **Chapter 2: Preliminary Analysis** shows the preliminary analysis that was performed to arrive at the final RDA design. This includes analytic simulations of the phased array technologies, trade studies, and design drivers, constraints, and requirements. **Chapter 3: Design and Simulation** covers the design of the RDA. This includes discussion of various antenna element

designs, the selected components, and the final architecture. **Chapter 4: Prototyping** discusses the prototyping method and fabrication. **Chapter 5: Testing** discusses all testing that was performed on the prototype. **Chapter 6: Mission Design and Recommendations** provides information on spacecraft mission design for the RDA. This includes operations, requirements, and interfaces. **Chapter 7: Future Work, Lessons Learned, and Conclusions** discusses future work that may be required to fully implement the RDA, or significantly improve it's performance. It also discusses lessons learned to provide insight for future developers.

Chapter 2

Preliminary Analysis

This chapter shows the preliminary analysis that was performed to arrive at the final RDA design. This includes analytic simulations of the phased array technologies, trade studies, and design drivers, constraints, and requirements.

2.1 Analytic Visualization

A series of analytical based plots are discussed here to highlight the intended benefits of phased array and retrodirectivity technology. This analysis will show that:

1. Arraying antennas improves gain, but decreases beamwidth.
2. Retrodirectivity compensates for the reduction in beamwidth.
3. The larger the array, the greater the advantage of retrodirectivity.

These plots use Equation (1.4) and Equation (1.9) to estimate the fixed and retrodirective array patterns. In the fixed array case, the steering angle (θ_0, ϕ_0) is fixed while the DOA angle (θ, ϕ) is swept. In the retrodirective array case, the steering angle (θ_0, ϕ_0) is equal to the DOA angle (θ, ϕ) throughout the sweep. Note that the plots do not show the beam pattern of the RDA at a single point in time, but rather the pattern when the array is beamforming to follow the interrogating signal. The RDA pattern at a single point in time is the same as the fixed array pattern when steered to the same angle. The code used to create these plots is shown in Appendix B.2. The results of this analysis was presented at the 2019 IEEE Aerospace conference and can be viewed in the conference proceedings. The paper publication is available in Appendix C.1 [Long et al., 2019].

2.1.1 Fixed Phased Array

For this simulation, a simple patch antenna was designed with the MATLAB antennas toolbox and placed into a 2x2 array with 0.5λ spacing. With a fixed steering angle of $(0°, 0°)$, the array pattern was calculated across the range of DOA angles using Equation (1.4) and Equation (1.9). The results

of the simulation are shown in Figure 2.1. The single element antenna gain is shown in blue, the AF is shown in green, and the 2x2 fixed array gain is shown in red. The X-axis is the DOA zenith angle at azimuth angle 0°. For these simulations, the zenith angle is plotted between 0° and 180° to show the front and back lobes.

The key results here are that for a fixed array, the gain increases but the beamwidth decreases. Ignoring all lossy effects, the gain of a 2x2 array is 6 dB higher than the individual element gain in the steering direction. As the DOA moves away from the steering direction, the array gain drops sharply. This is because the signals at each element of the array combine out-of-phase; this can be seen by the decrease in AF. When the AF passes below 0 dB, the array pattern becomes less than the single element pattern. This simulation does not account for mutual coupling or phase error. Measurements by Klein et al. [2014] showed an actual gain improvement of approximately 3 dB, which is 3 dB lower than the ideal value. This may have been due to phase imbalance or other unaccounted losses.

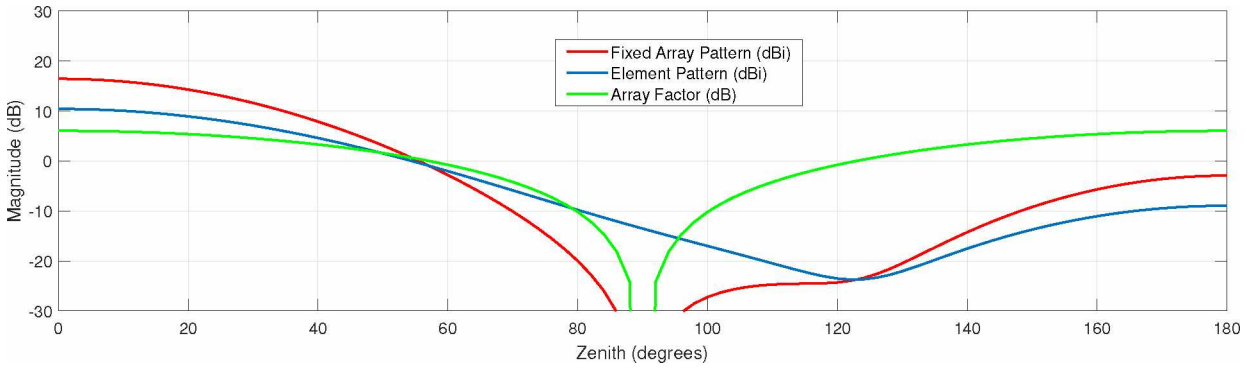


Figure 2.1: The base element pattern (blue), the array factor (green), and the resulting fixed array pattern (red) for a 2x2 array.

2.1.2 Retrodirective Phased Array

Figure 2.2 shows the benefits that a retrodirective array can offer. The same process was followed from the Fixed Phased Array simulation, but the steering angle was set to always be equal to

the DOA angle. Figure 2.2 shows the single element gain, the 2x2 fixed array gain, and the 2x2 retrodirective array gain.

The key results here are that with retrodirectivity, the benefits of increased gain are retained, while the issues of decreased beamwidth are solved. The retrodirectivity ensures the AF is always maximum. The 6 dB gain in the steering direction of the fixed array occurs across the entire field of view for the retrodirective array. The beam pattern of the single element antenna is preserved and increased by 6 dB in every direction.

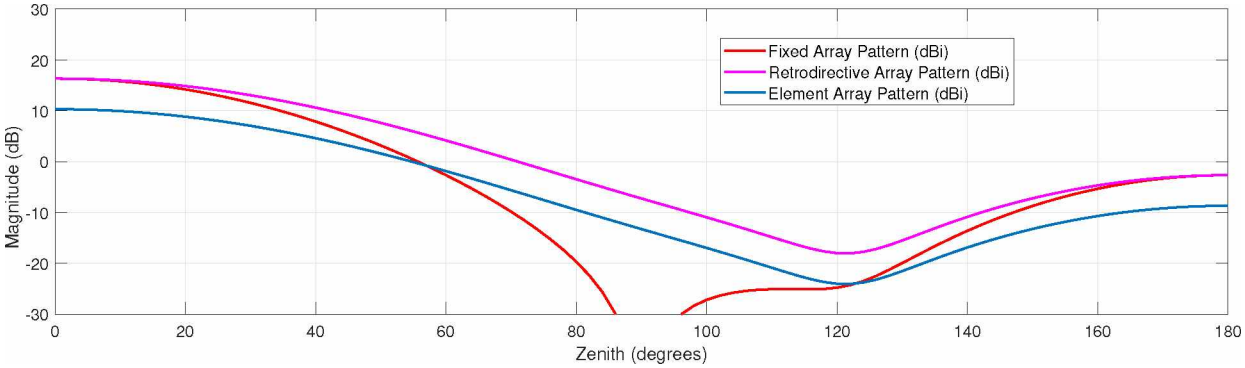


Figure 2.2: The base element pattern (blue), the fixed array pattern (red), and the retrodirective array pattern (magenta) for a 2x2 array.

2.1.3 Scaling to Larger Array

Figure 2.3 shows the effects of larger size arrays. The Retrodirective Phased Array simulation was repeated for a 4x4 array. Figure 2.3 shows the single element gain, the 4x4 fixed array gain, and the 4x4 retrodirective array gain.

The key results are that in larger arrays the gain increase and beamwidth decrease are more significant. Therefore the benefits of retrodirectivity are more significant in larger arrays. Ignoring all lossy effects, the 4x4 array has a gain improvement of 12 dB over the single element antenna. The fixed array pattern shows three nulls, compared to one in the 2x2 array. For the fixed array, the larger array requires more accurate pointing because of a narrower beamwidth and increased

number of nulls. The retrodirective array shows a constant 12 dB gain over the single element at all angles. This shows that larger array sizes give a greater benefit from retrodirective steering.

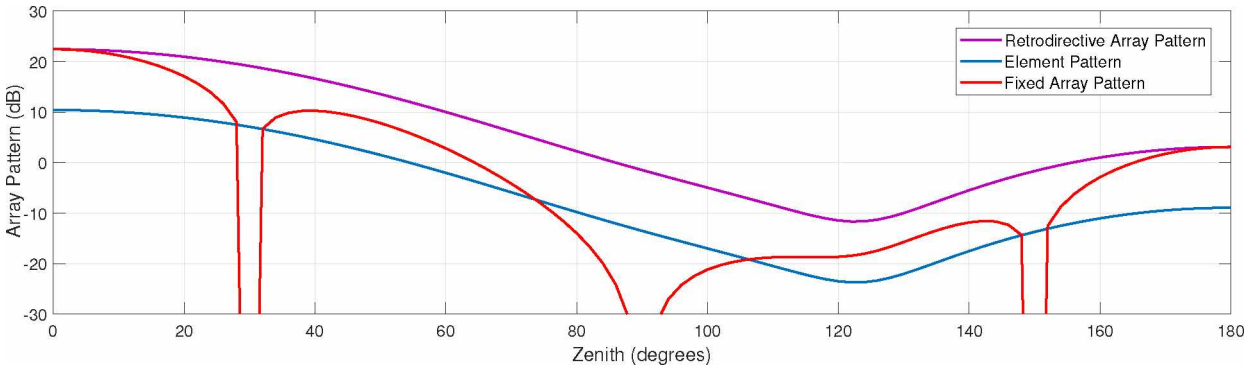


Figure 2.3: The base element pattern (blue), the fixed array pattern (red), and the retrodirective array pattern (magenta) for a 4x4 array.

2.2 Frequency

The selected frequency band for this project was 2.2 GHz. This is the S-band downlink frequency at which National Aeronautics and Space Administration (NASA) missions operate on the Near Earth Network (NEN). Specifically, satellite downlink occurs within 2200 MHz to 2290 MHz and satellite uplink occurs within 2025 MHz to 2110 MHz [Harris, 2016]. As this was a NASA funded project, a NASA frequency band was selected to maximize the benefit for NASA. Table 2.1 shows the list of NEN frequencies and bandwidths.

The proposed design will be optimized for the downlink capabilities of the antenna. Uplink capabilities are not typically a challenge for any earth-space satellite system, as the ground system has a large Effective Isotropically Radiated Power (EIRP).

As an alternative frequency, the 2.4 GHz frequency band is available for amateur radio use, and frequently used by CubeSats. The 2.4 GHz band has a large availability of Commercial-off-the-Shelf (COTS) products which could be used to create a low-cost and effective RDA system for CubeSats. Previous work, such as that done by Klein et al. [2014], used the 2.4 GHz frequency band.

Table 2.1: NASA Near Earth Network (NEN) frequencies.

Band	Frequency Band (MHz)	Bandwidth (MHz)	Maximum Bandwidth per Transmitter (MHz)
S Uplink	2,025-2,110	85	Typically <5
X Uplink	7,190-7,235	10	Typically <5
S Downlink	2,200-2,290	90	5
X Downlink	8,025-8,400	375	375
X Downlink	8,450-8,500	50	10
Ka Downlink	25,500-27,000	1,500	1,500

Higher frequency bands within X-band and Ka-band are also feasible and attractive for RDA systems. X-band is currently popular within military and large satellite applications. Additionally, there are X-band COTS products available that could be used to design a RDA system for CubeSats. Ka-band is becoming increasingly popular with NASA due to the large bandwidths available, but further technology development may be necessary before a miniaturized system can be developed for CubeSats. The use of an RDA in X-band or Ka-band requires ground stations that transmit within these bands, which is currently uncommon.

2.3 Antenna Element

There is a lack of available COTS components at the design frequency. Additionally, those wanting to replicate this project at other frequencies may encounter similar difficulties. For this reason, the design process for the antenna element was included in this project. A rectangular probe-fed patch antenna has been selected because it is simple to design, offers a moderate gain, and has a small form factor well suited to CubeSats. Based on existing antennas designed near the operational frequency, a single element directivity of 6 dBi is assumed for further preliminary analysis. Figure 2.4 shows a probe fed patch antenna for reference.

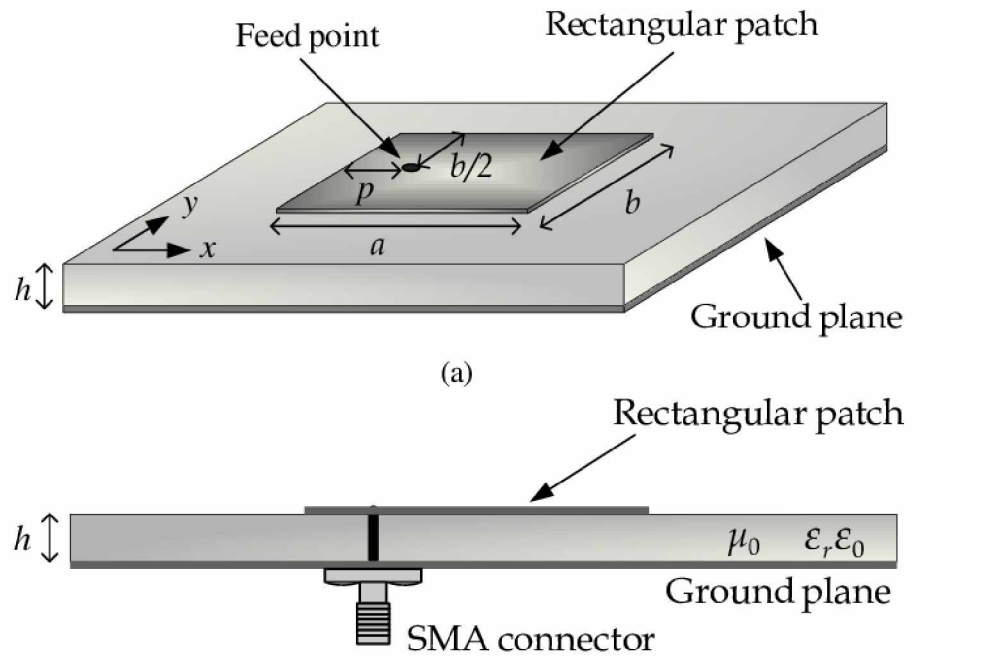


Figure 2.4: A probe-fed patch antenna [Nascimento and Lacava, 2014].

2.3.1 Dielectric Materials

The characteristics of the dielectric substrate determine the size and performance of the patch antenna. As the dielectric constant increases, the physical size of the patch shrinks. However, the matching of the antenna becomes more challenging, and the gain and bandwidth of the antenna tend to decrease. Table 2.2 shows the effects of the dielectric constant on performance. For this project, a dielectric constant of at least 8 is required to fit a 2x2 patch antenna array on a 1U CubeSat face. Rogers 6010LM was chosen as the dielectric, as it offers a dielectric constant of 10.7 with the a relatively low dissipation factor of 0.0023. Additionally, Rogers 6010LM is rated for in-space applications.

2.4 Front-End Electronics

The front-end electronics include everything between the radio interface and the antenna input. There are two key aspects for analysis here: the selection of the components and the design of the feed network (how to connect the components together).

Table 2.2: Effect of dielectric constant on the performance of a microstrip antenna (h=0.159 cm, f_0=3.0 GHz) [Kumar and Ray, 2003].

ϵ_r	L (cm)	W (cm)	BW (MHz)	Gain (dBi)
1	4.65	6.2	74	10
2.55	3.0	4.0	64	6.8
4.3	2.3	3.1	49	5.6
9.8	1.51	2.0	30	4.4

2.4.1 Feed Network Design

There are three major considerations in the feed network: where to place the phase shifter, how to implement Transmit/Receive (TR) switching, and where to place the power amplifier. Feed network design must include careful consideration of the impacts on operational characteristics (e.g. duplex operation, single channel or dual channel radio), output power (due to power handling and insertion loss), Power Added Efficiency (PAE), cost, complexity, and reliability. Figure 2.5 shows four example feed network designs, discussed below. Design 4 was the final selection chosen for the prototype, and is discussed further in Section 3.3.

Design 1 A TR switch at the antenna element separates the Transmit (TX) and Receive (RX) paths. Each channel has a separate phase shifter for transmit and receive. Each channel has a power amplifier in the TX path between the phase shifter and TR switch. This design can accommodate a high output power, but has the drawback of increased hardware and software complexity due to the additional phase shifters.

Design 2 The phase shifter is placed between the antenna element and the TR switch. A single power amplifier is placed at the radio output. This design reduces that hardware and software complexity, and increases Power Added Efficiency (PAE). The drawback is that the total power output of the system is limited by the power handling capabilities of the phase shifters.

Design 3 Separate arrays are used for the transmit and receive paths. This design provides full duplex operation and high output power, but requires sufficient surface area for the extra array.

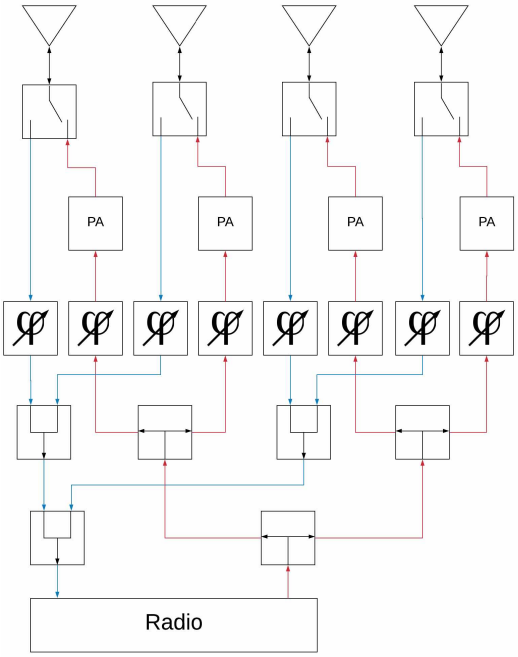
Design 4 Dual TR switches provide a single feed path to the radio. This is similar to Design 1, but only one phase shifter is required per channel. This design can support high output power and has less hardware and software complexity.

2.4.2 Component Selection

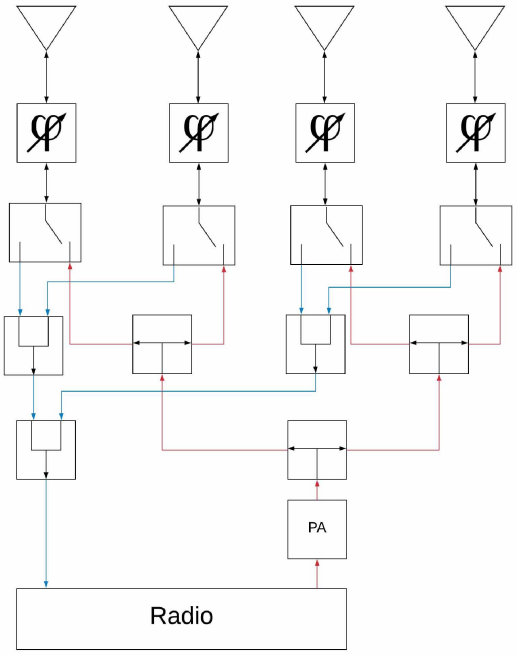
There are five major components to the phased array: phase shifter, high power amplifier, switch, splitter, and controller. The purpose of each of these components and available commercial options are discussed in the following sections. Performance and specifications for the final component selections is discussed in Section 3.3.

Phase Shifter

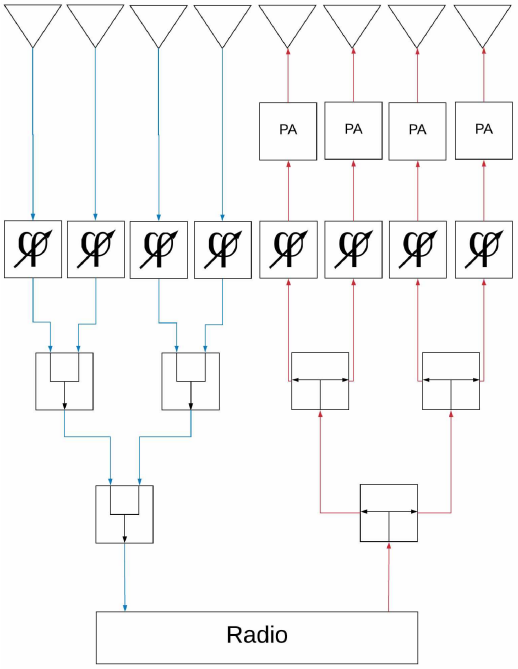
The phase shifter is the most critical component of the phased array. Each antenna element requires a phase shifter, which generates the phase shifts that allow beamforming in the phased array. The phase shifters should be selected to minimize random phase and amplitude errors, as this error critically affects the performance of the array. The phase shifters should also provide fine phase resolution control. Secondarily, they should reduce Insertion Loss (IL) and have a high maximum power rating to support high transmit power. Two COTS phase shifters were found that could operate in the desired frequency range; these phase shifters are compared in Table 2.3. The PE44820 was selected for the reduction in phase and amplitude error, which are the critical performance parameters.



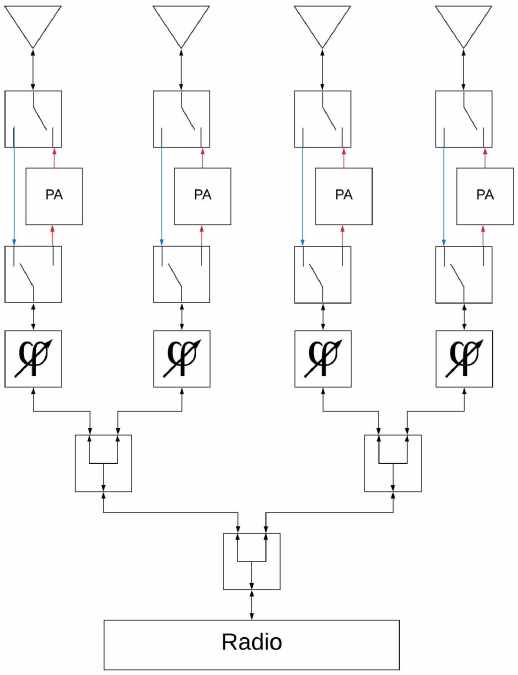
(a) Design 1



(b) Design 2



(c) Design 3



(d) Design 4

Figure 2.5: Four examples of RDA feed architecture.

Table 2.3: Phase shifter component comparison.

Part	Resolution	RMS Phase Error	Amplitude Error	Insertion Loss	Max Rec. Power
PE44820	1.4°	1°	0.1 dB	6 dB	25 dBm
MAPS-010163	5.6°	3°	0.4 dB	5 dB	25 dBm

High Power Amplifier (HPA)

The High Power Amplifier (HPA) provides the transmit power amplification. The HPA must provide adequate gain to provide the desired output power for a specified input power. Beyond this, the HPA should be selected to minimize random phase and amplitude error, maximize PAE, and maximize output power. Typically, amplification circuitry uses a multi-stage amplifier design in which the first stages maximize gain and the final stage maximizes efficiency. One component was found that includes a three stage amplifier with good gain and PAE performance, the Skyworks SKY66294. Alternatively, a custom multi-stage amplifier can be designed, but increases design complexity. To provide up to 2 W power out from a 1 mW input signal, the power amplifier must have greater than 33 dB of gain. To keep the DC power consumption below 10 W, the power amplifier must have a PAE of at least 20%. The Skyworks SKY66294 typically has 36 dB of gain and greater than 35% PAE.

Switch

The switch is used to change signal path between TX and RX paths. The switches must provide adequate isolation to protect other equipment, and have adequate power handling capabilities to protect itself. Beyond this, the switches should be selected to minimize random phase and amplitude error, and minimize IL. There are hundreds of COTS switches available for selection, so the following criteria were used as minimum performance metrics.

- $\geq$ 27 dB isolation
- $\geq$ 27 dBm 1 dB compression point
- $\leq$ 0.4 dB insertion loss

A comparison of the considered switches is shown in Table 2.4. The SKY13330 was selected because it provided the highest isolation and power handling capabilities.

Table 2.4: Switch component comparison.

Part	Insertion Loss	IP1dB	Isolation	Switching Speed
SKY13330	0.35 dB	39 dBm	33 dB	1.7 us
SKY13335	0.4 dB	29 dBm	27 dB	0.05 us
SKY13405	0.35 dB	38 dBm	27 dB	2.3 us
MASWSS0136	0.4 dB	28 dBm	27 dB	0.04 us

Splitter

The splitters take the signal from the radio and divide it between the channels of the array. The splitters should be selected to minimize phase imbalance and amplitude imbalance. Secondarily, the splitters should minimize IL. Only one COTS Monolithic Microwave Integrated Circuit (MMIC) splitter was available at the design frequency, the Skyworks 16406-381LF. The Skyworks 16406-381LF has an insertion loss of 0.3 dB, an isolation of 25 dB, a phase imbalance of less than 0.3° , and an amplitude imbalance of less than 0.05 dB. Other COTS splitters are not MMIC, which are too large and not well suited to the CubeSat form factor.

Controller

The controller provides the necessary control voltages and algorithms to perform all phased array and retrodirective operations. As all of these operations and the associated circuitry are simple,

nearly any microcontroller is adequate. Therefore, the microcontroller should be selected for power efficiency, computation time, and in-space durability. For this project, an in-house MSP430 development board was used. This microcontroller is compact, ultra low-power, verstile, and well suited to spacecraft applications.

2.5 Array Size

For the purpose of prototyping, this project only included the design of a 2x2 array designed for a 1U CubeSat. The design can be scaled to a much larger array. Table 2.5 shows the frequencies that have been considered for this technology, the number of elements that could fit on a 1U face, and the expected gain. Note that the directivity is based on the use of 5.5 dBi patch antenna, and assumes the components and architecture selected for the prototype.

Table 2.5: Example phased array sizes and characteristics for a 1U face.

Frequency	**Wavelength (cm)**	**# of Elements**	**Directivity (dBi)**
2.2 GHz	13.64	4	11.5
2.4 GHz	12.50	4	11.5
8 GHz	3.75	36	21.0
10 GHz	3.00	49	22.3
26 GHz	1.15	324	30.5

There are practical limits to the size of the array: (1) each element has a per-element overhead, (2) the Power Added Efficiency (PAE) of the High Power Amplifier (HPA) drops as the input power into each channel reduces, and (3) larger arrays require more complicated feed networks with increased IL. This means for a fixed power array, EIRP starts to diminish as the overhead power draw begins to overcome the benefits of increased gain.

Figure 2.6 shows an example of how the EIRP begins to diminish in larger arrays. This assumes a 5.5 dBi element gain with the front end electronics design and components discussed in Section 2.4.

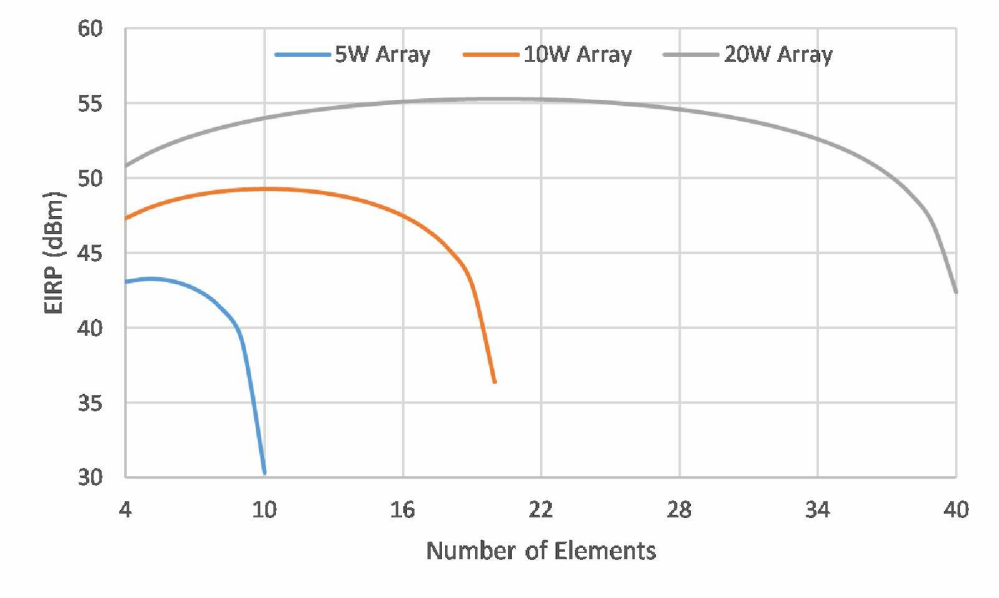


Figure 2.6: An example of diminishing returns for EIRP vs. array size.

2.6 Expected Performance

2.6.1 System Gain

An analysis was performed to show the expected gain of the prototype system, making the ideal assumptions that there are no board or connector losses. Figure 2.7 shows a block diagram that adds up the gains and losses of the selected components. The total gain of the system, without accounting for board losses, is approximately 27 dB. It is expected that the actual gain of the fabricated system will be up to 1 dB lower than this to account for board losses.

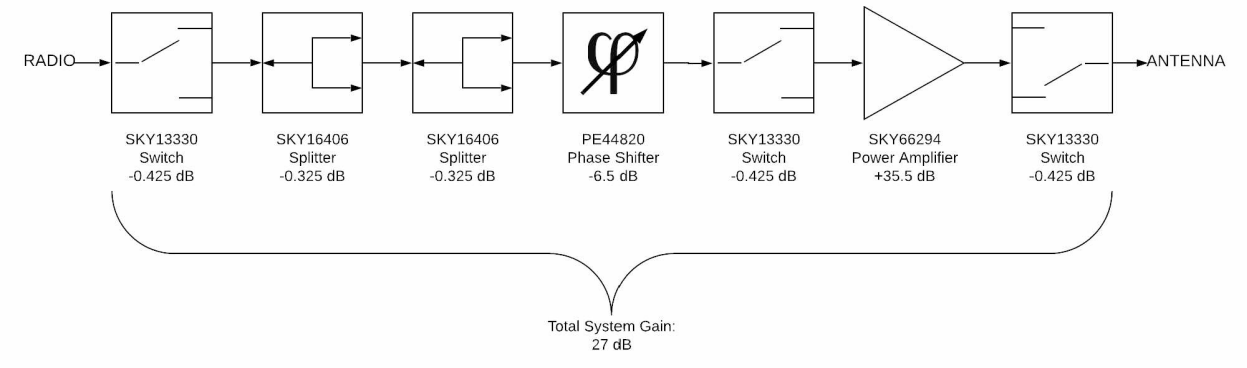


Figure 2.7: Block diagram of expected system gain.

2.6.2 Antenna Performance

Based on available COTS products, a single element provides a gain of 5.5 dB. The ideal array factor of a 2x2 array adds 6 dB to this gain, giving an ideal array gain of 11.5 dBi. However, this is an unrealistic expectation due to parasitic and other losses. Based on the previous work by Klein et al. [2014], an array factor of approximately 3 dB at broadside can be expected. This results in an expected array gain of 8.5 dBi at broadside. For reference, high performance patch antennas that occupy a similar area as the array provide 7 dBi of gain [EnduroSat, 2018].

2.6.3 Power Consumption

The front end electronics of the RDA draw significant power in both the TX and RX states. The selected power amplifier uses significant amounts of power even when disabled. To conserve power, the HPAs should be switched out of line and powered down using a switchable power rail. The HPAs operate more efficiently at low temperatures, so there is further reason to keep the device powered off until needed for TX. The power consumption per component and for each mode of operation is shown in Table 2.6.

2.7 Summary of Requirements

Driving requirements of the RDA have been discussed throughout this chapter, but are summarized here. These requirements were largely derived through conversations with CubeSat developers, inherent knowledge on the needs of CubeSat missions, and statements within various CubeSat mission reports.

Area The RDA shall have an area no larger than 10cm x 10cm. This corresponds to the 1U face of a CubeSat, allowing a 1U mission to implement the antenna without deployables.

Volume The total volume of the RDA shall be less than 0.25U of a CubeSat. This allows a 1U CubeSat mission to implement the antenna and still have volume available for avionics and payload.

Table 2.6: RDA power budget analysis.

Component	Power Each (mW)	Quantity	Total Power (mW)
Microcontroller [1]	2.68	1	2.68
Phase Shifter	0.50	4	1.98
Switch	0.13	8	1.00
HPA - Idle	375.0	4	1500.0
HPA - TX [2]	1250.0	4	5000.0
Mode			Total Power (W)
RX (HPA off)			0.0
RX (HPA idle)			1.5
TX			5.0

(note 1: Microcontroller power assumes a very conservative 20% active, 80% idle duty cycle)

(note 2: HPA TX power is for 1 W RDA output)

Frequency The RDA shall operate at 2.2 GHz, corresponding to the NEN downlink band. This mission is funded through NASA, and a NASA NEN manager requested that the device be designed at NEN frequency.

Gain The gain of the antenna at broadside shall exceed 8 dBi. This is equivalent to commercially available antennas of similar size.

Beamwidth The phased Half-Power Beamwidth (HPBW) of the RDA shall exceed 80°. This is an improvement upon the existing commercial products of similar size and gain.

Bandwidth The RDA shall have a bandwidth of at least 5 MHz, corresponding to the maximum allowable bandwidth for a single transmitter at the NEN frequency band.

Availability The RDA shall use COTS components, with the exception of Printed Circuit Board (PCB) layouts and antenna designs. COTS components allow the RDA to be cheaply fabricated and implemented, without requiring significant skills, facilities, or equipment. Custom PCBs and antenna elements will be required to perform the design.

Retrodirectivity The RDA shall have the capability to locate an interrogator signal, and transmit a signal back in the direction of the interrogator.

Chapter 3

Design and Simulation

This chapter discusses the RDA design, including a custom patch array design and the front-end electronics architecture and components.

3.1 Antenna Element Design

The design equations and process for an Rectangular Microstrip Antenna (RMSA) patch antenna are shown here. First, the dielectric permittivity must be corrected for the fringing field effects at the edge of the patch. This is done by finding the effective dielectric constant ϵ_{e1}, which modifies the dielectric constant ϵ_r with a correction for the fringing effects. Equation (3.1) shows the standard calculation of ϵ_{e1}, where u is the ratio of the patch width to substrate height (W/h) [Hammerstad and Jensen, 1980].

$$\begin{aligned} \epsilon_{e1}(u, \epsilon_r) &= \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2}\left(1 + \frac{10}{u}\right)^{-a(u)b(\epsilon_r)} \\ a &= 1 + \frac{1}{49}log\left(\frac{u^4 + \left(\frac{u}{52}\right)^2}{u^4 + 0.432}\right) + \frac{1}{18.7}log\left(1 + \left(\frac{u}{18.1}\right)^3\right) \\ b &= 0.564\left(\frac{\epsilon_r - 0.9}{\epsilon_r + 0.3}\right)^{0.053} \\ u &= \frac{W}{h} \end{aligned} \tag{3.1}$$

Next, Equation (3.1) is corrected for non-zero strip thickness (t). This is done with Equation (3.2), which provides ϵ_{e2} an additional adjustment to ϵ_{e1} that compensates for the strip thickness [Hammerstad and Jensen, 1980].

$$\epsilon_{e2}(u,t,\epsilon_r) = \epsilon_{e1}(u_r,\epsilon_r)\left(\frac{Z_{01}(u_1)}{Z_{01}(u_r)}\right)^2$$

$$Z_{01}(u) = \frac{\eta_0}{2\pi} log\left(\frac{f(u)}{u} + \sqrt{1+\left(\frac{2}{u}\right)^2}\right)$$

$$f(u) = 6 + (2\pi - 6)\, e^{-\left(\frac{30.666}{u}\right)^{0.7528}}$$

$$u_1 = u_0 + \Delta u_1 \tag{3.2}$$

$$\Delta u_1 = \frac{t}{\pi} log\left(1 + \frac{4e^1}{t\, coth\left(\sqrt{6.517u}\right)}\right)$$

$$u_r = u_0 + \Delta u_r$$

$$\Delta u_r = \frac{\Delta u_1}{2}\left(1 + \frac{1}{cosh\left(\sqrt{\epsilon_r - 1}\right)}\right)$$

Lastly, a dispersion correction is applied. This is done with the process shown in Equation (3.3), which provides the final effective dielectric constant, ϵ_e [Bhartia et al., 2001].

$$\epsilon_e = \epsilon_r - \frac{\epsilon_r - \epsilon_{e2}}{1+P}$$

$$P = P_1 P_2 \left((0.1844 + P_3 P_4)\, f_n\right)^{1.5763}$$

$$P1 = 0.27488 + u\left(0.6315 + \frac{0.525}{(1+0.0157 f_n)^2\, 0}\right) - 0.065683 e^{-8.7513u}$$

$$P_2 = 0.33622\left(1 - e^{-0.30442\epsilon_r}\right) \tag{3.3}$$

$$P_3 = 0.0363 e^{-4.6u_p}\left(1 - e^{-\left(\frac{f_n}{38.7}\right)^4 .97}\right)$$

$$P_4 = 1 + 2.751\left(1 - e^{-\left(\frac{\epsilon_r}{15.916}\right)^8}\right)$$

$$f_n = f_0 h * 10^{-6}$$

To begin a design, a resonance mode or starting geometry must be chosen. For TM_{10} mode, Equation (3.4) can be used for a starting width.

$$W = \frac{c}{2 * f_0 * \sqrt{\frac{\epsilon_r + 1}{2}}} \tag{3.4}$$

After a design geometry or resonance has been selected, the effective dielectric constant ϵ_e is calculated using Equation (3.1), then Equation (3.2), and finally Equation (3.3). After finding ϵ_e, Equation (3.5) is applied to find the resonant length by subtracting the length extension due to the fringing fields.

$$
\begin{aligned}
L &= L_e - 2\Delta L \\
L_e &= \frac{c}{2 * f_0 * \sqrt{\frac{\epsilon_r+1}{2}}} \\
\Delta L &= h\frac{\zeta_1\zeta_2\zeta_3\zeta_5}{\zeta_4} \\
\zeta_1 &= 0.434907\frac{{\epsilon_e}^{0.81} + 0.26}{{\epsilon_e}^{0.81} - 0.189}\frac{u^{0.8544} + 0.236}{u^{0.8544} + 0.87} \\
\zeta_2 &= 1 + \frac{u^{0.371}}{2.358\epsilon_r + 1} \\
\zeta_3 &= 1 + \frac{0.5274 \arctan\left(0.084u^{\frac{1.9413}{\zeta_2}}\right)}{{\epsilon_e}^{0.9236}} \\
\zeta_4 &= 1 + 0.0377 \arctan\left(0.067u^{1.456}\right)\left(6 - 5e^{0.036(1-\epsilon_r)}\right) \\
\zeta_5 &= 1 - 0.218e^{-7.5u}
\end{aligned}
\tag{3.5}
$$

Completing the first iteration of this process provides the standard half wavelength RMSA. This design method can also be applied as an iterative process with small modifications to the patch width between iterations to achieve a desired geometry. The *baseline_antennas.m* script provides the design for the standard half wavelength RMSA, as well as an iterative process to achieve a square microstrip patch. To summarize the design steps for a patch antenna:

1. Choose the patch width. Equation (3.4) will provide the width for TM_{10} mode resonance.
2. Apply Equation (3.1) to correct for fringing fields.
3. Apply Equation (3.2) to correct for patch thickness.
4. Apply Equation (3.3) to correct for dispersion effects.
5. Apply Equation (3.5) to acquire the patch length.

The *baseline_antennas.m* script was used to develop two patch antenna designs: a half wavelength RMSA designed for TM_{10} resonance, and a square RMSA design where the patch length is equal to the patch width. Two versions of each antenna were design for different dielectric thicknesses on Roger 6010 laminate, with a ϵ_r of 10.7. These designs were then simulated and optimized with the High Frequency Structure Simulator (HFSS).

Table 3.1 shows the optimized dimensions of the half wavelength RMSA patches, and Table 3.2 shows the optimized dimensions of the square RMSA patches. While these patch antennas meet the majority of the design requirements, they are not circularly polarized.

Table 3.1: Half wavelength microstrip antenna dimensions and simulated performance for two different thicknesses of Rogers 6010 laminate.

Parameter	Value	
Dielectric Thickness	0.127 cm	0.254 cm
Patch Length (L)	1.969 cm	1.936 cm
Patch Width (W)	2.908 cm	3.102 cm
Feed Location	(0.255, 0) cm	(0.277, 0) cm
Bandwidth (-14 dB)	9.8 MHz	20.5 MHz
Gain (Linear)	4.44 dBi	5.01 dBi
Beamwidth (X,Y)	106° 69°	107° 68°

3.1.1 Circularly Polarized Antennas

Circular polarization on a patch antenna can be achieved through multiple methods. For one method, two feeds can be used to excite orthogonal modes with equal magnitudes in phase quadrature. This method can provide a very low Axial Ratio (AR) with very high bandwidth, but requires significantly more complicated feed geometry. Alternatively, a single feed can be used with modified geometry in such a way that the resonant frequencies of the two orthogonal

Table 3.2: Square microstrip antenna dimensions and simulated performance for two different thicknesses of Rogers 6010 laminate.

Parameter	Value	
Dielectric Thickness	0.127 cm	0.254 cm
Patch Length ($L = W$)	1.969 cm	1.940 cm
Feed Distance	(0.255, 0) cm	(0.283, 0) cm
Bandwidth (-14 dB)	9.8 MHz	20.9 MHz
Gain (Linear)	4.43 dBi	4.93 dBi
Beamwidth (X,Y)	107° 68°	108° 68°

modes are close to each other, on either side of f_0. The feed point is selected so that it excites the two orthogonal modes in quadrature, creating a circularly polarized signal. While the return loss bandwidth of such an antenna is very broad due to the two resonant frequencies, the AR bandwidth is very narrow [Kumar and Ray, 2003]. The single point feed designs were explored extensively in this project, and it was found that none of these designs could maintain polarization during scanning because AR degrades away from broadside. A recommendation is made in Chapter 7 to explore dual feed designs as future work.

3.2 Prototype Array Design

An example antenna array design is presented here. This array uses the rectangular microstrip design described in Section 3.1, with further optimization after arraying. The array spacing was set to 0.44 wavelengths (6 cm) to allow for adequate spacing between elements while keeping the array compact. The final optimization resulted in a patch length of 1.932 cm, a width of 1.916 cm, a feed location at (0.237, 0) cm, and a thickness of 2.54 mm. The HFSS model of the antenna is shown in Figure 3.1.

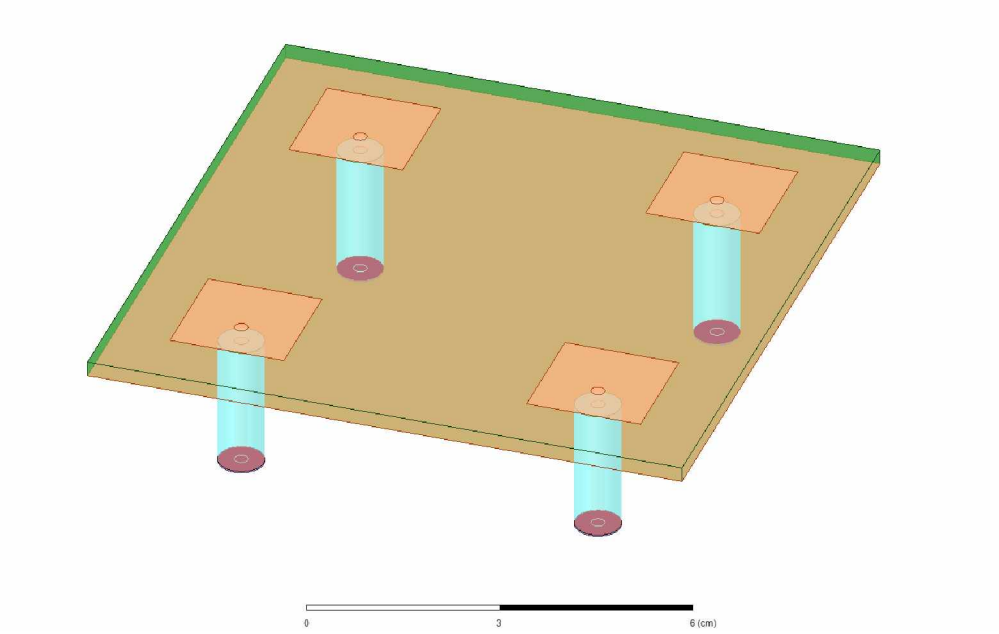


Figure 3.1: HFSS model of exemplary antenna array.

The array performance was simulated at broadside and various steering angles. Figure 3.2 shows the return loss of the array across frequency, and Figure 3.3 shows the return loss in a Smith chart format. The simulated bandwidth was 16 MHz, meeting the design requirements of 5 MHz.

The fixed array pattern when steered to broadside is shown in Figure 3.4. The peak gain of the antenna is 8.14 dBi with a beamwidth of 56°. The array was steered to various angles to characterize steering performance. Figure 3.5 shows the beam pattern when the array is steered to 30° off of broadside. With beamforming, the effective HPBW of the array increases to 82°. Note that the HFSS plots use a different coordinate system than elsewhere in this thesis.

Table 3.3 shows the antenna gain at various steer angles and DOAs. There is an interesting anomaly when the DOA is at 60°, the best gain does not actually occur when the array is steered to 60°. This is because as the array steers away from broadside, there is a loss in the signal polarization and power is transmitted in the cross-polarized signal. At 60° the useful steering range of this array has been exceeded, and performance is lost due to the cross-polarization leakage.

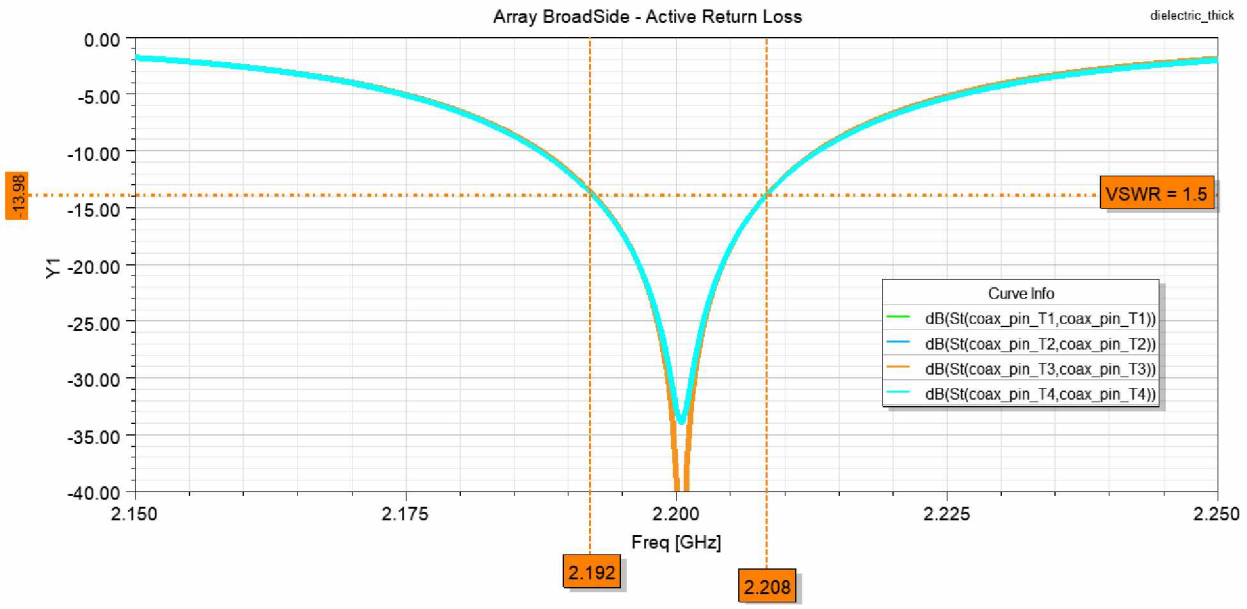


Figure 3.2: Broadside active return loss of example antenna array. 0° AZ cut (red) and 90° AZ cut (green).

Table 3.3: Antenna gain (dBi) at various scan angles and DOA angles.

		DOA Angle					
		0°	10°	20°	30°	41°	60°
Steer Angle	0°	**8.14**	7.74	6.54	4.57	1.50	-6.19
	10°	7.91	**8.07**	7.41	5.94	3.44	-3.44
	20°	7.17	7.90	**7.72**	6.67	4.52	-1.80
	30°	5.94	7.30	7.59	**6.90**	5.07	-0.86
	41°	4.03	6.22	7.03	6.71	**5.18**	-0.40
	60°	-0.43	3.71	5.50	5.78	4.69	**-0.44**

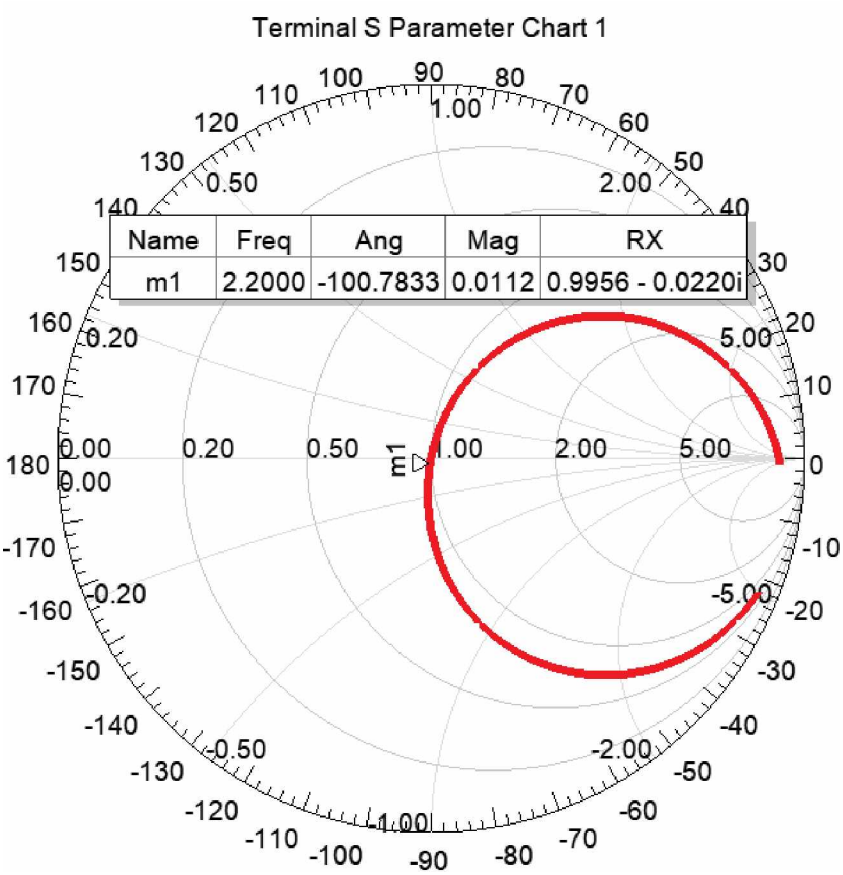


Figure 3.3: Broadside active smith chart of example antenna array.

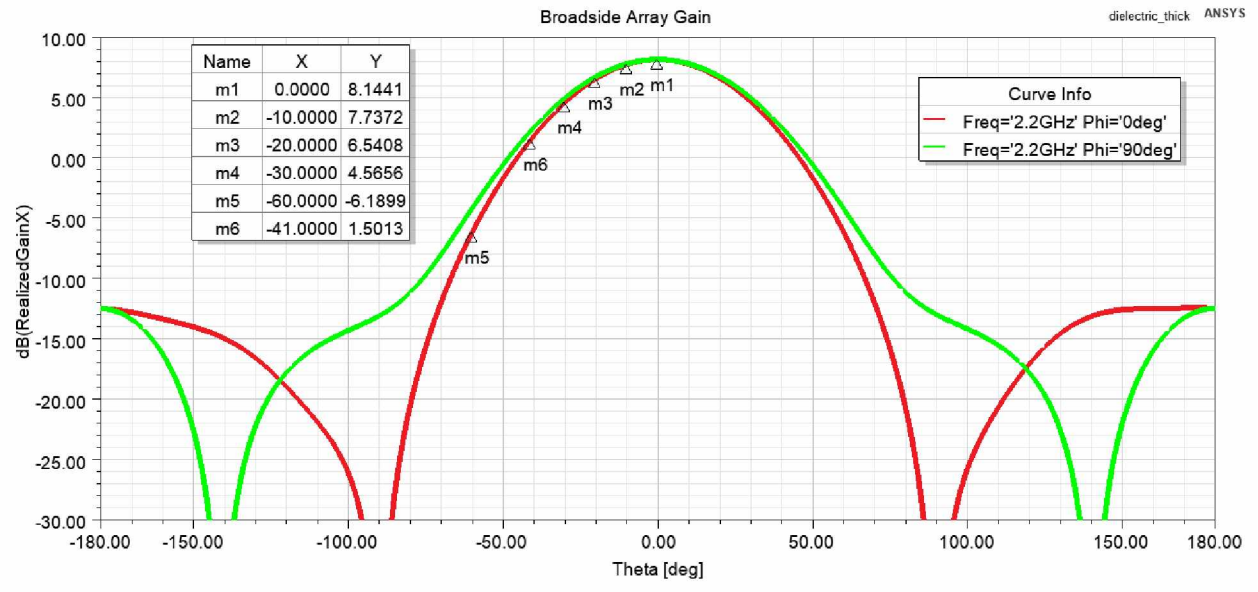


Figure 3.4: Broadside gain of example antenna array, X-axis elevation cut (red) and Y-axis elevation cut (green).

Figure 3.5: Gain of example antenna array when steered 30 degrees off broadside, X-axis elevation cut (red) and Y-axis elevation cut (green).

3.3 Front-End Electronics

All of the components and architectures shown in Chapter 2 were analyzed to determine the combination of components and architecture that best optimized phase error, power output, gain, and efficiency. For the prototype array designed in this project, the architecture shown in Figure 3.6 was implemented. This architecture was selected because it allows high output power and utilizes a single Transmit/Receive (TR) path with simple circuitry.

The component trade studies and final selection was shown in Section 2.4.2. The following subsections further discuss design considerations for the selected components.

3.3.1 Phase Shifter - Peregrine Semiconductor PE44820

The PE44820 is made by Peregrine Semiconductor specifically for phased array systems in the 1.7 to 2.2 GHz frequency range, with an extended range up to 3 GHz. The package is a 32 pin 5mm x 5mm Quadrature Flat No-Leads (QFN) package, and no external circuitry is required. The phase shifter can be powered through a 3.3 V or 5 V rail. A 3.3 V rail is recommended and used for the prototype.

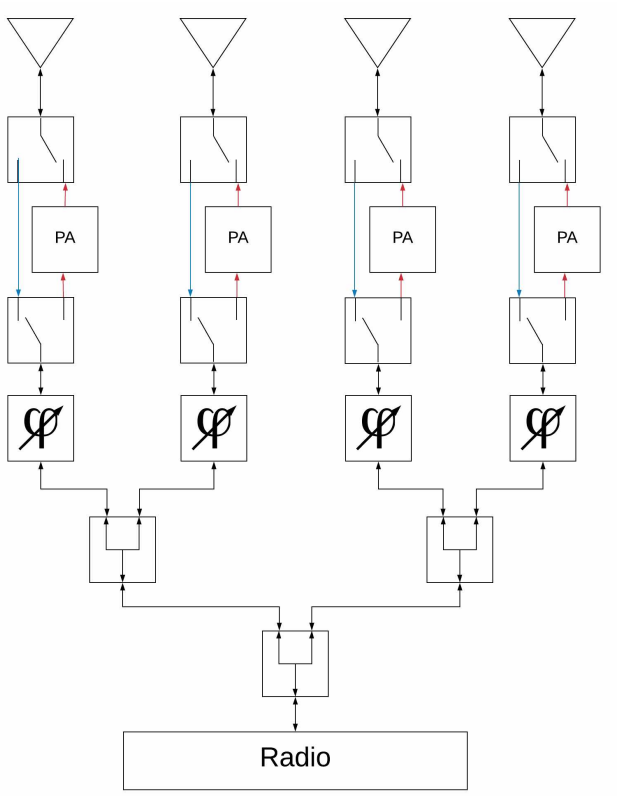


Figure 3.6: Prototype front-end electronics architecture.

The phase shifter has both a serial mode and parallel mode programming option. In this design, the parallel programming method is used as it results in both simpler hardware and controller software. Eight data bits are used to program the phase, with the same eight lines being used to program all phase shifters. Each phase shifter has a unique latch enable. To set the phase of a specific phase shifter: the parallel data bits are set, and then the latch enable for the desired phase shifter is toggled. The data bits must be held for 100 nanoseconds before and 100 nanoseconds after the latch enable. The latch enable must be held for at least 30 nanoseconds. As the max clock speed of the microcontroller is roughly 20 MHz (50 ns period), it is necessary to ensure in software that at least two full clock periods are passing between the latch enable and data programming activities.

There is an optional 9th data bit, known as the OPT bit. The OPT bit can be used, along with a calibration table that can be requested from Peregrine Semiconductor, to significantly improve performance at a given design frequency. The OPT bit was not used (i.e. was tied to the 8th data bit) in the prototype design, but is recommended for future work in Chapter 7.

Table 3.4 shows the data sheet specifications for the PE44820 using the selected parallel programming interface at the design frequency. The table assumes a 25°C operating temperature.

3.3.2 High Power Amplifier - SkyWorks SKY66294-11

The SKY66294-11 was originally designed for LTE small cell base stations in the 2.0 to 2.3 GHz frequency range. The device comes in a 16 pin 5mm x 5mm QFN package. Biasing circuitry is included internally to compensate the performance over temperature and voltage variations. Matching networks are also internal to the chip, making the only required external circuitry the bypass capacitors and decoupling capacitors on the supply voltages. The power amplifier uses low voltage control signals, which are not available on the selected microcontroller. Logic level transformation is necessary to interface with the power amplfiier.

The chip includes a Power Amplifier Enable (PAEN) pin that enables or disables the amplifier gain. However, even in a disabled state the device draws 75 mA (0.375 W), so it is recommended

Table 3.4: PE44820 phase shifter specifications for selected usage, assuming 25 °C operation.

Parameter	Value	Units
Supply Voltage	3.3	V
Supply Current	130	μA
Digital Control Voltages	0 and 3.3	V
Maximum Input Power	25	dBm
Operating Temperature	-40 to +105	°C
Minimum Latch Enable Pulse Width	30	ns
Phase Resolution	1.4	°
1.4°Bit Phase Accuracy	-0.7	°
2.8°Bit Phase Accuracy	-0.3	°
5.6°Bit Phase Accuracy	0.4	°
11.2°Bit Phase Accuracy	1.2	°
22.5°Bit Phase Accuracy	0.3	°
45°Bit Phase Accuracy	0.3	°
90°Bit Phase Accuracy	-1.2	°
180°Bit Phase Accuracy	0.7	°
RMS Phase Error	1.6	°
RMS Amplitude Error	0.25	dB
Return Loss S11	17	dB
Return Loss S22	14	dB
Insertion Loss (Reference State)	6.5	dB

in Chapter 7 that the design includes a 5V switchable power supply. Table 3.5 shows the data sheet specifications of the SKY66291-11 at the design frequency and power levels, assuming operation at 25°C.

Table 3.5: SKY66291-11 power amplifier specifications for selected usage, assuming 25 °C operation.

Parameter	Value	Units
Supply Voltage	5	V
Quiescent Current	75	mA
Digital Control Voltages	0 and 2	V
Power Added Efficiency (0.25 W)	24	%
Power Added Efficiency (0.5 W)	33	%
Gain	35	dB
Operating Temperature	-40 to +100	°C
Return Loss S11	17	dB

The SKY66294-11 has not been designed for phased array applications, meaning that no information was available for performance of the SKY66294 regarding phase imbalance. This makes calibration for the power amplifier challenging, as it is unknown which variables affect the phase imbalance.

3.3.3 Switch - Skyworks SKY13330

The SKY13330 is a wide-band silicon-on-insulator Single-Pole, Double-Throw (SPDT) switch. It comes in a 2mm x 2mm QFN package. It uses a 3.3 V power supply, and no external circuitry. The switching time of this device is 1.7μs. As the max clock speed of the microcontroller is roughly 20 MHz (50 ns period), it is necessary to ensure in software that at least 34 full clock periods passed after the switching activity and before activating the power amplifier. Like the SKY 66294-11 power amplifier, the switches also uses low voltage control signals that are not available on the selected

microcontroller. Logic level transformation is required to provide these voltages. Table 3.6 shows the data sheet specifications for the SKY13330 at the design frequency.

Table 3.6: SKY13330 switch specifications at design frequency.

Parameter	Value	Units
Supply Voltage	3.3	V
Quiescent Current	75	mA
Digital Control Voltages	0 and 2	V
Switching Speed	1.7	μs
Maximum Input Power	40	dBm
Input to Output Isolation	30	dB
Insertion Loss	0.5	dB
Operating Temperature	-40 to +85	°C
Return Loss S11	27	dB

3.3.4 Splitter - Skyworks SKY16406-381LF

The SKY16406-381LF is a 2-way Wilkinson-like power divider designed for satellite communication purposes in the 2.2 to 2.8 GHz frequency range. The package is a 6 pin Dual Flat No-Leads (DFN). The device is passive, and requires no external circuitry. Table 3.7 shows the data sheet specfications for the SKY16406-381LF at the design frequency.

Table 3.7: SKY16406-381LF splitter specifications at design frequency.

Parameter	Value	Units
Digital Control Voltages	0 and 2	V
Maximum Input Power	34	dBm
Isolation	17	dB
Insertion Loss	0.2	dB
Phase Balance	3.5	°
Amplitude Balance	0.2	dB
Operating Temperature	-55 to +105	°C
Return Loss S11	20	dB
Return Loss S22	37	dB

3.3.5 Controller - Texas Instruments MSP4306779A

The Texas Instruments MSP4306779A is an ultra-low power microcontroller with a 25 MHz 32 bit CPU. The controller comes in a 100 pin LQFP package, has 62 input/output pins with port mapping capabilities, and includes UART, IrDA, SPI, and I^2C interfaces. The controller has 512 KB of single-cycle flash, and 32 KB of RAM with single-cycle access. Some external circuitry is required (decoupling capacitors, watch crystal, voltage regulation, UART interface, and JTAG programming interface). All of the necessary external circuitry is included on the SSEP daughter board. Table 3.8 shows the data sheet specifications of the MSP4306779A for the SSEP development board implementation.

Table 3.8: MSP4306779A microcontroller specifications for selected usage.

Parameter	Value	Units
Supply Voltage	3.3	V
Supply Current	100	μA
Digital High Voltages	2.75 to 3	V
Digital Low Voltages	0 to 0.25	V
Maximum Clock Speed	25	MHz
Flash Memory	512	KB
RAM	32	KB
Operating Temperature	-40 to +85	°C

Chapter 4

Prototyping

This chapter discusses the prototyping method and fabrication.

4.1 Component Boards

A prototype was constructed by building several independent component level boards, and then connecting these boards together to create a functional prototype system, see Figure 4.1. All of these boards are designed to be interfaced and controlled through a SSEP development board, which utilizes the MSP4306779A microcontroller.

An additional board expands the available header connections on the SSEP development board, allowing all of the component level board to be controlled through a single microcontroller. The following subsection will detail the design and fabrication of each independent board.

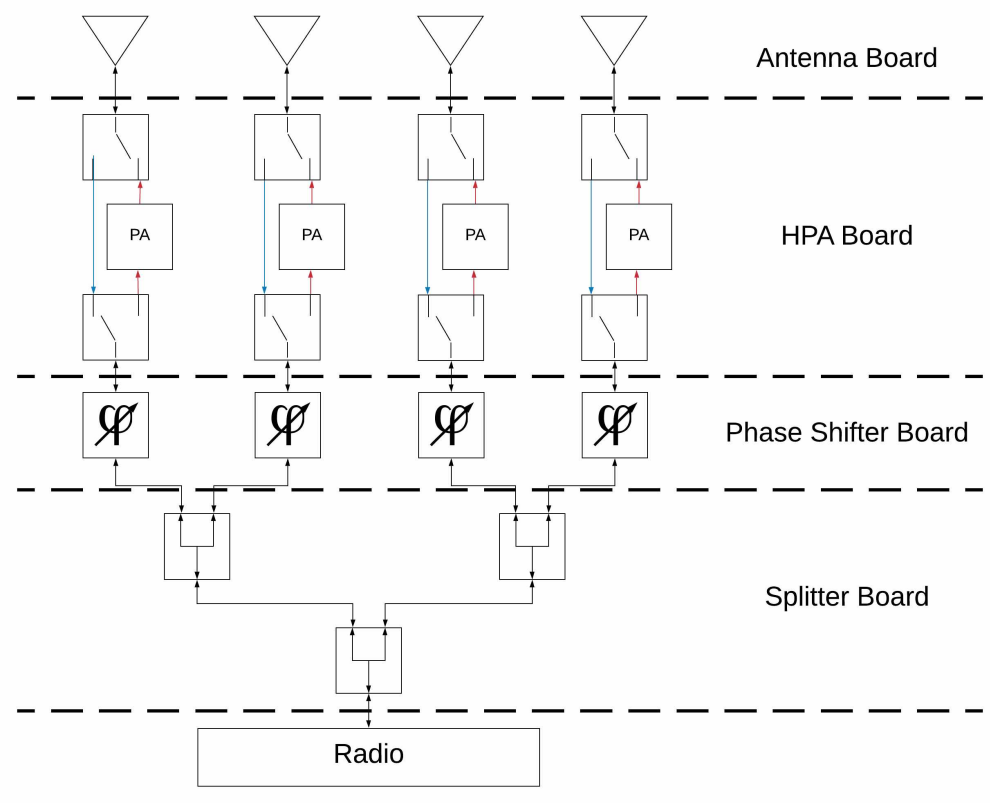


Figure 4.1: Breakdown of the component level boards.

4.1.1 SSEP Development Board

The SSEP development board was already on hand with existing hardware templates and software libraries. The development board was developed in-house by SSEP for rapid prototyping and testing of satellite subsystems and other projects. The board uses a MSP4306779A microcontroller, where all of the prototype code is executed. It provides breakout connections for several of the microcontroller input/output pins, which were used to connect to the component board. These include a regulated 3.3V bus and a pin for external 5V supply, which will be needed to power the front end electronics.

4.1.2 HPA Board

The HPA component board consists of four channels of power amplifiers with TR switching networks. Figure 4.2 shows the schematic of one HPA channel. The board draws 3.3 V and 5 V from the daughter header, though the 5 V line is a pass-through to an external supply. Each amplifier uses several decoupling capacitors at the three voltage biasing stages. These capacitor values come from data sheet recommendations. Port 7, pins 0 through 2 (P7.[0..2]), of the SSEP daughter header is used for control of the HPA board. Each channel has a pull pin on the PAEN line, allowing the power amplifier to be physically disconnected in the events of overpower or erroneous transmissions during testing. The control logic of the switches is tied together, allowing all switches to be controlled through a single control line and single enable line. MMCX Radio Frequency (RF) connectors are used on both sides of the switches to provide the RF input and output.

Figure 4.3 shows the layout of the HPA component board. The board is a four layer FR-4 design, with signal layers on the top and bottom, and ground and power planes for the inside layers. All components are placed on the top layer, between the SSEP daughter headers. A copper ground pour surrounds the components on the top layer, to improve power dissipation and signal isolation. A via matrix exists between the top layer copper ground pour and the internal ground

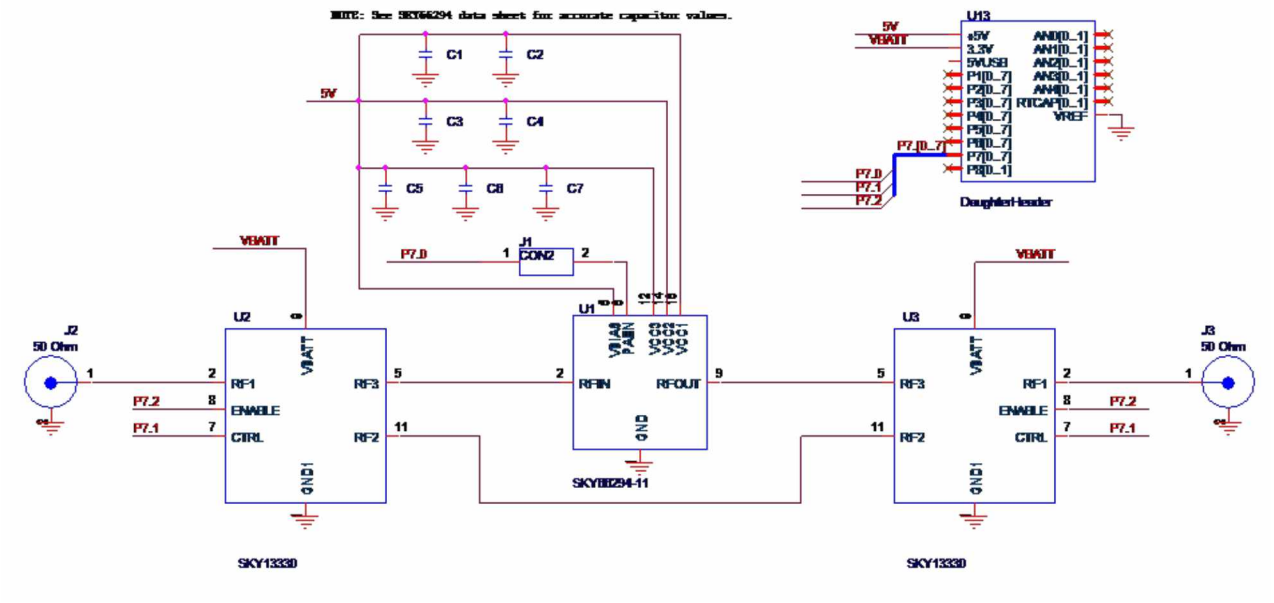


Figure 4.2: Schematic for one channel of the HPA prototype board.

plane. All RF traces are sized for 50 Ω impedance. It was ensured that the phase length of each trace section was identical between channels to maintain phase balance.

4.1.3 Phase Shifter Board

The phase shifter component board consists of four channels of phase shifters. Figure 4.4 shows the schematic of the phase shifter component board. The four phase shifters are connected in parallel. Port 5 of the SSEP daughter header provides the 8 data bits. The OPT bit is synchronized to the 90° bit (P5.1) for standard operation, as per the PE44820 data sheet. Port 6, pins 0 through 3 (P6.[0..3]), are used to provide the individual latch enable to each pin. The 3.3 V supply is provided through the SSEP header. MMCX connectors are placed at each end of the phase shifter to provide signal input and output. The serial (SI) and clock (CLK) pins of the phase shifters are left floating, as they are not used for parallel programming mode. The serial/parallel select pin (S/P) is tied to ground to keep the phase shifters in parallel mode. The exterior bias voltage (VSS_EXT) is tied to ground as recommended for standard operations in the PE44820 data sheet.

Figure 4.5 shows the layout of the phase shifter component board. The board is a four layer FR-4 design, with signal layers on the top and bottom, and ground and power planes for the

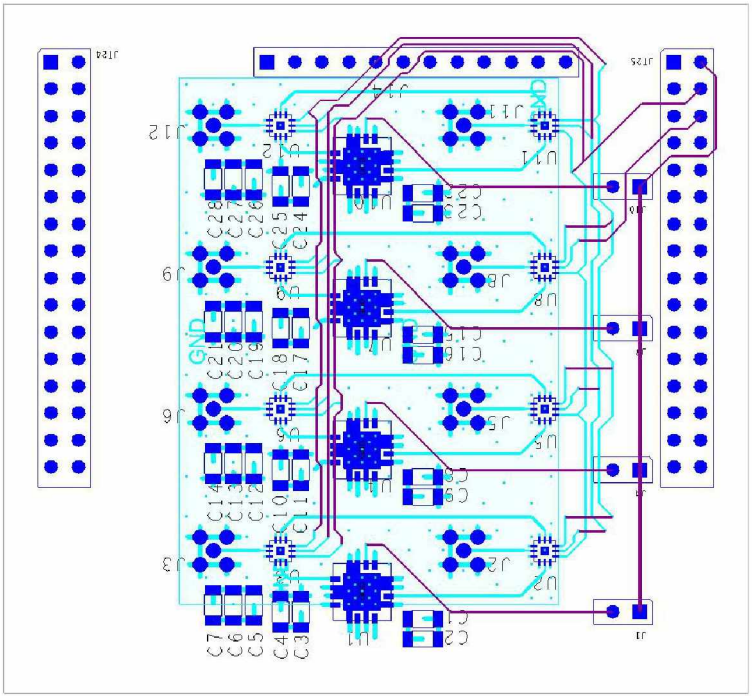


Figure 4.3: Board layout of the HPA prototype board.

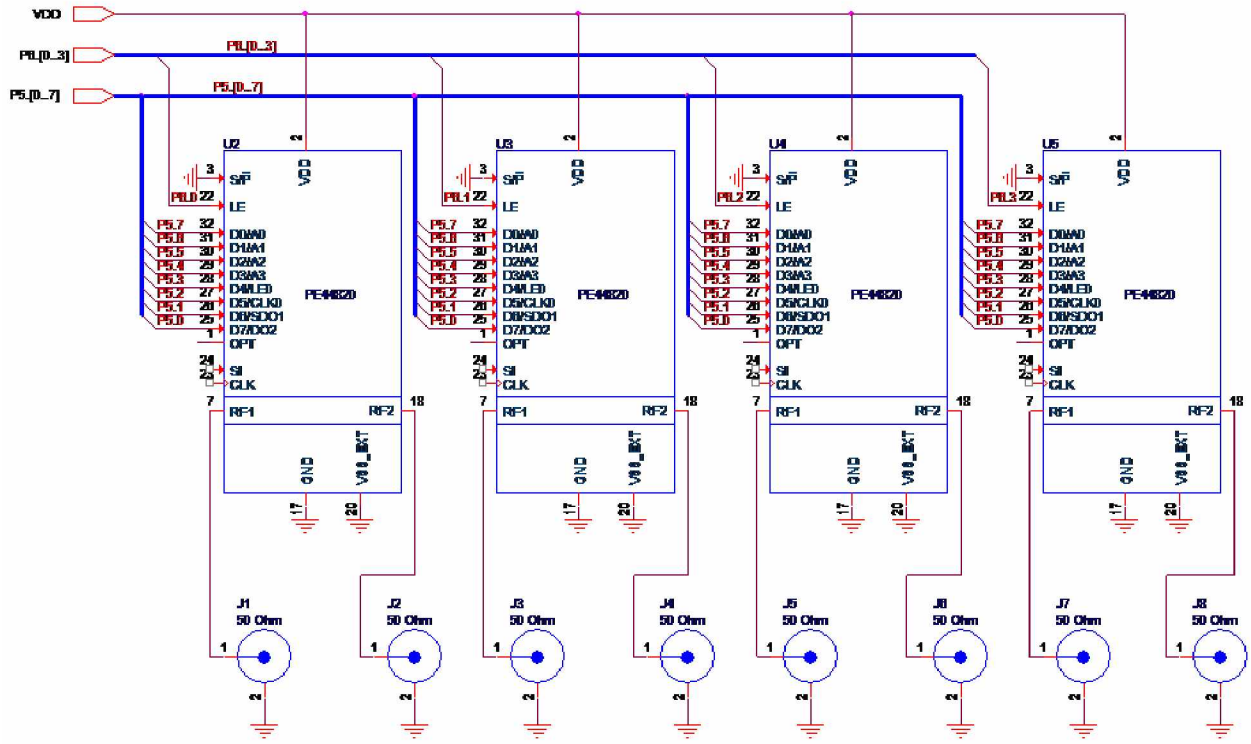


Figure 4.4: Schematic of the phase shifter prototype board.

inside layers. All components are placed on the top layer, between the SSEP daughter headers. To improve channel isolation, each phase shifter channel exists on an isolated finger of the internal ground plane with voids between channels. The fingers are connected at one side of the board, away from all RF traces. All RF traces are sized for 50 Ω impedance. It was ensured that the phase length of each trace section was identical between channels, to maintain phase balance.

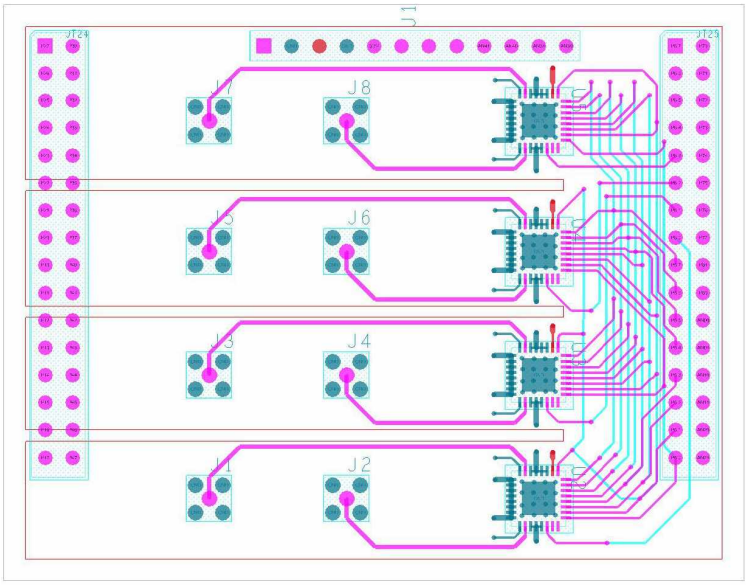


Figure 4.5: Board layout of the phase shifter prototype board.

4.1.4 Splitter Board

The splitter component board uses three signal splitter/combiners to merge the four separate channels into one. A power detector was also included on the board, but was not used for any testing presented in this thesis. A switch provides the choice between the internal power detector or an MMCX connector for external measurements. Figure 4.6 shows the schematic of the splitter component board. P7.3 of the daughter header is the enable pin of the switch, and P7.4 is the control pin. A matching network is included to match the impedance to the power detector, specified by the power detector datasheet. P7.5 is the enable pin of the power detector. The analog output of the power detector is connected to AN0.1, an analog port of the SSEP header. The daughter header

provides 3.3 V for the switch and 5 V for the detector. Several decoupling capacitors are included on the 5 V line, as recommended by the datasheet.

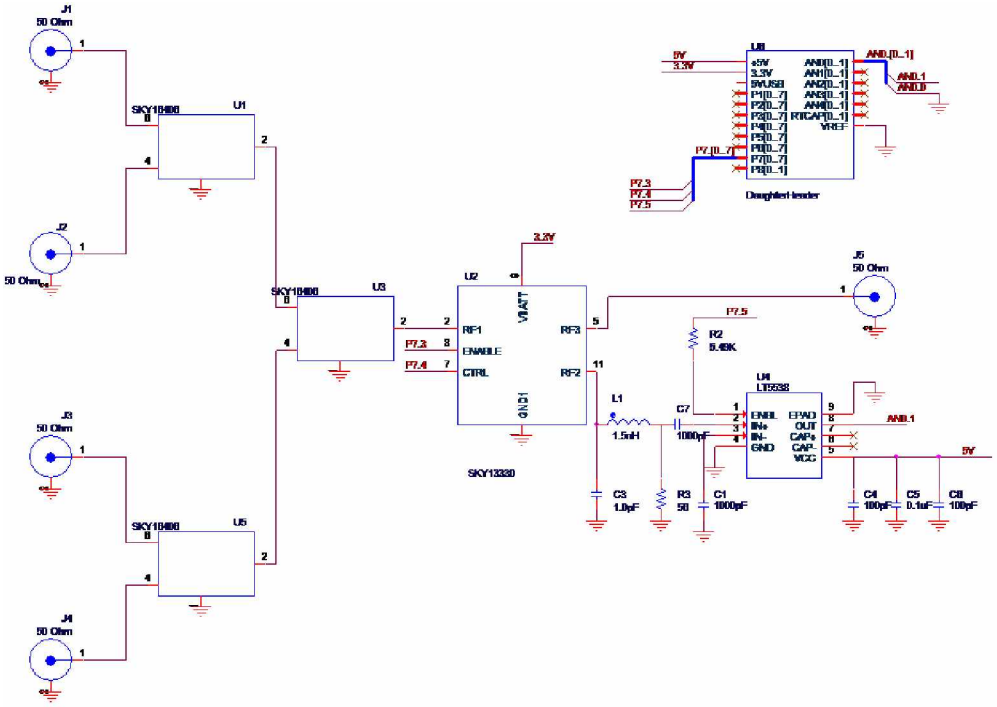


Figure 4.6: Schematic of the splitter prototype board.

Figure 4.7 shows the layout of the splitter component board. The board is a four layer FR-4 design, with signal layers on the top and bottom, and ground and power planes for the inside layers. All components are placed on the top layer, between the SSEP daughter headers. All RF traces are sized for 50 Ω impedance, and are surrounded by two rows of via fencing to improve isolation. It was ensured that the phase length of each trace section was identical between channels, to maintain phase balance.

4.1.5 Expansion Board

In order to combine multiple component level boards onto a single development board, a daughter expansion board was created. This board expanded the number of daughter boards that could be connected to a single development board from one board to three, using a fanout of the daughter

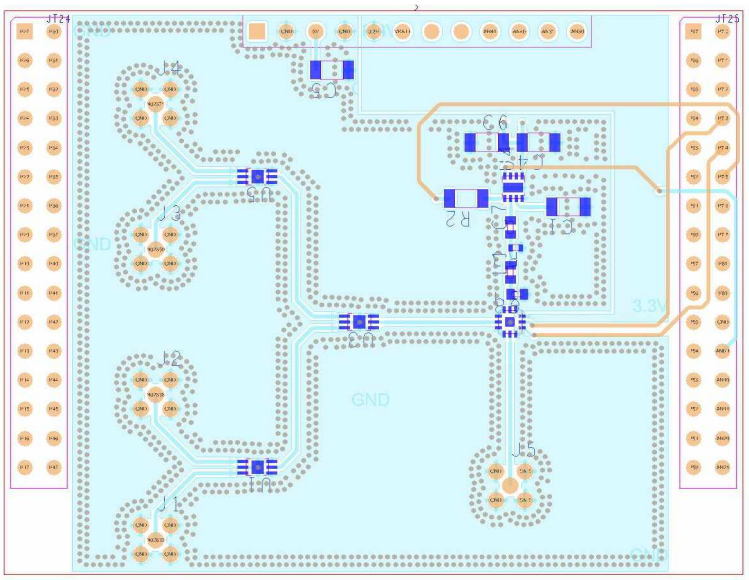


Figure 4.7: Board layout of the signal splitter prototype board.

headers. Figure 4.8 shows the schematic of the expander board. Figure 4.9 shows a fabricated expander board with all three component level boards mated.

4.2 System

The complete RDA prototype is composed of the five component boards: the HPA board, phase shifter board, and splitter board mate to the top of the expander board. The SSEP development board mates to the bottom of the daughter expander board. Four MMXC to MMCX cables connect the splitter board to the "radio" ports of the phase shifter board. Four more MMCX to MMCX cables connect the "antenna" ports of the phase shifter board to the "input" ports of the HPA board. The complete prototype cabling is shown in Figure 4.10. Because of calibration, it is critical to ensure that the proper channels of each board are connected (i.e. HPA channel 1 to phase shifter channel 1).

4.3 Software

The project code exists as register based C on the MSP430 microcontroller, or MATLAB code on the test computer. The following sections walk through the component software pieces, the system

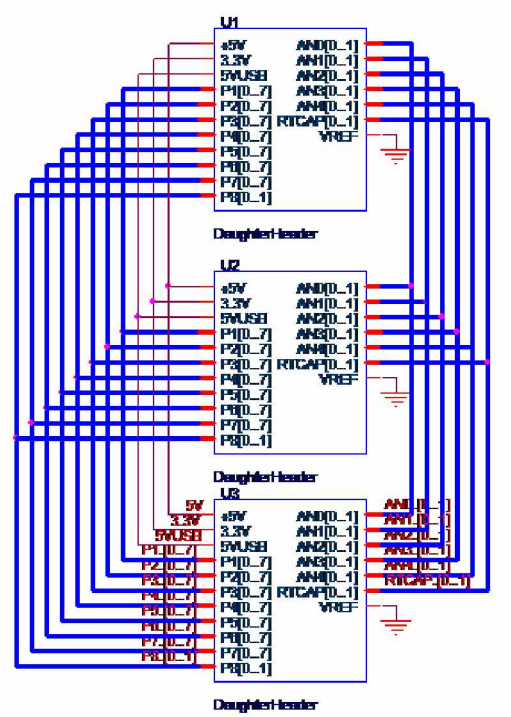


Figure 4.8: Schematic of the expander board.

level functions and algorithms, the UART interface commands, and the MATLAB algorithms and test code. Figure 4.11 shows the software block diagram for the RDA.

4.3.1 Component Software

Power Amplifier and Switches

The power amplifier uses a single enable bit to activate or deactivate the amplifier gain. At the lowest level, control of the HPA requires flipping a single bit. The switches have a control bit and an enable bit. The enable bit can be disabled to reduce power draw, but utilizing this bit will require extra clock cycles to perform switching (to first enable, and then switch). On the prototype, the enable bit is left high, and control of the switch is left to the single control bit. On the prototype, the HPA enable bit is P7.0, the switch control bit is P7.1, and the switch enable bit is P7.2. The two switches of the HPA board are logically connected. See the **TX** and **RX** commands in *RDA.c* of Appendix B.5, which are used to enable and disable the HPA as well as perform switching. After

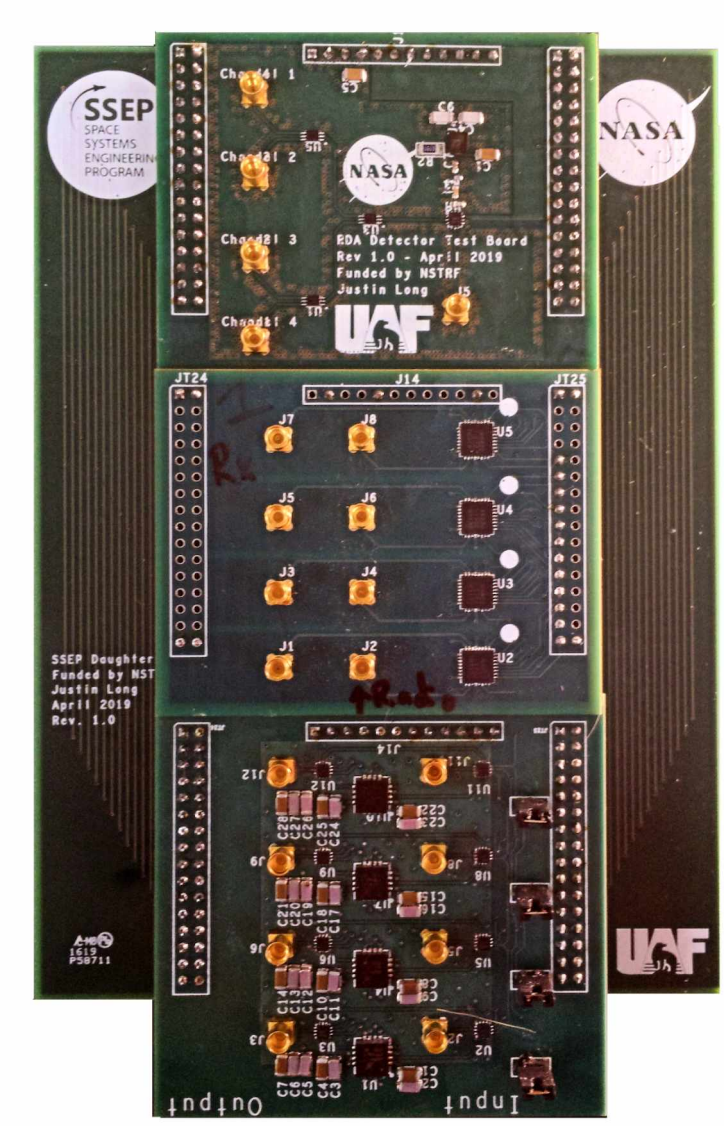


Figure 4.9: Expander board with all three component boards mated.

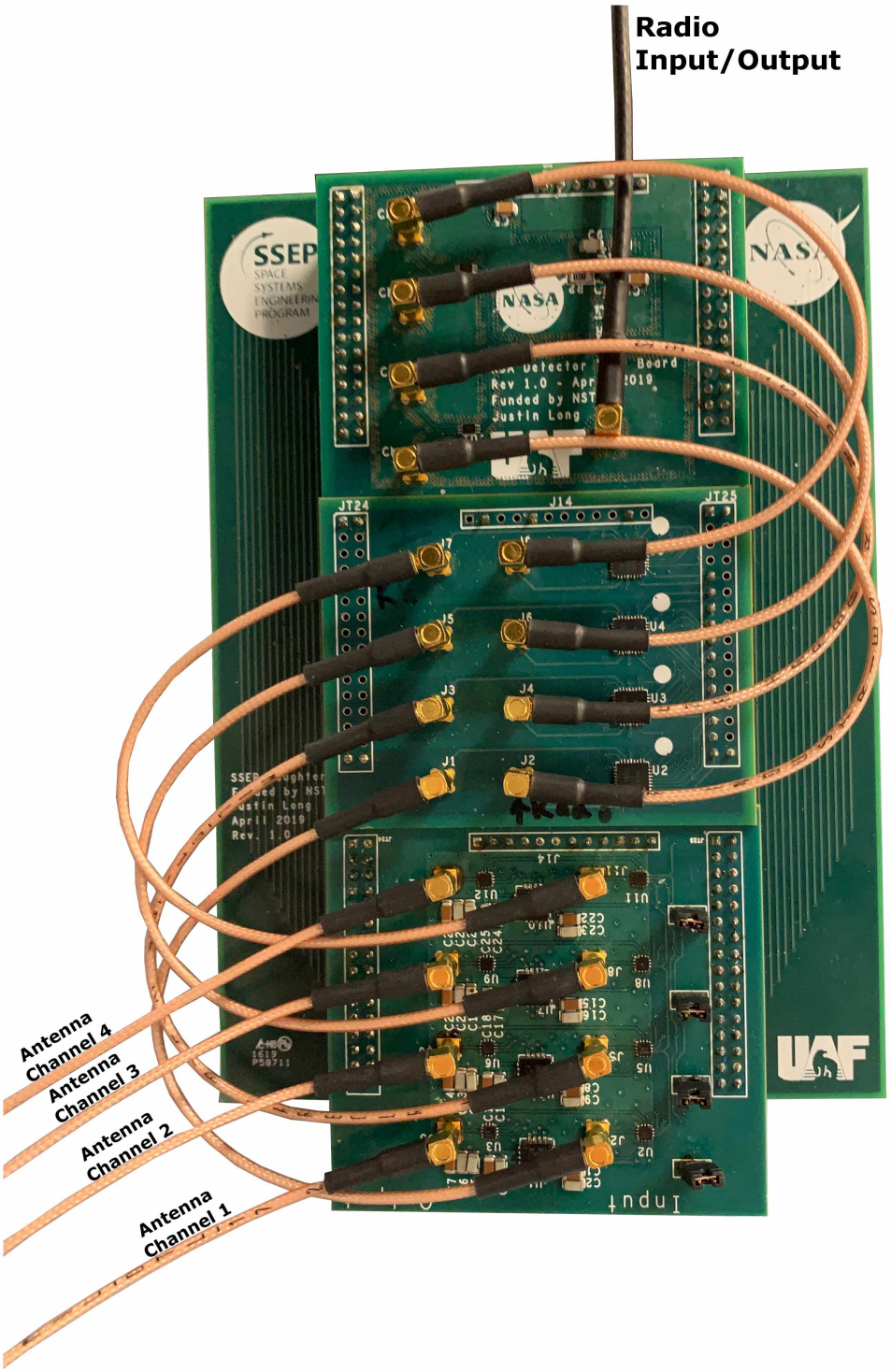


Figure 4.10: RDA prototype with all cabling.

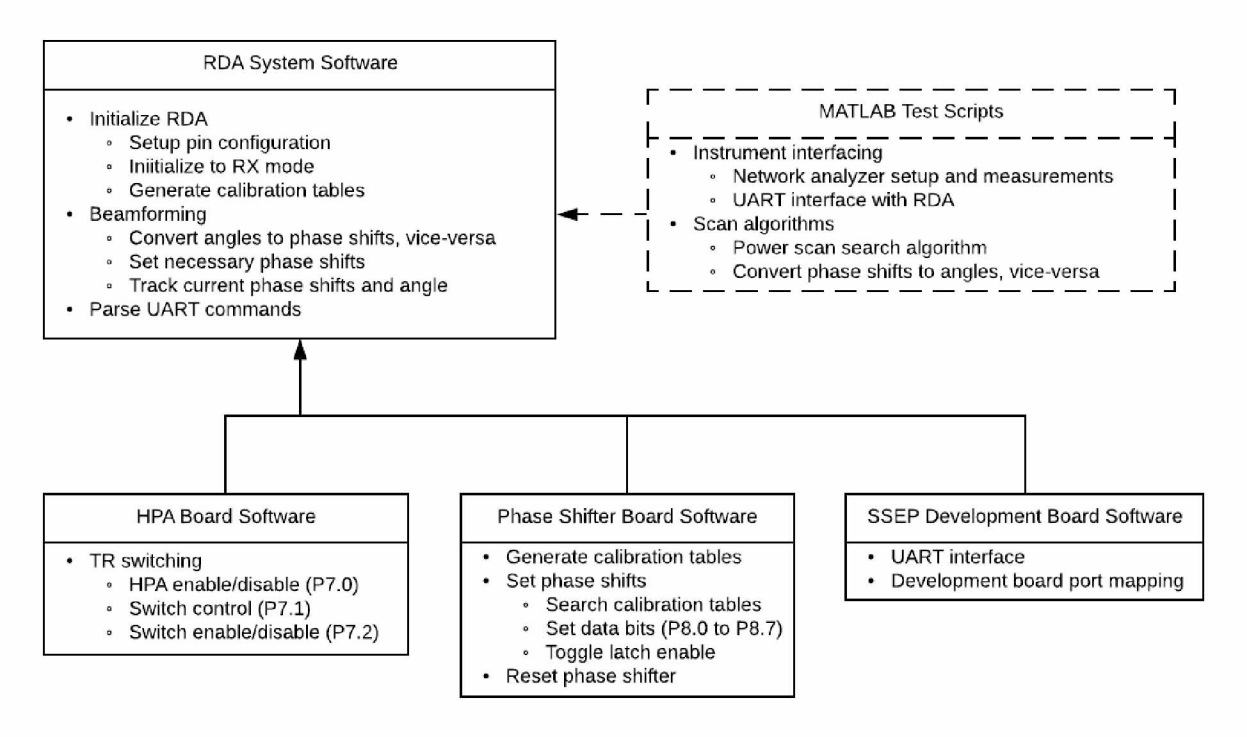


Figure 4.11: Software block diagram.

setting the switch control bits low, the **TX** command waits 50 clock cycles (2μs) before enabling the HPA to ensure that the switching action has completed. After disabling the HPA, the **RX** command waits 50 clock cycles (2 μs) to ensure power has dissipated before setting the switch control bits high.

Phase Shifter

The phase shifter component board uses port 8 of the microcontroller to set the data bits, then the phase is set by flipping the latch enable (LE) high for the desired phase shifter. There are several commands associated with the phase shifters, all of which can be seen in *RDA.c* of Appendix B.5.

The **setShift** command is used to set the phase shift of the RDA channels. It takes two inputs: **channelSelect** and **shiftSelect**. **channelSelect** is the desired channel (1, 2, 3, or 4) of the array, and **shiftSelect** is the desired phase shift. This command uses calibration lookup tables to set the data bits as close as possible to the desired phase shift. A switch statement is used to select

the appropriate calibration table, and then a *for* loop runs through the calibration table to find the value closest to **shiftSelect**. The corresponding value is stored as **dataValue**, and the index location within the array is saved as **index**. The value of **index** is then used to find and set the corresponding data bits. Lastly, a switch statement is used to flip the latch enable line for the desired phase shifter, and the global variables that track phase shift are updated with **dataValue**. These global variables are **channel1Shift**, **channel2Shift**, **channel3Shift**, and **channel4Shift**. **setPhase** will add 360 or subtract 360 from a desired phase shift as necessary, so the shiftSelect input does not need to be within a specific range.

The **calTable** command generates the calibration lookup table based on calibration measurements. The calibrated phase shift for each data bit is defined in RDA.h (see Appendix B.5). **calTable** uses a *for* loop to populate the calibration tables, which are global variables, with every possible phase shift. Each channel has a calibration table. The calibration process is discussed in Section 5.3.1.

The **resetShifter** command resets all of the phase shifters to 0° (broadside), and clears the latch enable lines.

4.3.2 System Software

The basic component code pieces are combined with RDA algorithms discussed in Chapter 1 for the prototype system software.

Setup

On startup, the RDA runs the **RDA_Setup** command, which sets up the control ports as necessary. For setup of the HPA and switches, port 7 must be setup as an output. The HPA enable pin is driven low to ensure the HPA is not on, and switch control and enable lines are set high to initialize the RDA into RX mode. For setup of the phase shifter, all of port 5 and port 6 pins 4 through 7 are set as outputs. The **calTable** command is run to generate the calibration tables, and the **resetShifter** command is run to initialize the RDA to broadside.

TR Switching

TR switching is performed with the **TX** and **RX** commands discussed in the component software.

Beamforming

The software allows two methods of beamforming: either the desired phase shifts can be specified, or the desired beamforming angles. The **setPhase** command, discussed in the component software, is used to set the desired phase shift. The **setAngle** command is used to set the desired beamforming angles. **setAngle** uses Equation (4.1) and Equation (4.2) to convert the desired angles to phase shifts, X and Y. Using the **setPhase** command, channel 1 is set to 0° , channel 2 is set to X° , channel 3 is set to Y° , and channel 4 is set to (X+Y)°.

$$X = S * sin\left(\phi * \frac{\pi}{180}\right) * cos\left(\theta * \frac{\pi}{180}\right) * 360 \tag{4.1}$$

$$Y = S * sin\left(\phi * \frac{\pi}{180}\right) * sin\left(\theta * \frac{\pi}{180}\right) * 360 \tag{4.2}$$

Since the actual phase shift won't be equal to the desired values **X** and **Y**, the global phase tracking variables are used to calculate the actual beamforming angles. The actual X becomes (**channel2Shift** − **channel1Shift**), and the actual Y becomes (**channel3Shift** − **channel1Shift**). Equation (4.3) and Equation (4.4) shows the derivation of the beamforming angles from X and Y. This process is implemnted as the command **angles** in *RDA.c*.

$$\theta = atan2\left(Y * \frac{\pi}{180}, X * \frac{\pi}{180}\right) * \frac{180}{\pi} \tag{4.3}$$

$$\phi = asin\left(\frac{\sqrt{\left(X * \frac{\pi}{180}\right)^2 + \left(Y * \frac{\pi}{180}\right)^2}}{\pi}\right) * \frac{180}{\pi} * \frac{0.5}{S} \tag{4.4}$$

UART commands

All of the commands have been mapped to UART commands. The 9600 baud UART interface is written into the SSEP development board, and is accessed with a mini-USB port on the development board. The UART commands available are listed below. Inputs to the command are surrounded by [ ] brackets.

- **"help"** provides a list of available commands.
- **"help [command]"** gives the help description for a specific command.
- **"beamform [ZE] [AZ]"** calls the setAngle command.
- **"setPhase [channelSelect] [shiftSelect]"** calls the setPhase command.
- **"readAngle"** calls the "angles" command and returns the current beamformed ZE and AZ angles.
- **"readPhase"** returns the value of the global phase tracking variables.
- **"resetPhase"** calls the "resetShifter" command.

4.3.3 Scan Algorithm

The scanning algorithm is implemented in the MATLAB code for testing purposes, see Appendix B.4. The power scanning algorithm proposed by Akagi et al. [2008] is used. The RDA is swept through the entire beam pattern, and the location of maximum power is recorded.

Chapter 5

Testing

This section discusses all testing that was performed on the prototype.

5.1 Unit Level Testing

5.1.1 Phase Shifter Board

The phase shifter component board performance was measured at the operating frequency of 2.2 GHz. Each channel of the phase shifter board was connected to a network analyzer and the phase change and IL were measured. Figure 5.1 shows the measurement of the phase error and insertion loss of one phase shifter across the phase shifter range. At each data point, 1000 measurements were made. Figure 5.1 shows the average of these measurements, and the standard deviation of the phase error across the 1000 measurements. The test was run on eight phase shifters, and all eight show similar performance. The performance is within the expectations of the data sheet. Note that the IL measurements include board and connector losses, and are not adjusted to remove these losses.

5.1.2 HPA Board

The HPA component board includes the high power amplifier and two TR switches. The Transmit (TX) path, which includes the high power amplifier, was measured using a network analyzer at the design frequency. The TX path showed a gain of approximately 33.5 dB over a range of output power from 5 mW to 4 W. The gain varied between channels by ±0.2 dB. The Receive (RX) path, which included only the two TR switches was also measured using a network analyzer at the design frequency. The total RX path loss is approximately 2.0 dB. The RX loss varied between channels by ±0.1 dB. All of these results are within expected values.

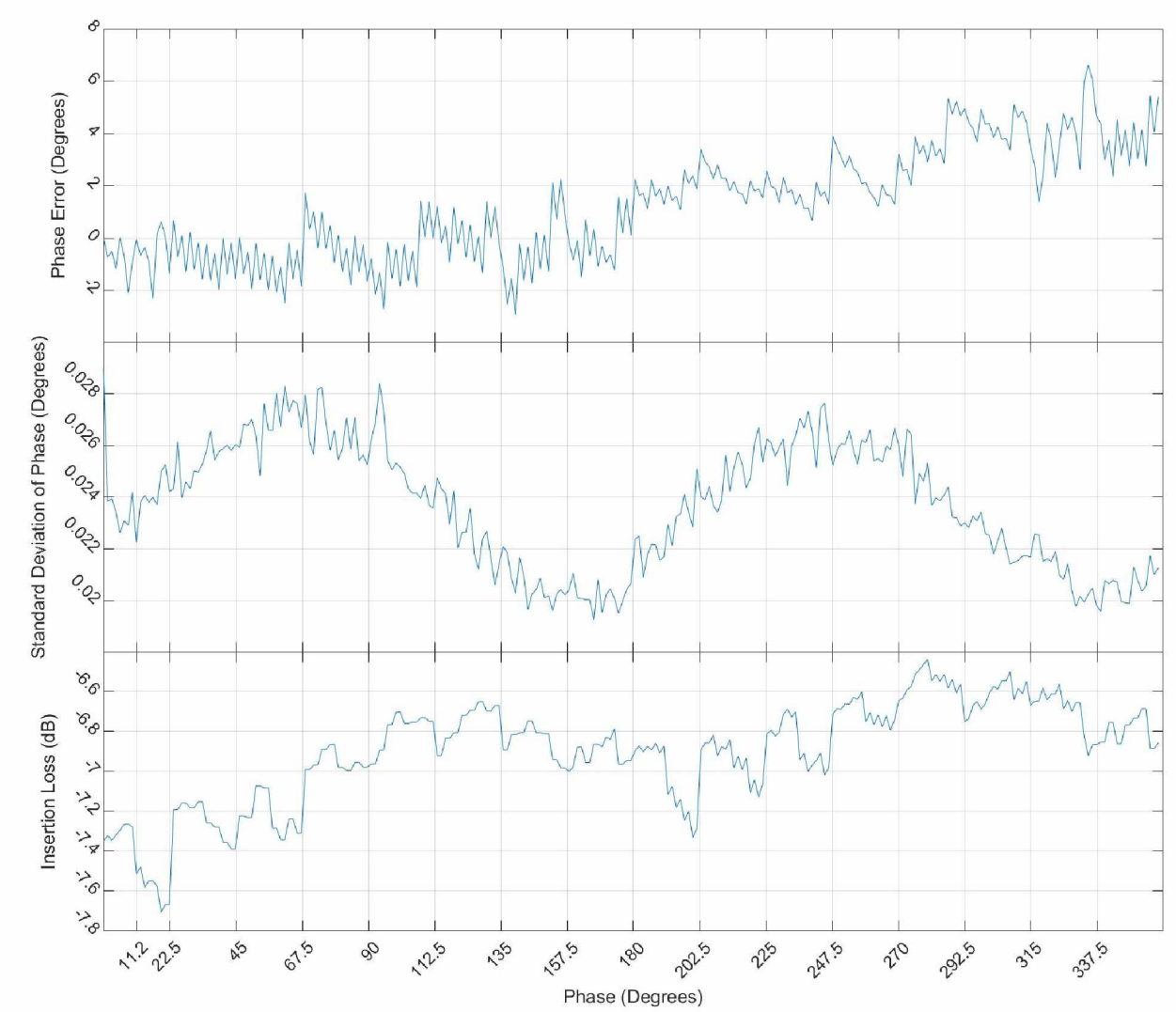


Figure 5.1: Phase shifter phase error (top), standard deviation of phase error (middle), and insertion loss (bottom).

5.2 Scanning Tests

5.2.1 Scanning Tests

The scanning accuracy of the phase shifter component board was measured using the test setup shown in Figure 5.2. A signal generator generates an unmodulated interrogator signal at 2.2 GHz. The interrogator signal is split into four channels using a Wilkinson divider. A PE44820 phase shifter component board is used to simulate the DOA for the signal. A second PE44820 phase shifter component board is used to perform the scanning operation of the receiving array. The four channels are recombined and measured in a spectrum analyzer. The test is automated through GPIB and UART connections using a MATLAB script. The script scans the receiving phase shifter array though all possible phase shifts, and measures the power at all phase shifts. The total possible phase shifts are 2^8 in the X direction and 2^8 in the Y direction, for a total of $2^8 * 2^8 = 65,536$ measurements per scan.

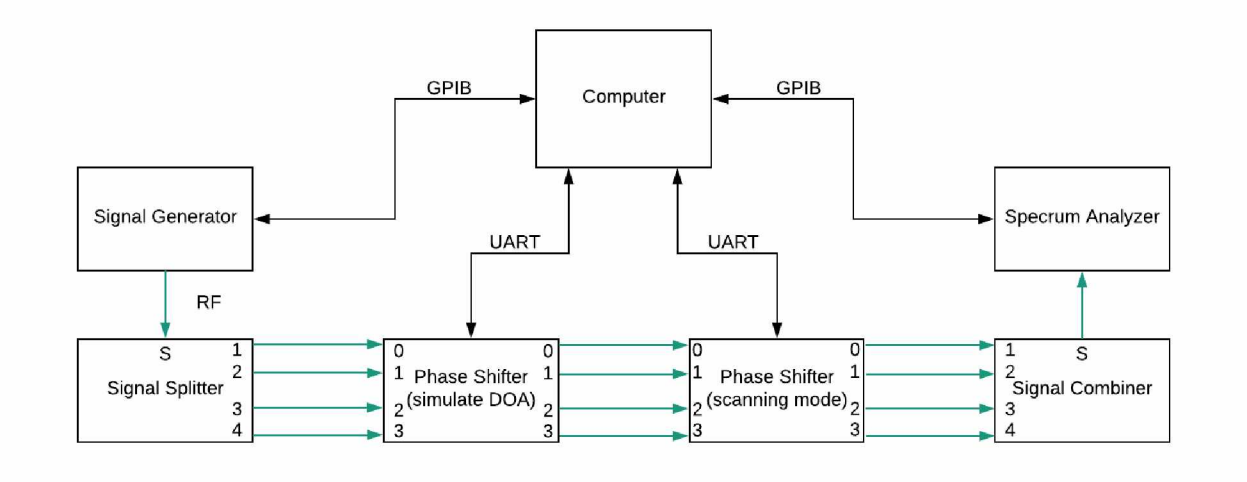


Figure 5.2: Test setup for the conceptual scanning test.

The phase error due to mechanical tolerances and uncertainty within the phase shifters was optimized through calibration. This calibration was performed by connecting each set of phase shifters, with the exact cabling used for testing, to a network analyzer and measuring the true phase shift of each phase shifter bit. There is additional error due to the amplitude imbalance across the phase range of the phase shifters, which is not calibrated for and is expected to be the

primary remaining source of error in the DOA determination. The results for several DOA angles are shown in Table 5.1. The error value is a central angle error, calculated using Equation (5.1). Some non-calibrated measurements are also presented to show the necessity of the calibration.

$$Error = cos^{-1}[cos(\phi_{DOA}) \cdot cos(\phi_{scan}) + sin(\phi_{DOA}) \cdot sin(\phi_{scan}) \cdot cos(\theta_{DOA} - \theta_{scan})] \tag{5.1}$$

Table 5.1: Test results showing angle determination error.

	DOA		Scan Result		
	$\phi(°)$	$\theta(°)$	$\phi(°)$	$\theta(°)$	**Error(°)**
Uncalibrated	0	0	5.7	225	**5.7**
	30	60	11.5	191.3	**38.5**
	45	180	29.3	141.5	**27.4**
Calibrated	0	0	1.5	276.0	**1.5**
	5.1	56.9	8.9	70.3	**4.1**
	10.2	198.0	7.4	191.9	**2.9**
	15.4	14.1	15.6	8.4	**1.5**
	30.0	59.7	32.8	56.1	**3.4**
	45.9	234.4	42.0	234.4	**3.9**
	59.7	114.7	62.7	109.4	**5.5**

These tests were presented at the 2019 IEEE Aerospace Conference, and can be viewed in the conference proceedings. The published paper is available in Appendix C.1 [Long et al., 2019].

5.3 System Level Testing

5.3.1 System Scanning Tests

A scanning accuracy test was run on the complete RDA. Figure 5.3 shows the test setup. Figure 5.4 shows a photo of the test setup in the lab. The test was performed using a network analyzer to

generate a test signal which is split into four channels by a Wilkinson divider and sent through a PE44820 phase shifter component board to simulate the DOA angle of an interrogator signal. This simulated DOA signal is received by the prototype RDA system, and the final RDA signal is measured at port 2 of the network analyzer. The RDA scans through the pattern, and measured power is recorded at all steer angles. The scripts that run the test and communicate with the network analyzer (via Ethernet) and the SSEP development boards (via UART) are shown in Appendix B.4. The network analyzer used in this test was a Keysight Technologies Field Fox.

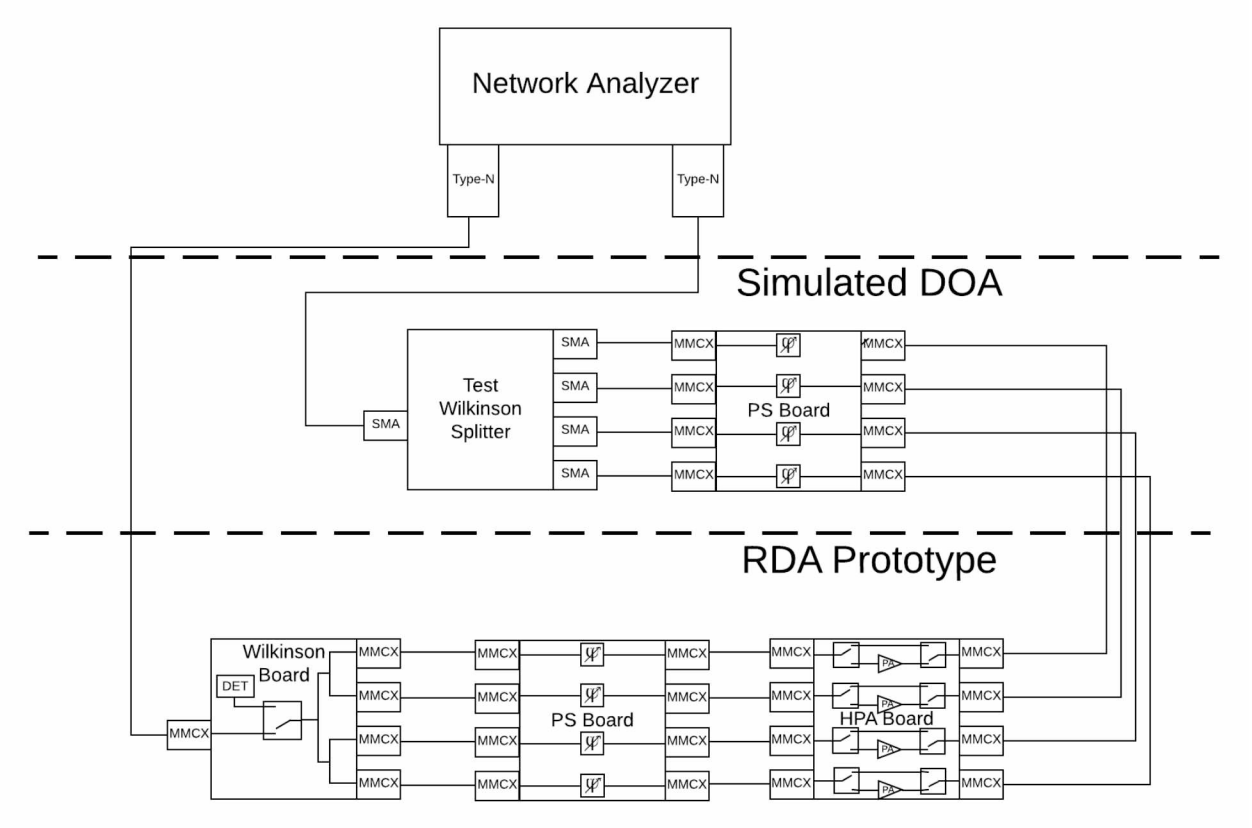


Figure 5.3: Test setup for the system scanning test.

DOA Calibration

Both the DOA and RDA require calibration. The DOA calibration setup is shown in Figure 5.5. Port 1 of the network analyzer is connected to the test splitter. The signal is divided to each channel of the phase shifter component board. Three of the channels are terminated in 50 Ω, and

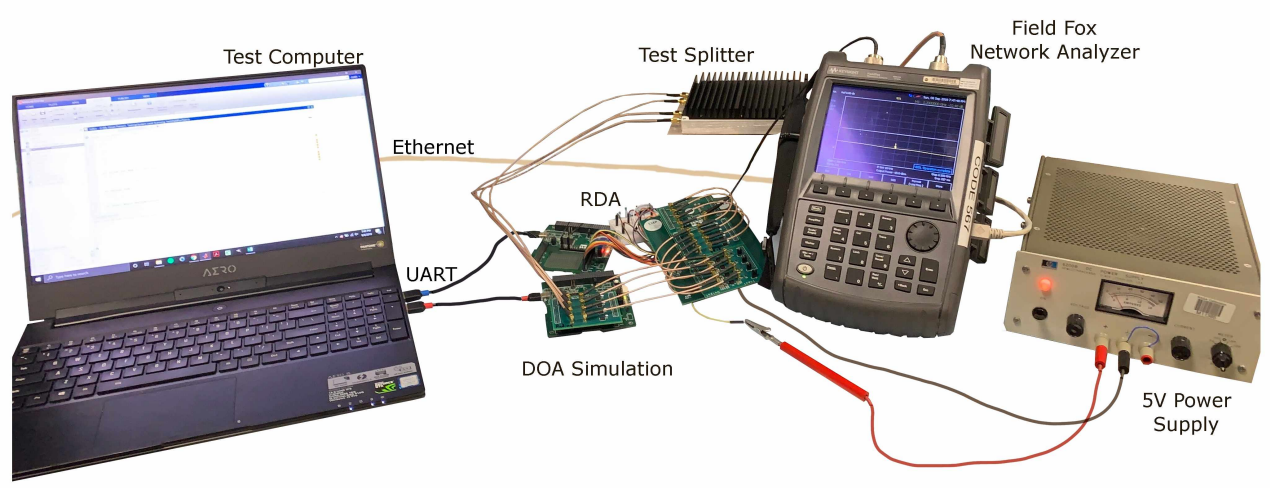


Figure 5.4: Photo of test setup for the system scanning test.

the remaining channel is connected to port 2 of the network analyzer. The exact cabling used for the system scanning test is used for calibration, from port 1 of the network analyzer up to and including the cables that connect the simulated DOA to the RDA. A MMCX to Type-N adapter is required to connect the MMCX-MMCX cable to port 2 of the network analyzer. Since this is a delta calibration (the phase of each channel is subtracted by the phase of channel 1), this adapter does not affect the calibration. MMCX to SMA adapters are required to connect the 50 Ω SMA loads to the MMCX-MMCX cables.

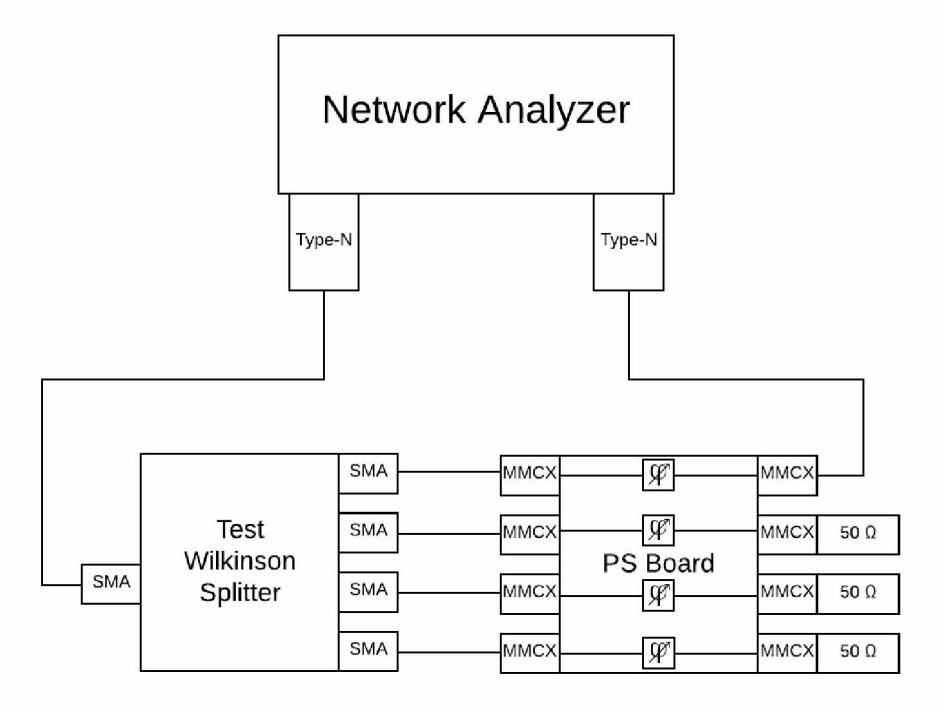


Figure 5.5: DOA calibration setup for the system scanning test.

The network analyzer is set to measure S21 phase at the design frequency. All of the phase shifters are reset to 0°, and the phase shifter connected to network analyzer is run through the calibration procedure. Each of the phase shifter bits are individually flipped, and the S21 phase measurement of the network analyzer is recorded for each bit (and no bits). This setup and calibration process is repeated for each phase shifter. Table 5.2 shows the results of the DOA calibration. The S21 Phase Measurement columns show the measurements from the network

analyzer, while the phase calibration columns shows these measurements referenced to channel 1 (i.e. -169.8° is subtracted from every value).

Table 5.2: DOA Calibration Results.

		S21 Phase Measurement(°)				Calibrated Phase Shift(°)			
Channel		1	2	3	4	1	2	3	4
	none	-169.8	-168.4	-167.5	-171	0	1.4	2.3	-1.2
	1.4	-169.1	-167.6	-166.8	-170.3	0.7	2.2	3	-0.5
	2.8	-167.2	-165.8	-165	-168.5	2.6	4	4.8	1.3
	5.6	-164	-162.8	-161.7	-165.2	5.8	7	8.1	4.6
Bit	11.2	-158	-156.9	-155.6	-159.4	11.8	12.9	14.2	10.4
	22.5	-146.5	-145.4	-144.4	-148	23.3	24.4	25.4	21.8
	45	-123.2	-122.4	-121.1	-125.5	46.6	47.4	48.7	44.3
	90	-79	-77.7	-76.6	-80.9	90.8	92.1	93.2	88.9
	180	13.4	16.9	17.3	13.6	183.2	186.7	187.1	183.4

RDA Calibration

The RDA calibration setup is shown in Figure 5.6. Port 1 of the network analyzer is connected to the combiner side of the RDA. On the HPA side, three of the channels are terminated in 50 Ω, and the remaining channel is connected to port 2 of the network analyzer. The exact cabling used for the system scanning test is used for calibration, from port 1 of the network analyzer and throughout the RDA. The MMCX-MMCX cables that connect the RDA to the simulated DOA were already accounted for in the simulated DOA calibration. A MMCX to Type-N cable is used to connect port 2 of the network analyzer to the RDA. Since this is a delta calibration (all channels are calibrated relative to each other), this cable does not affect the calibration. MMCX to SMA adapters are required to connect the 50 Ω SMA loads to the RDA.

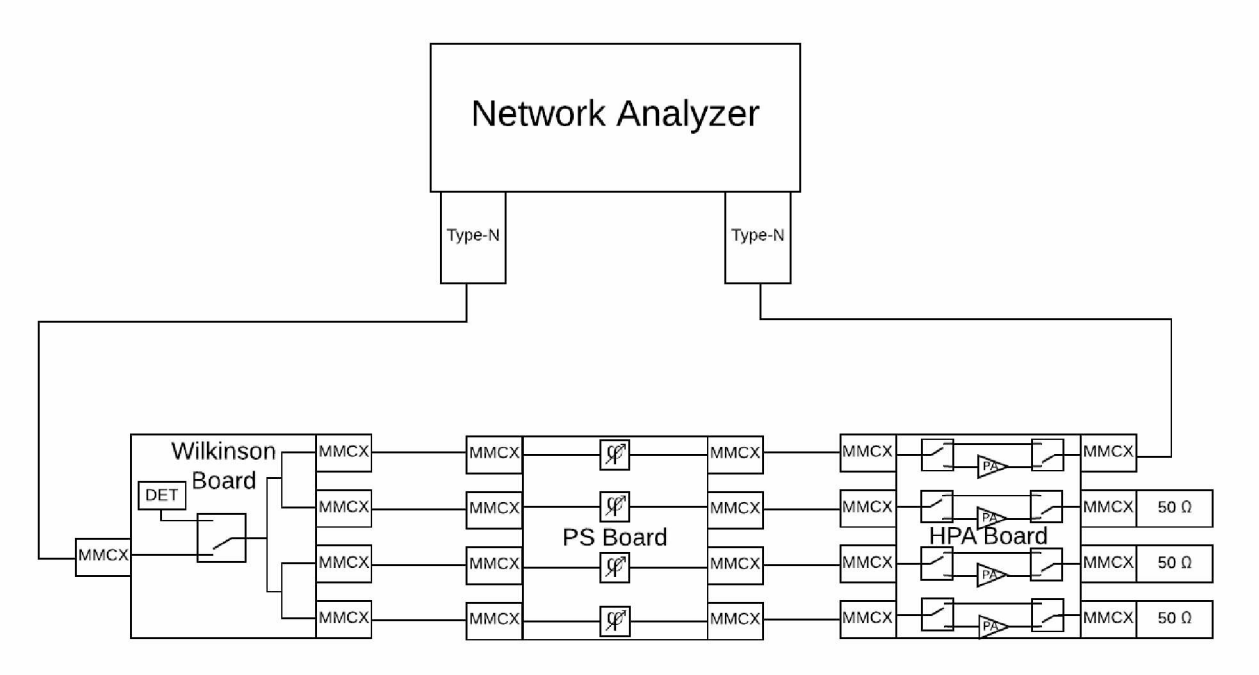


Figure 5.6: RDA calibration setup.

The network analyzer is set to measure S21 phase at the design frequency. All of the phase shifters are reset to 0°, and the phase shifter connected to network analyzer is run through the calibration procedure. Each of the phase shifter bits are individually flipped, and the S21 phase measurement of the network analyzer is recorded for each bit (and no bits). This setup and calibration process is repeated for each phase shifter. Table 5.3 shows the results of the RDA calibration. The S21 Phase Measurement columns show the measurements from the network analyzer, while the phase calibration columns shows these measurements referenced to channel 1 (i.e. -11.3° is subtracted from every value)

Results

The *fieldFoxTest.m* script steers the RDA through it's entire beam pattern, and for every point saves the power measured from the network analyzer. After the test is completed, the test data is imported into *scanTestAnalysis.m*, which finds the location of maximum power and converts the corresponding phase shift to an angle (ϕ, θ). This test was performed for several DOA angles,

Table 5.3: RDA Calibration Results.

		S21 Phase Measurement(°)				Calibrated Phase Shift (°)			
Channel		1	2	3	4	1	2	3	4
	none	-11.3	-12.7	-2.8	-17.9	0	-1.4	8.5	-6.6
	1.4	-10.3	-11.7	-1.7	-17	1	-0.4	9.6	-5.7
	2.8	-9.2	-10.5	-0.5	-15.9	2.1	0.8	10.8	-4.6
	5.6	-6.7	-8.1	2	-13.6	4.6	3.2	13.3	-2.3
Bit	11.2	-2.4	-3.6	6.7	-8.6	8.9	7.7	18	2.7
	22.5	-11.8	10.3	20.2	5.1	-0.5	21.6	31.5	16.4
	45	34	32.7	42.4	29	45.3	44	53.7	40.3
	90	76.6	75.5	85	70.1	87.9	86.8	96.3	81.4
	180	174.3	172.9	-178.4	164.2	185.6	184.2	-167.1	175.5

and the resulting central angle scan error is reported in Table 5.4. Note that for ease of readability, the DOA angle shown is not the value reported by the DOA simulator, as it was in Table 5.1. This value is typically less than 1° different than the nominal value.

The results are worse than the same test performed on a single phase shifter board, see Table 5.1. This is to be expected, as the addition of more components and cabling leads to increased amplitude imbalance, which is not accounted for in the calibration. There are two trends observed from the results:

1. In general, larger scan angles result in larger scan errors. This is expected from the component level testing that was performed, as the higher phase shifter bits have greater insertion loss, which results in greater amplitude imbalance (see Figure 5.1).

2. The more time that passes between calibration and scanning, the greater the scanning error. This implies that there is an unexpected time variant aspect to the phase error. Tempera-

Table 5.4: Results of the system scanning tests.

	Simulated DOA		Scan Result		
Date	ZE (°)	AZ (°)	ZE (°)	AZ (°)	Central Angle Error (°)
8-Sep	45.0	300.0	46.6	293.1	5.2
9-Sep	15.0	0.0	12.7	35.5	8.7
9-Sep	15.0	60.0	7.9	67.5	7.2
10-Sep	15.0	120.0	27.4	115.7	12.5
11-Sep	30.0	300.0	26.9	292.2	4.9
12-Sep	30.0	240.0	27.9	227.0	6.6
13-Sep	30.0	120.0	24.2	140.6	10.9
13-Sep	30.0	180.0	39.1	155.6	16.4
14-Sep	30.0	60.0	26.9	66.2	4.3
15-Sep	30.0	0.0	35.3	0.0	5.3
16-Sep	45.0	0.0	48.8	16.5	12.6
16-Sep	45.0	60.0	23.8	63.7	21.3
17-Sep	45.0	120.0	27.7	167.6	32.0
24-Sep	0.0	0.0	22.0	101.0	22.0

ture variance is also possible, but unlikely as the tests occurred in a controlled laboratory environment.

Some tests showed relatively small scanning errors, within the range of expected values due to calibration and test errors. Other tests showed greater error, beyond the range of expected values. Based on the results of these tests, the following conclusions are drawn:

- Phase and amplitude errors within the phase shifter (DOA and RDA) are not responsible for the significant errors. As shown in the previous tests (see Table 5.1), the phase shifters contribute $< 5°$ of scan error.
- Static amplitude imbalances throughout the RDA are not the greatest source of error, as originally expected. These errors are not phase dependent, and should be consistent throughout all tests. Since some of the scan errors reported in Table 5.4 are not much greater than those reported in Table 5.1, it is unlikely that the static amplitude imbalances within the RDA and test setup are the cause of the large errors.
- The common trend in the large scan error tests are time. These tests occurred a significant duration (several days) after the calibration. This implies that an unknown time-varying phase and/or amplitude imbalance is causing the error.

It is recommended in Chapter 7 that an engineer development unit should be produced, which will reduce errors resulting from the test setup, and make it easier to identify the remaining unknown sources of error.

5.3.2 System Power Tests

Electronics Gain

In Section 2.6.1, it was shown that the expected power gain provided between the radio and antenna ports would be approximately 27 dB. Test were performed on the prototype system to verify this analysis, and the test results are shown in Table 5.5. The measured values were adjusted

by removing the test cable losses and board losses that would not be present in the final system. Before adjustment, the measured system gain was 20.6 dB. After adjusting for test cables and estimated PCB losses, the system gain was 26.8 dB. The gain was checked from 0.01W to 1W power output, and very little variation was seen (<0.25 dB).

The cable and test equipment losses could be accurately measured and calibrated out. However, there is significant uncertainty in the estimated board losses. A trace loss of 0.25 dB per inch was assumed based on loss measurements performed by Rogers Corporation [John Coonrad, 2011], and a coaxial to PCB connector loss of 0.1 dB was estimated as a conservative guess using engineering judgment. It is recommended in Chapter 7 that further testing is performed on a high-performance engineering model that utilizes high frequency materials and consolidates the entire RF chain between the radio and antenna ports onto a single PCB .

Table 5.5: System gain measurements.

	Component	Inline Components (for adjustment)	S21 (dB) Expected /Assumed	Measured	Adjusted
Test Setup	Adapter		-0.05		
	Trace		-0.25 / inch		
	RF-PCB (Connector)		-0.1		
	Cable 1 (N-SMA)	Adapter		-0.25	-0.20
	Cable 2 (SMA-MMCX)	2x Adapter		-1.08	-0.98
	Cable 3 (MMCX-MMCX)	2x Adapter		-0.67	-0.57
	Cable 4 (N-MMCX)	Adapter		-0.35	-0.30
	ZB4PD-232 (Splitter)	Cable 1, Cable 2, Adapter	-1.20	-2.72	-1.49
System Components	Phase Shifter	2x Adapter, Cable 2, Cable 3, 2.25in Trace, 2x RF-PCB	-6.50	-9.47	**-7.06**
	Switch Network	Cable 2, Cable 4, Adapter, 1.75in RF-PCB, 2x RF-PCB	-0.85	-2.97	**-1.00**
	Splitter Network	Cable 3,Cable 4, Adapter, 2.25in Trace, 2x RF-PCB	-1.07	-4.17	**-2.49**
Full System	System Receive	Cable 1, Cable 2, 2x Cable 3, Cable 4, 6.25in Trace, 6x RF-PCB, ZB4PD	-8.42	-15.43	**-9.16**
	System Transmit	Cable 1, Cable 2, 2x Cable 3, Cable 4, 6.25in Trace, 6x RF-PCB, ZB4PD	27.09	20.55	**26.82**

Chapter 6

Mission Design and Recommendations

This chapter provides information on spacecraft mission design for the RDA. This includes operations, requirements, and interfaces.

The intention of this chapter is to serve as a notional interface control document, and discuss the mission requirements that the RDA imposes. Note that much of the software discussed in this chapter was not implemented on the prototype. Figure 6.1 shows a simplified RDA block diagram.

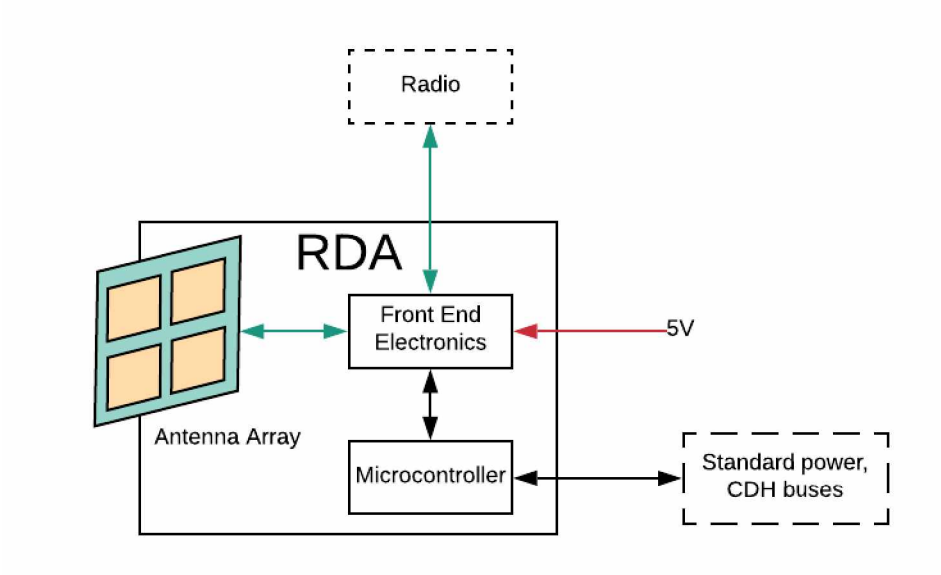


Figure 6.1: Simple RDA block diagram for system summary.

This chapter also discusses CubeSat Communications Platform (CCP), a CubeSat demonstration that is being designed to demonstrate the RDA along with other payloads.

6.1 Interface Control

6.1.1 Electrical Interfaces

The RDA subsystem requires three electrical interfaces: power, RF, and command/data.

Power

The RDA requires two power lines: 3.3 V and a switchable 5 V line. All components except the power amplifier are powered through the 3.3 V line. The power amplifiers require 5 V, and this rail should be switchable on the spacecraft side. This is for two reasons: (1) the idle power draw of the power amplifiers is 300 mA, 75 mA each multiplied by four amplifiers, and (2) the spacecraft should have the capability to terminate transmissions in the event of an RDA failure. Table 6.1 shows the voltage tolerance and maximum current draw for both supplies, based on components data sheets.

Table 6.1: RDA voltage rail tolerances and current draw.

Line	Voltage Tolerance (V)	Max Current Draw (mA)
3.3 V	0.1	20
5.0 V SW	0.1	2000

Radio Frequency

The RDA has a single external RF connection to radio. There are four internal RF connections between front-end electronics and the antenna elements. It is critical to correctly match each antenna element to the correct electronics channel. A method of identifying elements and channels relative to the spacecraft coordinates is required.

Command/Data

The command/data interface has not been specified. There are no high data rate requirements, but latency should be minimized because this affects the timing of retrodirective scans and TR switching.

6.1.2 Software Interfaces

This section describes the set of software commands and responses for communication with the RDA subsystem. Note that the command/data interface has not been specified, and this software was not implemented on the prototype. However, the code algorithms necessary to implement these commands were written in the process of testing the prototype.

Commands

Scan orders the RDA to perform a scan. After the scan the RDA will return to the prior steer angle, and return a scan report.

Set Angle orders the RDA to steer to a specified angle.

Set Phase orders the RDA to set a specific phase shift on each channel. This is useful for complicated beam patterns beyond basic steering.

Steer Scan orders the RDA to perform a scan. After the scan, the RDA will steer to the location of the strongest interrogator, and return a scan report.

TX orders the RDA to enter TX mode.

RX orders the RDA to enter RX mode.

Read Angle returns the current steer angle.

Read Phase returns the current phase states of each channel.

Read Status returns the status byte. See Figure 6.2.

Read Temp returns the temperature sensor readings.

Interrogator Report returns the power level and angle of the top three identified interrogators from the last scan.

Scan Report returns the most recent scan report in memory. The scan report is a matrix of all scan angles and received power values.

Packet returns the predefined telemetry packet. See Figure 6.3.

Status Byte

The status byte for the RDA is composed of three error state bits, a scan status flag, a high temperature flag (TEMP_H), a low temperature flag (TEMP_L), a TR state flag, and a retrodirectivity state (RETRO) flag. see Figure 6.2. The error state bits provide up to 7 possible error states, and one non-error state; these error states have not been specified yet. The scan flag is 1 while the RDA is performing a scan, and 0 otherwise. The temperature high flag or temperature low flag will be 1 when any of the temperature sensors read a value that exceeds the nominal range. The TR flag is 1 while in the transmit state, and 0 while in the receive state. The retrodirectivity state flag is 1 if the last steer scan successfully located an interrogator.

Figure 6.2: RDA status byte.

Telemetry packet

The standard telemetry packet for the RDA includes the current phase of each phase shifter, an interrogator report, the temperature sensor readings, and the status byte. See Figure 6.3.

6.1.3 Mechanical Interfaces

The mechanical mounting of the RDA has not been specified. However, there are several mounting considerations. First, no spacecraft components or structure can be placed in front of or beside the

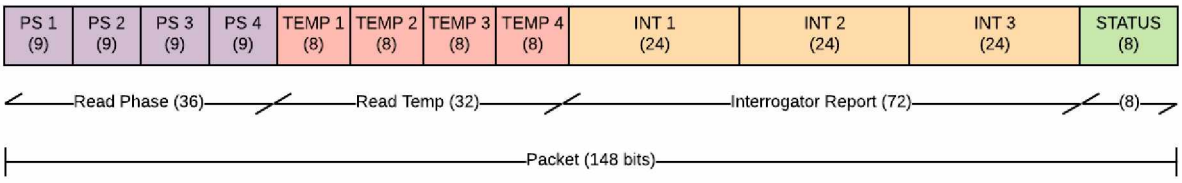


Figure 6.3: RDA telemetry packet structure.

antenna elements, without being accounted for in the antenna design. There must be an adequate conductive path between the antenna ground plane and the spacecraft chassis, utilizing several points of contact across the ground plane. The orientation of the antenna elements is critical, so the antenna elements must be identified according to the spacecraft coordinate axis and mounted accordingly. Figure 6.4 shows the numbering of the elements relative to the RDA coordinate axis.

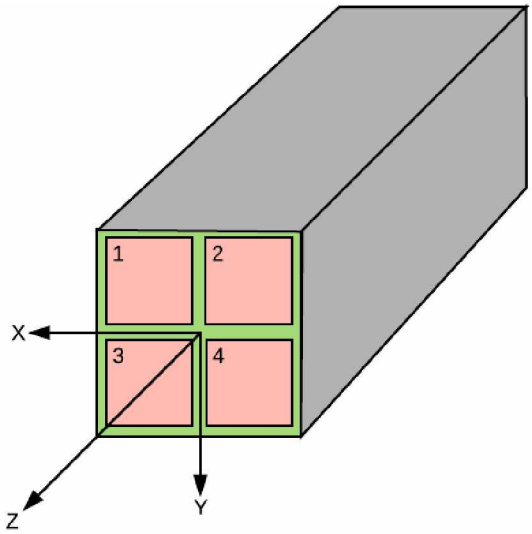


Figure 6.4: Element numbering on the RDA coordinate axis.

6.1.4 Thermal Interfaces

Thermal concerns related to the RDA were not explored in this thesis. The high power dissipation of the power amplifiers, 0.375 W each during idle and 1.25 W each during TX, requires thermal consideration and design.

6.2 Operations

6.2.1 RDA Modes

This section will explore modes of operation for the RDA.

Commanded Retrodirectivity

For the purpose of this thesis, commanded retrodirectivity is considered the standard operating method. In commanded retrodirectivity, the RDA performs scanning and beamforming when commanded to do so. This places the complexity of event coordination (TR switching and scan activities) on the spacecraft, rather than the RDA. Making the spacecraft the central event coordinator reduces the number of messages and time synchronization requirements between subsystems.

Autonomous Retrodirectivity

The RDA could be made more autonomous by placing the responsibility of choosing when to scan on the RDA. This places many of the event coordination responsibilities on the RDA. The RDA must determine when it should scan. This could be adaptive based on information provided from the spacecraft (see Scan Event Timing in Section 6.2.2), or on a predefined fixed interval. The spacecraft and RDA must also be synchronized on the timing of the scans and TR switching. This could be performed through synchronization and scheduling, or through real time notification.

Controlled Steering

As an alternative to retrodirectivity, the RDA can be used as a standard phased array. In this case, the spacecraft uses attitude and position knowledge to determine the necessary steer an-

gle. This requires some additional calculations on the spacecraft (see Steer Angle Determination in Section 6.2.2). By eliminating scanning, the spacecraft can continuously downlink without interruption.

Fixed Antenna

The RDA can be placed into a fixed beam pattern. Antenna pointing will be done mechanically, requiring spacecraft slew maneuvers. In this case, the front-end electronics are used only for TR switching and power amplification.

6.2.2 Operations Algorithms

Scan Event Timing

Equation (6.1) shows how to derive the necessary period between scans, T_{scan}. $ERROR_{max}$ is the maximum allowable steering error, and ω_{sat} is the angular velocity of the spacecraft relative to the ground station. $ERROR_{max}$ should be based on the available margin of the link, and the beam pattern of the RDA. As a rule of thumb, $ERROR_{max}$ should be selected to ensure that a 3 dB link margin is maintained at all times. For example, if the link is designed to maintain a 6 dB margin, then $0.5 * HPBW$ can be used for $ERROR_{max}$ to maintain a 3 dB link margin. Determining ω_{sat} is a problem that has been solved in the field of spacecraft attitude control (see [Chen et al., 2000]).

$$T_{scan} = \frac{ERROR_{max}}{\omega_{sat}} \tag{6.1}$$

TR Event Timing

The TR events must be coordinated between several spacecraft subsystems. How this occurs will be heavily dependent on the spacecraft design, but example sequences are presented here. The sequences described below assume real time command and response, as opposed to scheduled subsystem events.

Figure 6.5 shows the sequence for switching from TX to RX. First, the spacecraft Command and Data Handling (CDH) system commands the transceiver to stop transmissions. The transceiver replied with a confirmation when the transmission has stopped. CDH then commands the spacecraft Electric Power System (EPS) to power off the HPA power rail, and commands the RDA to switch to RX mode. The CDH waits the RDA switching time (currently estimated at 5 μs). At this point, the system is in the receive state.

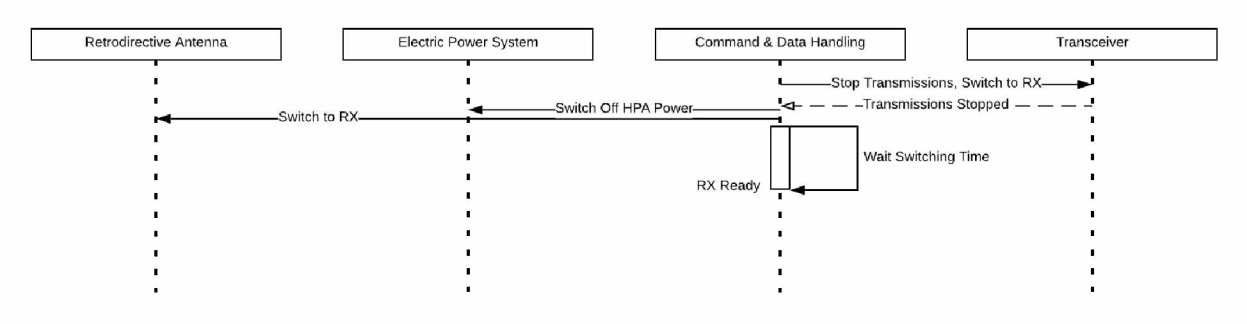


Figure 6.5: RX switching sequence.

Figure 6.6 shows the sequence for switching from RX to TX. First, CDH commands the EPS to power on the HPA power rail, and commands the RDA to switch to TX (these two activities may be done simultaneously). The CDH waits the RDA switching time (currently estimated at 5 μs), and then commands the transceiver to begin TX.

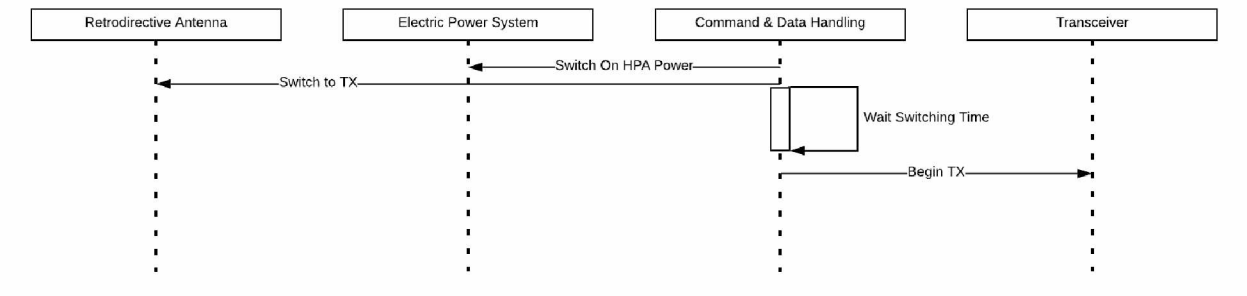


Figure 6.6: TX switching sequence.

Figure 6.7 shows the sequence for performing a scan. First, CDH commands the transceiver to transmit an interrogator request from the ground station, and then to stop transmissions. The

transceiver replies with a confirmation when the interrogator request has been sent and further transmissions have stopped. CDH then commands the RDA to perform the scan. The RDA switches to RX, performs the scan, switches back to TX, and then notifies CDH of the results. Lastly, CDH commands the transceiver to resume transmissions. Because the scan occurs quickly (less than 1 ms), it is unnecessary to have EPS power off the HPA power rail.

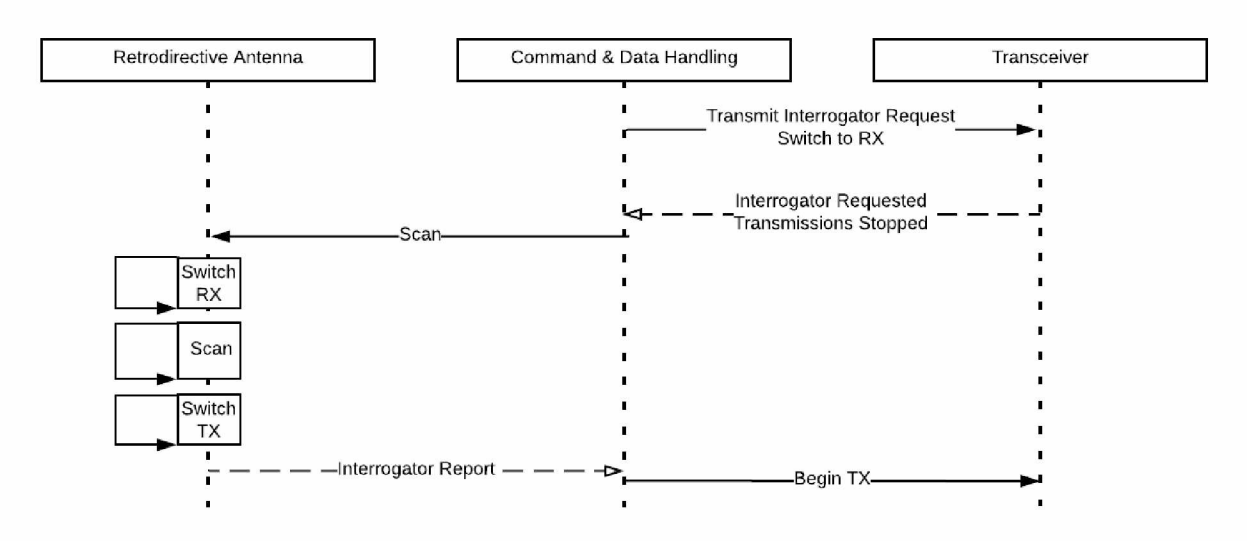


Figure 6.7: Scan coordination sequence.

Alternatively, the spacecraft could employ synchronized clocks on each subsystem, and use a scheduling system to coordinate TR events. This would require a timing analysis of all events in the sequence. The RDA handles internal switching time to ensure that high power from the HPA never appears on the RX path, so there is no concerns of exceeding RF power limits due to spacecraft switching errors. The estimated internal switching time for the RDA is 5 μs.

Steer Angle Determination

If the spacecraft is operating the RDA in the controlled steering mode, the spacecraft must be able to determine the necessary steer angle. Determining the angle to a target on the ground is a solved problem in the field on guidance and navigation. See Chen et al. [2000] for one example solution.

This is also necessary for missions that want to check the error of the RDA scanning results, such as the CCP mission.

Interrogator Timing

The RDA scan must occur while the interrogator is reaching the spacecraft, and the interrogator signal must be transmitted for a long enough duration to ensure that the scan finds it. This section will discuss how to ensure this occurs. This analysis assumed the controlled retrodirectivity mode of operation, as opposed to the autonomous retrodirectivity mode.

Figure 6.8 shows the timing diagram of the retrodirective sequence. The spacecraft first requests the interrogator from the ground station, and then commands the RDA to perform the scan.

To ensure that the RDA performs the scan while the interrogator is occurring, the spacecraft waits a time period (t_{setup}) between requesting the interrogator and commanding the RDA to scan. Equation (6.2) shows the derivation of t_{setup}. This is the signal propagation time (t_{prop}) to send the request to ground, plus the ground processing and response time (t_{ground}), plus the propagation time for the signal to get back to the spacecraft (t_{prop}), minus the scan command processing and response time (t_{comm}), minus the time required for the RDA to switch to RX (t_{SW}).

$$t_{setup} = t_{prop} + t_{ground} + t_{prop} - t_{comm} - t_{SW} \tag{6.2}$$

To ensure that the interrogator signal lasts long enough to complete the scan, the signal must be greater than or equal to the scan time (t_{scan}), plus the possible time errors in all previous actions. This is shown in Equation (6.3), where Δ denotes the possible error of a variable.

$$t_{int} \geq t_{scan} + \Delta t_{prop} + \Delta t_{SW} + \Delta t_{ground} + \Delta t_{comm} + \Delta t_{prop} \tag{6.3}$$

6.3 CubeSat Communications Platform

As part of this thesis, input was provided in the design of the CCP mission and spacecraft.

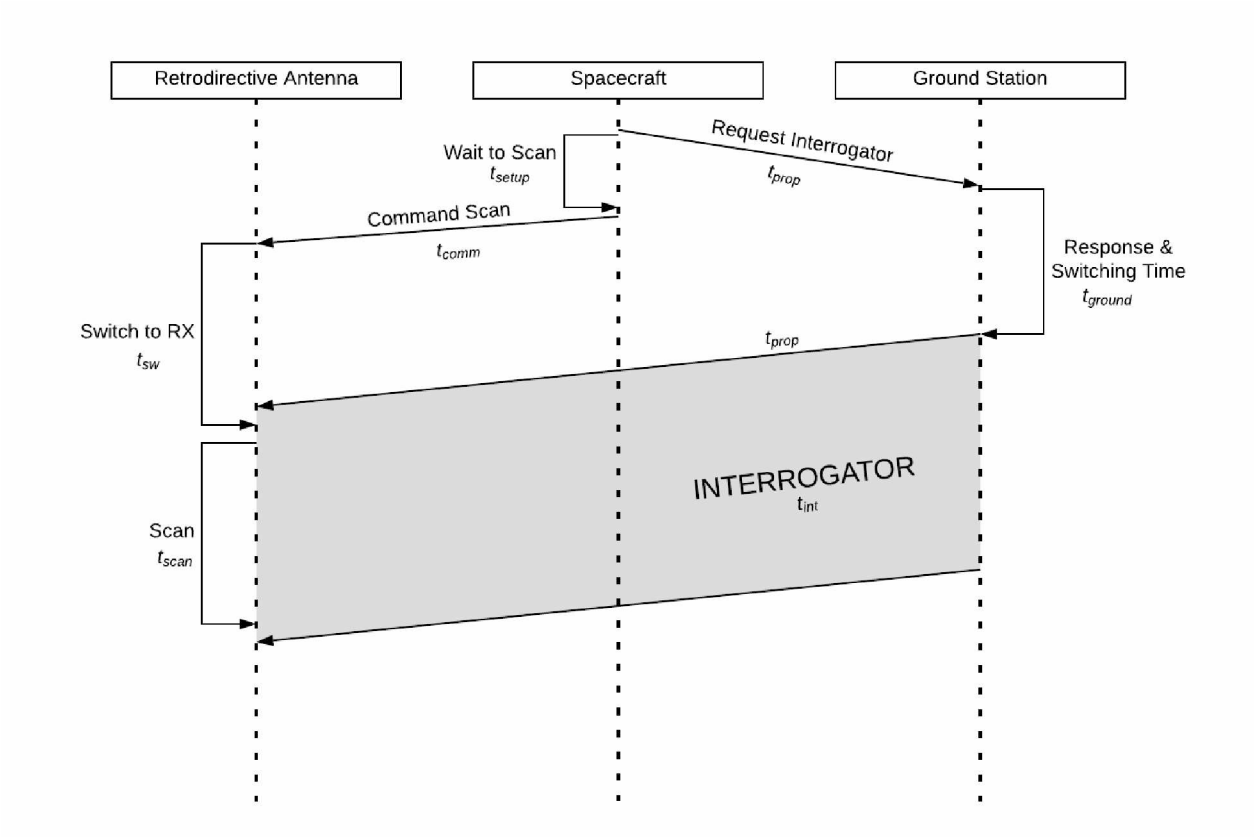


Figure 6.8: Interrogator timing diagram.

6.3.1 Measurement Flowdown

There are three key performance metrics: steering error, thermal load, and power efficiency. The thermal load and power efficiency are critical to understanding the RDA impact on spacecraft budgets, as well as understanding the in-space performance of the RDA with respect to temperature. Measuring beam pattern and gain in space is difficult; the steering error can be used to determine the antenna gain for a specific angle based on beam pattern measurements from the ground prior to flight. This is critical to understanding how the RDA performs compared to existing antenna solutions. Figure 6.9 shows the flowdown of how the key performance metrics are derived from observables.

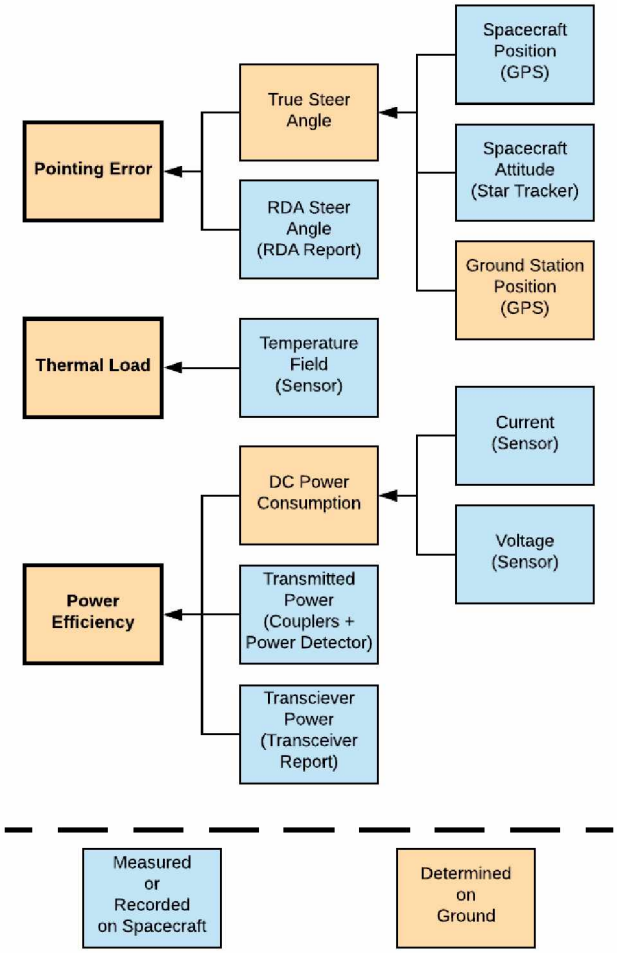


Figure 6.9: Flowdown of RDA experiment measurements.

Equation (6.4) shows the pointing error equation. Refer to Figure 6.10 to understand the central angle concept. The direction of the ground station (ϕ_{true}, θ_{true}) is projected as a point **P** onto a unit sphere, and the steering angle of the RDA (ϕ_{RDA}, θ_{RDA}) is projected as a point **Q** onto the sphere. The angle between these points in the **PQ** plane is the central angle, which is the absolute error of the RDA steering angle.

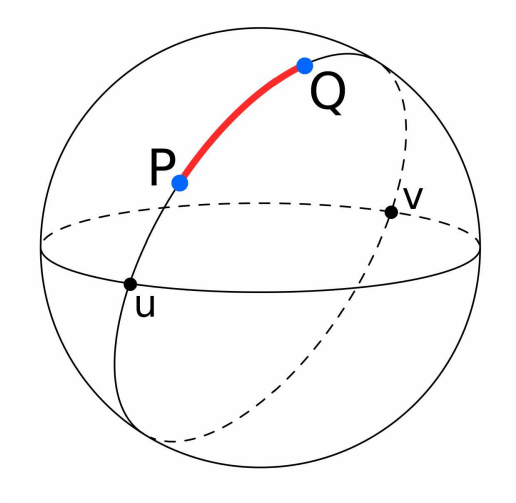


Figure 6.10: The central angle geometry.

$$Error = cos^{-1}[cos(\phi_{true}) \cdot cos(\phi_{RDA}) + sin(\phi_{true}) \cdot sin(\phi_{RDA}) \cdot cos(\theta_{true} - \theta_{RDA})] \tag{6.4}$$

The thermal load is derived from repeated temperature sensor measurements over time. The Power Added Efficiency (PAE) derivation is shown in Equation (6.5). P_{TX} is the power transmitted from the RDA and P_{in} is the power sent from the transceiver, both of which can be measured directly using couplers on the spacecraft. P_{DC} is the measured power draw of the RDA through the spacecraft current and voltage sensors.

$$PAE = \frac{P_{TX} - P_{in}}{P_{DC}} \tag{6.5}$$

6.3.2 Experiment Plan

Table 6.2 shows the experiment phases, the RDA and Attitude Determination and Control System (ADCS) modes for each phase, and the key measurements for each phase. The experiment phase of the CCP mission starts with a set of baseline measurements, which will (1) provide a baseline for fixed array performance to compare the future experiments against, and (2) provide a controlled set of passes during which to measure power efficiency and thermal load. During these baseline passes, the RDA will be fixed to broadside and the ADCS will be tracking the ground station.

After that, the experiment phases to determine pointing error will begin. This will consist of measuring the pointing error from both the commanded retrodirectivity and controlled beam-forming for three different operational scenarios: spacecraft pointed nadir, spacecraft pointed to Earth's limb, and spacecraft in a tumble scenario. Nadir and limb pointing have been selected because they are common earth observing satellite attitudes, and measure the extremes of the RDA visible range. Tumbling has been selected to test the RDA usability in the event of ADCS failure, which often results in random and uncontrolled tumble. The experiments have been ordered such that the safest experiments occur first, and experiments that may endanger spacecraft health occur last.

Table 6.2: RDA experiment phases.

	Baseline	Nadir Retrodirectivity	Nadir Controlled	Limb Retrodirectivity	Limb Controlled	Tumble Retrodirectivity	Tumble Controlled
RDA Mode	Antenna Fixed Broadside	Commanded Retrodirectivity	Controlled Steering	Commanded Retrodirectivity	Controlled Steering	Commanded Retrodirectivity	Controlled Steering
ADCS Mode	ADCS Ground Station Tracking	ADCS Nadir Pointed	ADCS Nadir Pointed	ADCS Limb Pointed	ADCS Limb Pointed	ADCS Tumble	ADCS Tumble
Key Measurements	Power Efficiency Thermal Load	Pointing Error	Pointing Error	Pointing Error	Pointing Error	Pointing Error	Pointing Error

Chapter 7

Future Work, Lessons Learned, and Conclusions

This chapter discusses future work that may be required to fully implement the RDA, or significantly improve the performance. It also discusses lessons learned to aid in future development.

7.1 Future Work Recommendations

This section outlines key improvements that could be made to the RDA. They are ordered in the order of perceived importance.

7.1.1 Engineering Model Development and Testing

The prototype model breaks the front end electronics into three separate boards, with coaxial cables connecting the RF paths. While this is adequate for proof of concept and functional testing, it does not allow for adequate performance testing of the system. As the critical next step of this development, an engineering unit must be developed for performance testing.

7.1.2 Antenna Design

The example antenna array presented in Section 3.2 is a simple linearly polarized microstrip with narrow bandwidth. This antenna array is the limiting factor in the RDA performance, and an improved antenna array design could provide improvements to antenna gain, beamwidth, and bandwidth.

As part of this thesis, methods of circular polarization were explored. Implementing circular polarization would increase the realized gain by 3 dB, by eliminating the 3 dB polarization loss. It was found that single feed point methods for circular polarization (i.e. modified edge and modified corner) methods were not suitable because the axial ratio was not maintained during beamforming, and the axial ratio bandwidth was narrow. NASA antenna engineers recommended dual feed and quadrature feed methods to achieve circular polarization. Kumar and Ray [2003] provides examples of these design methods.

Methods of improving bandwidth were also explored. The current bandwidth is adequate for a single transmitter, but will require redesign for each individual mission. The patch antennas presented in this thesis have a bandwidth (VSWR<1.5) of around 1%. To cover the entire downlink band (2200-2290 MHz), a bandwidth of 4% is required. To additionally cover the uplink band (2025-2110MHz), a bandwidth of 13% is required. NASA engineers recommended a stacked patch design, which can attain a bandwidth of 20% (VSWR<1.5); see Targonski et al. [1998].

Other antenna dielectrics should also be explored. Patch antennas on Rogers 6006 may meet the size requirements of the RDA. Rogers 6006 is better suited to patch antenna design than Rogers 6010, but the larger patch size may contribute to increased coupling between elements.

7.1.3 High Frequency Dielectric

The prototype unit used standard FR-4 for the front-end electronics board, which has a variable dielectric constant with respect to temperature, high IL, and poor impedance tolerance relative to high frequency materials. Coonrod [2011] provides more information about high frequency materials and the usage of FR-4. Also see the attached Roger Corporation memo in Appendix C.1, which discusses space grade high frequency dielectrics. Note that in the memo, the listed NASA website has been replaced by outgassing.nasa.gov.

7.1.4 Improved Phase Shifter Performance

The PE44820 phase shifter has an optional extra data bit that can be used to reduce phase error. For the prototype, the phase shifter was used in standard configuration, in which the optional data bit is not used. For low phase error applications, Peregrine Semiconductor provides calibration look-up tables for the phase shifter using the optional data bit. Utilizing the optional bit with these look-up tables is expected to reduce the phase error and improve scanning accuracy. For further information, see Peregrine Semiconductor application note AN45.

7.1.5 Power Amplifier Design

The SKY66294-11 power amplifier has several unknowns that are critical to performance, including phase imbalance. Testing showed a variation of 12° between devices, which results in significant beamforming error (on the order of 5°). Despite search efforts, other suitable COTS components were not found. A custom transistor level design could be optimized for phase balance. Using gallium nitride power amplifier technology on the final amplifier stage could greatly improve power efficiency, but would require using higher voltages.

7.1.6 Null Scanning

This prototype demonstration showed proof-of-concept with power scanning techniques. The same architecture can be used for null-scanning techniques, which leads to an improvement in scanning resolution. A reasonable next step is to implement null scanning, which only requires software modifications.

7.1.7 Smart Scanning Techniques

The scanning technique demonstrated required scanning through every possible phase range. Smart scanning techniques could be employed to reduce scanning time. For example, a coarse power with a 5° resolution could be used to locate the general location of the interrogator, and then a finer resolution scan could be performed on a smaller window of the scanning range.

7.1.8 Temperature Sensors

Temperature sensors could be added near the power amplifier. These temperature sensors would be for safety and calibration purposes. When a temperature limit is exceeded, the sensors can initiate a warning. The gain and efficiency of the power amplifiers is dependent on temperature, so the sensors could be used to improve knowledge on the instantaneous operating performance of the RDA.

7.1.9 Low Noise Amplifiers

The front end electronics add significant noise to the received signal. Low noise amplifiers could be added on the RX path of the TR switching network. This will significantly improve receive signal integrity of the received signal, but is not critical to the phased array functionality. It may have a negative effect on scanning error if the low noise amplifiers add further phase and amplitude imbalance.

7.1.10 Logic Shifters

The SKY66294-11 power amplifier and SKY13330 switch both use low voltage control signals. In the prototype, voltage dividers were used to step down the MSP430 control voltage from 3.3 V to 1.8 V. However, logic shifters could be added to eliminate voltage dividers and improve efficiency.

7.1.11 Switchable Power Rail

A 5 V power switch could be included so that the RDA can be powered down during the RX state. Most spacecraft can provide a switchable power rail and switch off the TX power on the spacecraft side.

7.2 Lessons Learned

This section describes lessons that were learned during the thesis. The purpose of sharing these lessons is to improve work quality and efficiency for future efforts.

7.2.1 RF Simulation and Optimization

Several RF simulation tools were explored during this thesis. It was found that freely available tools did not accurately simulate common antenna array phenomena, especially parasitic effects, active array coupling, and axial ratio during beamforming. Access to high performance RF simulation tools is critical to antenna array design.

7.2.2 RF PCB Layout

Various RF layout design techniques were explored during this thesis. The following paragraphs will outline the most effective methods, lessons learned, and tips provided by NASA engineers.

RF traces should be surrounded by ground vias in a process known as "fencing". There should be multiple rows of fencing, with each row staggered from the previous. Two rows of fencing is minimum, three rows is preferred. The spacing between vias should be approximately one eighth of the design wavelength.

Placing voids in the ground plane between channels further improves isolation. Each channel should be placed on an isolated "finger" or "room" of the ground plane.

Simulating RF PCB layouts using a high end RF simulator helps identify issues and verify performance prior to fabrication.

Extending the footprint of a pin from the underside of a pin makes fabrication and rework easier. It is especially recommended for QFN packages where the pins are no longer visible once the component has been placed. The extension should be at least 0.5 mm from the end of the package, and 1 mm is preferable.

7.2.3 Fabrication Processes

NASA engineers made several suggestions about how to improve fabrication processes and performance. These suggestions are summarized in the following paragraphs.

Many fabrication specialists do not use paste stencils, instead applying the paste to individual pins by hand for components down to the 0201 form factor. This is highly recommended for RF fabrication, because it provides good visual inspection and verification of the solder paste amount.

A high end microscope is critical. The individual beads in solder paste should be clearly visible through the microscope. Much of the fabrication should be done under this microscope. See Figure 7.1 for an example of what should be visible under the microscope. Visual inspection with a high end microscope following fabrication can quickly show errors that need to be corrected.

A fillet should be visible on every pin, there should not be a bulge of solder, nor should there be a visible gap between the component and solder pad. Having a tool small enough to push individual beads of solder paste is also critical. A Swiss oiler, modified to have a spade like tip, is a recommended tool.

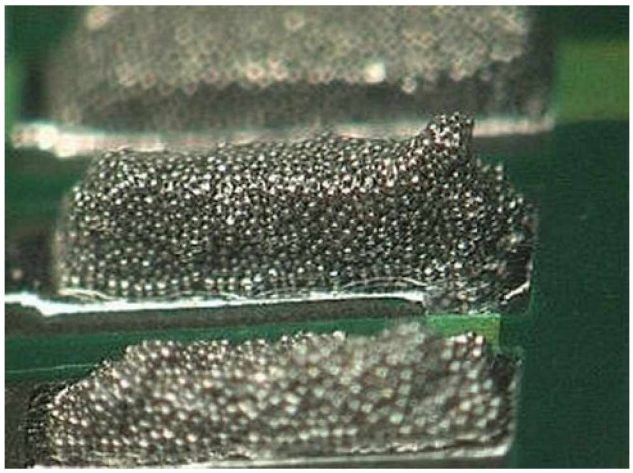


Figure 7.1: Example of PCB solder under a high end microscope.

7.3 Frequently Asked Questions

This section will discuss technical questions that were commonly asked, often during design reviews or conferences.

7.3.1 Software (IQ) Controlled System

Why use hardware phase shifters rather than software processing of the in-phase and quadrature (IQ) waveforms? Control of the IQ waveforms greatly improves the performance and versatility of the platform, while decreasing the hardware complexity. However, real time processing of the IQ waveforms requires significant computation time and power, more than is typically available on a CubeSat platform. Additionally, this requires a radio that has a port for each channel of the array, adding significant requirements on the radio and reducing scalability. In short, such a system is unlikely to be viable on a 1U CubeSat in the near future, and does not meet the design requirements of this project.

7.3.2 Monopulse Comparison Method

Could a monopulse phased array system be used for this application? The monopulse comparison method is another method of performing direction finding on an interrogating signal. This method takes the ratio of the sum and difference signals of sub-arrays (such as the ratio of the sum and difference of the top elements to the bottom elements, or the right elements to the left elements). This method requires significantly more complex hardware to acquire the sum and difference signal of the subarrays, which increases cost, decreases reliability, and adds volume. As a matter of opinion (an in-depth analysis was not performed), the monopulse comparison method seems unlikely to better meet the needs of a CubeSat mission than the power scanning method.

7.3.3 Time Delay Control

Would a time-delay beamforming method improve performance?. In this method, time delays are used to create phase shifts rather than a digital phase shifter. The primary advantage of the time delay method is less variation in phase shift as frequency changes. However, this requires significantly more volume for the time-delay circuitry. The phase shift change over the selected bandwidth was not significant, and the cost of increased volume is very significant on a CubeSat mission.

7.3.4 Is The RDA Worth It?

The RDA has been designed for a specific class of mission, and is not well suited to all missions. It is particularly well suited to small (1U to 3U) and Low Earth Orbit (LEO) missions. Other high gain antennas often require larger volume and deployables, which is a strenuous requirement on very small satellites. The benefits of beamforming are evident in LEO, where the spacecraft attitude relative to the communication target may be changing frequently. The phased array will require less frequent slew corrections to maintain a strong link, replacing those slew corrections with beamforming. The list below provides other potential missions in which the phased array may be useful.

Rover Relay A single satellite is providing relay communications for multiple planetary rovers. Utilizing a phased array can allow the spacecraft to communicate with several rovers over a large area. The phased array will provide a larger area of coverage compared to the fixed array, and higher data rates compared to wider beamwidth antennas.

Swarms and Mesh Many CubeSats are flying in a swarm formation with mesh network communications. Utilizing a phased array can allow these CubeSats to maintain strong signal strength with multiple swarm members. Again, the phased array will provide a larger area of coverage compared to the fixed array, and higher data rates compared to wider beamwidth antennas. Additionally, the phased array offers null forming capabilities if particular targets are too noisy.

Instrument Observation A satellite needs to downlink data, but a critical event is occurring and valuable science will be lost if the spacecraft is slewed to point the antenna. The phased array antenna may be able to close the link through beamforming, without having to slew the spacecraft.

The MarCO Case Study

The RDA has been compared to the MarCo reflectarray numerous times. MarCo is currently the only deep-space CubeSat mission to have flown; it provided communication coverage during the NASA InSight Mars landing. A UHF system was used to communicate to the InSight Lander, and a large X-band reflectarray was used to communicate back to earth. For that mission, the reflectarray is the better choice. A phased array of that scale would have been inefficient, costly, and the beamforming capabilities would not have been useful enough to outweigh the cost.

These same conditions hold true for many deep-space missions. For the majority of deep-space missions, the phased array is not a good choice as the primary antenna for earth communications. However, MarCo also used a medium gain antenna for short range communications between

spacecraft and as a back-up for the primary antenna. The medium gain antenna was a 2x2 patch array; the phased array would be a very suitable replacement for the MarCo medium gain antenna.

7.4 Conclusions

This thesis presented the design and prototype development of a phased array Retrodirective Antenna (RDA). The RDA was designed especially for 1U CubeSats in Low Earth Orbit (LEO).

The demonstrated performance of the RDA prototype provides an approximate 1 dB improvement in antenna gain, and a 10° improvement in beamwidth compared to existing COTS antennas (see [EnduroSat, 2018]). With some future work, significant improvements can be made to the RDA design to further increase this improvement. The front-end electronics and RDA algorithms add significant mission complexity. It is left to mission developers to determine if the performance improvements outweigh the added complexity.

Beyond the standard CubeSat mission, there are some applications where the RDA is simpler than alternatives, see the cases discussed in Section 7.3.4. There is also strong history of beamforming for electronics warfare applications, where it is desirable to place a null on an interrogating noise source, or locate and identify an unknown signal. The RDA also has more potential application at higher frequencies, where larger and more efficient arrays can be meet the size requirements of the CubeSat form factor.

Bibliography

Justin M Akagi, Alexis Zamora, Monte K Watanabe, and Wayne A Shiroma. A Self-Steering Array Using Power Detection and Phase Shifting. *IEEE MTT-S International Microwave Symposium Digest*, pages 1325–1328, 2008. doi: 10.1109/MWSYM.2008.4633307.

Prakash Bhartia, Inder Bahl, Ramesh Garg, and Apisake Ittipiboon. *Microstrip Antenna Design Handbook*. Artech House Publishers, 2001.

Xiaojiang Chen, Willem H. Steyn, and Yoshi Hashida. Ground-target tracking control of earth-pointing satellites. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000. ISBN 9781563479786.

Kiruthika Devaraj, Matt Ligon, Eric Blossom, Joseph Breu, Bryan Klofas, Kyle Colton, and Ryan Kingsbury. Planet High Speed Radio: Crossing Gbps from a 3U Cubesat. Technical report, Planet Labs, 2019.

EnduroSat. S-Band Patch Antenna Type II Data Sheet, 2018. URL https://www.endurosat.com/cubesat-store/all-cubesat-modules/s-band-patch-antenna/.

Erik Hammerstad and O. Jensen. Accurate Models for Microstrip Computer-Aided Design. *IEEE MTT-S International Microwave symposium Digest*, pages 407–409, 1980.

Mark Harris. *Near Earth Network (NEN) Users' Guide*. National Aeronautics and Space Administration, 2016.

Reece T. Iwami, Alexis Zamora, Tyler F. Chun, Monte K. Watanabe, and Wayne A. Shiroma. A retrodirective null-scanning array. In *IEEE MTT-S International Microwave Symposium Digest*, 2010. ISBN 9781424477326. doi: 10.1109/MWSYM.2010.5518000.

John Coonrad. Understanding When To Use FR-4 Or High Frequency Laminates. *Onboard Technology*, 9 2011.

Jonathan Klein, Joe Hawkins, and Denise Thorsen. Improving cubesat downlink capacity with active phased array antennas. In *IEEE Aerospace Conference Proceedings*, 2014. ISBN 9781479916221. doi: 10.1109/AERO.2014.6836238.

John D Kraus. The Corner-Reflector Antenna. *Proceedings of the I.R.E.*, pages 513–519, 1940.

Erik Kulu. Nanosatellite Database, 2019. URL http://www.nanosats.eu.

Garish Kumar and K.P. Ray. Broadband Microstrip Antennas. In *Compact and Broadband Microstrip Antennas*, pages 232–278. John Wiley & Sons, Inc., New York, USA, 2003. doi: 10.1002/0471221112.ch7. URL http://doi.wiley.com/10.1002/0471221112.ch7.

Michael Ligori, Laura Bradbury, Robert Spina, Robert E Zee, and Stephane Germain. GHGSat Constellation: The Future of Monitoring Greenhouse Gas Emissions. Technical report, GHGsat Inc., 2019.

Justin Long, Denise Thorsen, and Obadiah Kegege. Retrodirective Phased Array Antenna for CubeSats. In *IEEE Aerospace Conference Proceedings*, volume 2019-March, 2019. ISBN 9781538668542. doi: 10.1109/AERO.2019.8741562.

D. C. Nascimento and J. C. da S. Lacava. Probe-Fed Linearly-Polarized Electrically -Equivalent Microstrip Antennas on FR4 Substrates. *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, 13(1):55–66, 6 2014.

Hirobumi Saito, Koji Tanaka, Shinichi Nakasuka, and Seiko Shirasaka. The Development Status of the First Demonstration Satellite of Our Commercial Small Synthetic Aperture Radar Satellite Constellation. Technical report, JAXA, 2019.

Eugene D Sharp. Van Atta Reflector Array. Technical report, 1958.

Grant S Shiroma, Ryan Y Miyamoto, and Wayne A Shiroma. A Full-Duplex Dual-Frequency Self-Steering Array Using Phase Detection and Phase Shifting. *IEEE Transactions on Microwave Theory and Techniques*, 54(1), 2006. doi: 10.1109/TMTT.2005.860330.

Thomas A. Sielicki, Jon Hamkins, and Denise Thorsen. Variable Coded Modulation software simulation. In *IEEE Aerospace Conference Proceedings*, 2013. ISBN 9781467318112. doi: 10.1109/AERO.2013.6497354.

Warren Stutzman and Gary Thiele. *Antenna Theory and Design*. John Wiley & Sons, Inc, Hoboken, 3 edition, 2012. ISBN ISBN 978-0-470-57664-9.

Stephen Targonski, Rodney Waterhouse, and David Pozar. Design of wide-band aperture-stacked patch microstrip antennas. *IEEE Transactions on Antennas and Propagation*, 46(9):1245–1251, 1998. doi: 10.1109/8.719966.

The CubeSat Program. CubeSat Design Specification Rev. 13, 2014.

Appendices

Appendix A

Coordinate System Definition

An azimuth-zenith coordinate system is used throughout this thesis. Azimuth angles are denoted by either the letters AZ or the lowercase Greek letter theta θ. Zenith angles are denoted by either the letters ZE or the lowercase Greek letter phi ϕ. Zenith angles (as opposed to elevation angles) are used so that the antenna elements can lay in the X-Y plane, and the broadside direction can be denoted by (ϕ = 0). The elements exist in the first quadrant of the X-Y plane, with the feed of element 0 being at the origin. The coordinate system is graphically shown in Figure A.1.

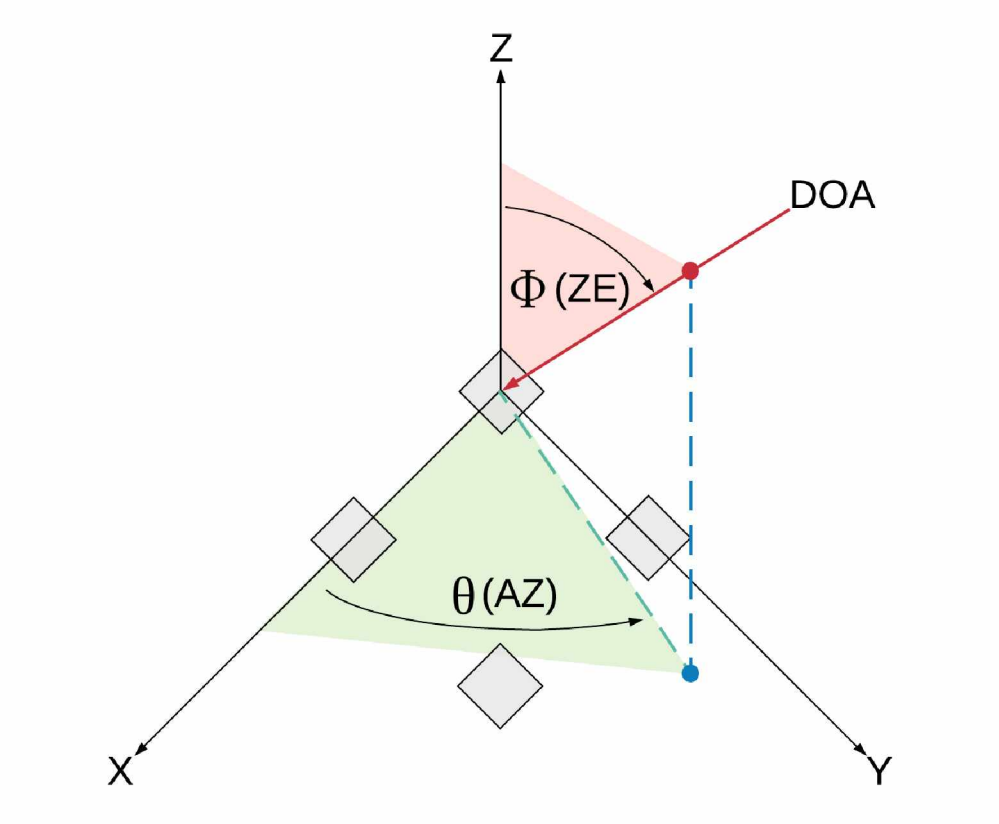


Figure A.1: The coordinate system used throughout this project.

Appendix B

Code

B.1 Antenna Element Design Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Design of patch antennas
% References:
% "Accurate Models for Microstip Computer-Aided Design", Hammerstad, 1980
% "Microstrip Antenna Design Handbook", Garg, 2001
% Justin Long, 2019
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;clear;close all;

%% Initial Inputs
f0 = 2200e6;            % Design Frequency (Hz)
eps_r = 10.7;           % dielectric constant
h = 1.27e-3;            % substrate height (meters)
t = 35e-6;              % copper thickness (meters)

% intermediate values
lambda0 = 3e8/f0;
k0 = 2*pi/lambda0;
lambda = 3e8/(f0*eps_r^0.5);
k = 2*pi/lambda;

%% Rectangular Microstrip Antenna (RMSA) Dimensions (Kumar Method)
% Choose a width. This is the equation for lambda/2
RMSA_W = 3e8/(2*f0*((eps_r+1)/2)^0.5); %meters

% Instead refer to the Hammerstad [1980] paper
% Hammerstad [1980] Equations for effective dielectric constant (1 to 9)
u = RMSA_W / h;
% Strip thickness correction for u (Hammerated [1980] equations 6 and 7)
del_u1 = t/pi*log(1+(4*exp(1)/(t*(coth((6.517*u)^0.5))^2)));
u1 = u + del_u1;
del_ur = 0.5 * (1+1/cosh((eps_r-1)^0.5)) * del_u1;
ur = u + del_ur;
% homogenous medium impedance (Hammerstad [1980] equations 1 and 2)
f = 6 + (2*pi-6)*exp(-(30.666/ur)^0.7528);
Z01 = (376.73/(2*pi)) * log(f/ur+(1+(2/ur)^2)^0.5);
Z02 = (376.73/(2*pi)) * log(f/u1+(1+(2/u1)^2)^0.5);
% effective dielectric (Hammerstad [1980] equations 3 to 5)
a = 1 + 1/49*log((ur^4+(ur/52)^2)/(ur^4+0.432)) + 1/18.7*log(1+(ur/18.1)^3);
b = 0.564*((eps_r-0.9)/(eps_r+0.3))^0.053;
eps_e0 = (eps_r+1)/2 + (eps_r-1)/2*(1+10/ur)^(-a*b);
% Apply strip thickness correction (Hammerstad [1980] equations 8 and 9)
Z0 = Z01 / (eps_e0)^0.5;
eps_e1 = eps_e0 * (Z01/Z02)^2;

% dispersion correction
fn = (f0*1e-9)*(h*1e3);
P1 = 0.27488 + u*(0.6315+0.525/(1+0.0157*fn)^20) - 0.065683*exp(-8.7513*u);
```

```
P2 = 0.33622*(1-exp(-0.03442*eps_r));
P3 = 0.0363*exp(-4.6*u)*(1-exp(-(fn/38.7)^4.97));
P4 = 1 + 2.751*(1-exp(-(eps_r/15.916)^8));
P = P1*P2*((0.1844+P3*P4)*fn)^1.5763;
eps_e = eps_r-(eps_r-eps_e1)/(1+P);

% Effective length (Kumar and Ray [2008] Equation 2.7) (lambda/2)
L_e = 3e8/(2*f0*(eps_e)^0.5);

% length extension (Kumar and Ray [2008] Equations B.42 through B.47)
zeta_1 = 0.434907 * (eps_e^0.81+0.26)/(eps_e^0.81-0.189) * ...
    (ur^0.8544+0.236)/(ur^0.8544+0.87);
zeta_2 = 1 + ur^0.371/(2.358*eps_r+1);
zeta_3 = 1 + (0.5274*atan(0.084*ur^(1.9413/zeta_2)))/(eps_e^0.9236);
zeta_4 = 1 + 0.0377*atan(0.067*ur^1.456)*(6-5*exp(0.036*(1-eps_r)));
zeta_5 = 1 - 0.218*exp(-7.5*ur);
delta_L = h * zeta_1 * zeta_2 * zeta_3 * zeta_5 / zeta_4;

% Associated Resonant Length (Kumar and Ray [2008] Equation 2.4)
RMSA_L = L_e-2*delta_L;

clear a b del_u1 del_ur delta_L eps_e1 eps_e2 f fp F L_e u u1 ur Z01 Z02...
   Z0 zeta_1 zeta_2 zeta_3 zeta_4 zeta_5 fn P P1 P2 P3 P4 u2 W2

%% Circular Microstrip Antenna (CSMA) Dimensions
ae = 0.08794/((f0*1e-9)*sqrt(eps_r));

syms a
eqn = ae == a*(1+(2*h)/(pi*a*eps_r)*
 ↪  (log(a/(2*h))+(1.41*eps_r+1.77)+h/a*(0.268*eps_r+1.65)))^0.5;
sola = solve(eqn,a);
CSMA_a = double(sola);

clear sola eqn ae a
%% Square Microstrip Antenna (SMSA) Dimensions
% Choose a starting width. This is the equation for lambda/2
SMSA_L = 3e8/(2*f0*((eps_r+1)/2)^0.5); %meters

% initialize the residual variable
res = 1;

while res>1e-9
    % Instead refer to the Hammerstad [1980] paper
    % Hammerstad [1980] Equations for effective dielectric constant (1 to 9)
    u = SMSA_L / h;
    % Strip thickness correction for u (Hammerated [1980] equations 6 and 7)
    del_u1 = t/pi*log(1+(4*exp(1)/(t*(coth((6.517*u)^0.5))^2)));
    u1 = u + del_u1;
    del_ur = 0.5 * (1+1/cosh((eps_r-1)^0.5)) * del_u1;
    ur = u + del_ur;
    % homogenous medium impedance (Hammerstad [1980] equations 1 and 2)
    f = 6 + (2*pi-6)*exp(-(30.666/ur)^0.7528);
    Z01 = (376.73/(2*pi)) * log(f/ur+(1+(2/ur)^2)^0.5);
    Z02 = (376.73/(2*pi)) * log(f/u1+(1+(2/u1)^2)^0.5);
```

```
    % effective dielectric (Hammerstad [1980] equations 3 to 5)
    a = 1 + 1/49*log((ur^4+(ur/52)^2)/(ur^4+0.432)) + 1/18.7*log(1+(ur/18.1)^3);
    b = 0.564*((eps_r-0.9)/(eps_r+0.3))^0.053;
    eps_e0 = (eps_r+1)/2 + (eps_r-1)/2*(1+10/ur)^(-a*b);
    % Apply strip thickness correction (Hammerstad [1980] equations 8 and 9)
    Z0 = Z01 / (eps_e0)^0.5;
    eps_e1 = eps_e0 * (Z01/Z02)^2;

    % dispersion correction
    fn = (f0*1e-9)*(h*1e3);
    W2 = SMSA_L + (t/pi)*(1+log(4/((t/h)^2+(1/pi)^2/(SMSA_L/t+1.1)^2)^0.5));
    u2 = (SMSA_L+(W2-SMSA_L)/eps_r)/h;
    P1 = 0.27488 + u2*(0.6315+0.525/(1+0.0157*fn)^20) - 0.065683*exp(-8.7513*u2);
    P2 = 0.33622*(1-exp(-0.03442*eps_r));
    P3 = 0.0363*exp(-4.6*u2)*(1-exp(-(fn/38.7)^4.97));
    P4 = 1 + 2.751*(1-exp(-(eps_r/15.916)^8));
    P = P1*P2*((0.1844+P3*P4)*fn)^1.5763;
    eps_e = eps_r-(eps_r-eps_e1)/(1+P);

    % Effective length (Kumar and Ray [2008] Equation 2.7) (lambda/2)
    L_e = 3e8/(2*f0*(eps_e)^0.5);

    % length extension (Kumar and Ray [2008] Equations B.42 through B.47)
    zeta_1 = 0.434907 * (eps_e^0.81+0.26)/(eps_e^0.81-0.189) * ...
        (ur^0.8544+0.236)/(ur^0.8544+0.87);
    zeta_2 = 1 + ur^0.371/(2.358*eps_r+1);
    zeta_3 = 1 + (0.5274*atan(0.084*ur^(1.9413/zeta_2)))/(eps_e^0.9236);
    zeta_4 = 1 + 0.0377*atan(0.067*ur^1.456)*(6-5*exp(0.036*(1-eps_r)));
    zeta_5 = 1 - 0.218*exp(-7.5*ur);
    delta_L = h * zeta_1 * zeta_2 * zeta_3 * zeta_5 / zeta_4;

    % Associated Resonant Length (Kumar and Ray [2008] Equation 2.4)
    SMSA_L2 = L_e-2*delta_L;
    res = abs(SMSA_L2 - SMSA_L);
    SMSA_L = SMSA_L2;
end

clear a b del_u1 del_ur delta_L eps_e1 eps_e2 f fp F L_e u u1 ur Z01 Z02...
   Z0 zeta_1 zeta_2 zeta_3 zeta_4 zeta_5 fn P P1 P2 P3 P4 u2 W2 SMSA_L2
```

B.2 Analytic Visualization Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% analyticalPattern.m
%
% Author: Justin Long
% Date: 11/10/2017
%
% This work was supported by a NASA Space Technology Research Fellowship
%
% This script imports a MATLAB antenna element and uses analytical
% equations to find the array factor, fixed beam pattern, and
% retrodirective array patterm.
%
```

```
% NOTE: This code is not meant to be run from beginning to end. Instead,
% the code should be run section-by-section, with the user plugging in
% their desired inputs.
%
% NOTE: The antenna element used by default in this code is not the RHCP
% antenna that was designed in this thesis. A element with a simpler beam
% pattern is designed in testElement.m.
%
% INPUTS:
% rows      Number of rows in array
% columns   Number of columns in array
% spacing   Spacing between elements in wavelengths
% freq      Design frequency of array
% steerZE   The zenith steering angle for the fixed array (degrees)
% steerAZ   The azimuth steering angle for the fixed array (degrees)
% grid      The angle resolution used for analysis
% filename  Filename of the imported antenna element
% AZcut     The azimuth cut angle for plots (degrees)
% ZEcut     The elevation cut angle for plots (degrees)
%
% OUTPUTS:
% fixedAF   The array factor when fixed to the steer angle
% fixedPat  The array pattern when fixed to the steer angle
% retroAF   The array factor when using retrodirectivity
% retroPat  The array pattern when using retrodirectivity
% Various plots of the above and the element pattern
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;clc;clf;close all;

%% Setup
rows = 2;               % number of rows (elements lying on X-axis)
columns = 2;            % number of columns (elements lying on Y-axis)
spacing = 0.5;          % element spacing in wavelengths
freq = 2.2e9;           % define the center frequency

steerZE = 0;            % zenith
steerAZ = 0;            % Azimuth
grid = 2;               % angle increment size

%% Import antenna element
% input: Set the filename of the saves antenna element
filename = 'PatchElement.mat';

% Load the specified antenna element
load(filename, 'patchElement');

% Tilt antenna to match defined coordinate system
patchElement.Tilt = 90;
patchElement.TiltAxis = [0 0 1];

% get directivity vector
sigAzimuth = -180:grid:180;
sigZenith = 90:grid:270;
```

```
[D] = pattern(patchElement, freq, sigAzimuth, sigZenith);

% rotate the elevation angles to zenith angles
sigZenith = sigZenith-90;

%% Array pattern for fixed steering angle
% Sweep through each zenith and azimuth angle of the array
for k = 1:length(sigAzimuth)
    for c = 1:length(sigZenith)
        fixedAF(c,k) = arrayFactor(steerAZ,steerZE,sigAzimuth(k),...
            sigZenith(c),spacing,rows,columns);
    end
end

% convert the AF to a power dB and add to element pattern
fixedPat = db(fixedAF.^0.5)+D;

%% Array pattern for retrodirective steering
% Sweep through each zenith and azimuth angle of the array
for k = 1:length(sigAzimuth)
    for c = 1:length(sigZenith)
        retroAF(c,k) = arrayFactor(sigAzimuth(k),sigZenith(c),sigAzimuth(k),...
            sigZenith(c),spacing,rows,columns);
    end
end

% convert the AF to a power dB and add to element pattern
retroPat = db(retroAF.^0.5)+D;

%% 3D Element Plots
% Element Pattern
figure;
contourf(sigAzimuth,sigZenith,D,30)
xlabel('Azimuth (degrees)')
ylabel('Zenith (degrees)')
title('Element Pattern');
colorbar;

% 3D surface plot of the element pattern
figure;
patternCustom(D',sigZenith,sigAzimuth,'CoordinateSystem',...
    'rectangular');

% Spherical Plot of the fixed Array Pattern
figure;
patternCustom(D',sigZenith,sigAzimuth);

%% 3D Fixed Array Plots
% Fixed Array AF
figure;
contourf(sigAzimuth,sigZenith,fixedAF)
xlabel('Azimuth (degrees)')
ylabel('Zenith (degrees)')
title(sprintf(...
```

```
121     'Fixed Array Factor when Steered to %.1f Zenith and %.1f Azimuth'...
122     ,steerZE,steerAZ));
123 colorbar;
124
125 % Fixed Array Pattern
126 figure;
127 contourf(sigAzimuth,sigZenith,fixedPat,30)
128 xlabel('Azimuth (degrees)')
129 ylabel('Zenith (degrees)')
130 title(sprintf(...
131     'Fixed Array Pattern when Steered to %.1f Zenith and %.1f Azimuth'...
132     ,steerZE,steerAZ));
133 colorbar;
134
135 % 3D surface plot of the fixed array pattern
136 figure;
137 patternCustom(fixedPat',sigZenith,sigAzimuth,'CoordinateSystem',...
138     'rectangular');
139
140 % Spherical Plot of the fixed Array Pattern
141 figure;
142 patternCustom(retroPat',sigZenith,sigAzimuth);
143
144 %% 3D Retrodirective Array Plots
145 % Retrodirective Array AF
146 figure;
147 contourf(sigAzimuth,sigZenith,retroAF)
148 xlabel('Azimuth (degrees)')
149 ylabel('Zenith (degrees)')
150 title(sprintf(...
151     'Retrodirective Array Factor when Steered to %.1f Zenith and %.1f Azimuth'...
152     ,steerZE,steerAZ));
153 colorbar;
154
155 % Retrodirective Array Pattern
156 figure;
157 contourf(sigAzimuth,sigZenith,retroPat,30)
158 xlabel('Azimuth (degrees)')
159 ylabel('Zenith (degrees)')
160 title(sprintf(...
161     'Retrodirective Array Pattern when Steered to %.1f Zenith and %.1f Azimuth'...
162     ,steerZE,steerAZ));
163 colorbar;
164
165 % 3D surface plot of the retrodirective array pattern
166 figure;
167 patternCustom(retroPat',sigZenith,sigAzimuth,'CoordinateSystem',...
168     'rectangular');
169
170 % Spherical Plot of the retrodirective Array Pattern
171 figure;
172 patternCustom(retroPat',sigZenith,sigAzimuth);
173
174 %% Zenith cut angles
```

```
% input: choose cut angle
ZEcut = 0;          % Angle to take the cut

% Rectangular

% Element
figure;
patternCustom(D',sigZenith,sigAzimuth,'CoordinateSystem',...
    'rectangular','Slice','theta','SliceValue',ZEcut);
xlabel('Azimuth (degrees)')
ylabel('Element Directivity (dBi)')
title('Element Directivity (dBi)');

% Fixed AF
figure;
patternCustom(fixedAF',sigZenith,sigAzimuth,'CoordinateSystem',...
    'rectangular','Slice','theta','SliceValue',ZEcut);
xlabel('Azimuth (degrees)')
ylabel('Array Factor')
title(sprintf(...
    'Fixed Array Factor, Zenith Cut %.1f, Steered to %.1f ZE %.1f AZ',...
    ZEcut,steerZE,steerAZ));

% Fixed Array Pattern
figure;
patternCustom(fixedPat',sigZenith,sigAzimuth,'CoordinateSystem',...
    'rectangular','Slice','theta','SliceValue',ZEcut);
xlabel('Azimuth (degrees)')
ylabel('Array Pattern (dB)')
title(sprintf(...
    'Fixed Array Pattern, Zenith Cut %.1f, Steered to %.1f ZE %.1f AZ',...
    ZEcut,steerZE,steerAZ));

% Retro AF
figure;
patternCustom(retroAF',sigZenith,sigAzimuth,'CoordinateSystem',...
    'rectangular','Slice','theta','SliceValue',ZEcut);
xlabel('Azimuth (degrees)')
ylabel('Array Factor')
title(sprintf(...
    'Retrodirective Array Factor, Zenith Cut %.1f, Steered to %.1f ZE %.1f AZ',...
    ZEcut,steerZE,steerAZ));

% Fixed Array Pattern
figure;
patternCustom(retroPat',sigZenith,sigAzimuth,'CoordinateSystem',...
    'rectangular','Slice','theta','SliceValue',ZEcut);
xlabel('Azimuth (degrees)')
ylabel('Array Pattern (dB)')
title(sprintf(...
    'Retrodirective Array Pattern, Zenith Cut %.1f, Steered to %.1f ZE %.1f AZ',...
    ZEcut,steerZE,steerAZ));

% Polar
```

```
229
230 % Element
231 figure;
232 patternCustom(D',sigZenith,sigAzimuth,'CoordinateSystem',...
233     'polar','Slice','theta','SliceValue',ZEcut);
234
235 % Fixed Array Pattern
236 figure;
237 patternCustom(fixedPat',sigZenith,sigAzimuth,'CoordinateSystem',...
238     'polar','Slice','theta','SliceValue',ZEcut);
239
240 % Retro Array Pattern
241 figure;
242 patternCustom(retroPat',sigZenith,sigAzimuth,'CoordinateSystem',...
243     'polar','Slice','theta','SliceValue',ZEcut);
244
245 %% Azimuth cut angles
246 AZcut = 0;          % Angle to take the cut
247
248 % Rectangular
249
250 % Element Directivity
251 figure;
252 patternCustom(D',sigZenith,sigAzimuth,'CoordinateSystem',...
253     'rectangular','Slice','phi','SliceValue',AZcut);
254 xlabel('Zenith (degrees)')
255 ylabel('Element Directivity (dBi)')
256 title(sprintf(...
257     'Element Directivity, Azimuth Cut %.1f, Steered to %.1f ZE %.1f AZ',...
258     AZcut,steerZE,steerAZ));
259
260 % Fixed AF
261 figure;
262 patternCustom(fixedAF',sigZenith,sigAzimuth,'CoordinateSystem',...
263     'rectangular','Slice','phi','SliceValue',AZcut);
264 xlabel('Zenith (degrees)')
265 ylabel('Array Factor')
266 title(sprintf(...
267     'Fixed Array Factor, Azimuth Cut %.1f, Steered to %.1f ZE %.1f AZ',...
268     AZcut,steerZE,steerAZ));
269
270 % Fixed Array Pattern
271 figure;
272 patternCustom(fixedPat',sigZenith,sigAzimuth,'CoordinateSystem',...
273     'rectangular','Slice','phi','SliceValue',AZcut);
274 xlabel('Zenith (degrees)')
275 ylabel('Array Pattern (dB)')
276 title(sprintf(...
277     'Fixed Array Pattern, Azimuth Cut %.1f, Steered to %.1f ZE %.1f AZ',...
278     AZcut,steerZE,steerAZ));
279
280 % retro AF
281 figure;
282 patternCustom(retroAF',sigZenith,sigAzimuth,'CoordinateSystem',...
```

```
    'rectangular','Slice','phi','SliceValue',AZcut);
xlabel('Zenith (degrees)')
ylabel('Array Pattern (dBi)')
title(sprintf(...
    'Retrodirective Array Factor, Azimuth Cut %.1f, Steered to %.1f ZE %.1f AZ',...
    AZcut,steerZE,steerAZ));

% Fixed Array Pattern
figure;
patternCustom(retroPat',sigZenith,sigAzimuth,'CoordinateSystem',...
    'rectangular','Slice','phi','SliceValue',AZcut);
xlabel('Zenith (degrees)')
ylabel('Array Pattern (dBi)')
title(sprintf(...
    'Retrodirective Array Pattern, Azimuth Cut %.1f, Steered to %.1f ZE %.1f AZ',...
    AZcut,steerZE,steerAZ));

% Polar

% Element Pattern
figure;
patternCustom(D',sigZenith,sigAzimuth,'CoordinateSystem',...
    'polar','Slice','phi','SliceValue',AZcut);

% Fixed Array Pattern
figure;
patternCustom(fixedPat',sigZenith,sigAzimuth,'CoordinateSystem',...
    'polar','Slice','phi','SliceValue',AZcut);

% Retro Array Pattern
figure;
patternCustom(retroPat',sigZenith,sigAzimuth,'CoordinateSystem',...
    'polar','Slice','phi','SliceValue',AZcut);
```

B.2.1 Array Factor Algorithm

```
function [arrayFactor] = arrayFactor( steerAZ, steerZE, sigAZ, sigZE, spacing, N, M )
%arrayFactor calculates the array factor of a planar array for a given
%steering direction and signal direction.
%   steerAZ is the steered azimuth angle in degrees
%   steerZE is the steered Zenith angle in degrees
%   sigAZ is the signal azimuth angle in degrees
%   sigZE is the signal zenith angle in degrees
%   spacing is the spacing between elements in wavelengths
%   N is the number of rows
%   M is the number of columns
%
% Author: Justin Long
% Date: 11/10/2017
%
% This work was supported by a NASA Space Technology Research Fellowship

sigAZ = degtorad(sigAZ);
```

```
sigZE = degtorad(sigZE);
steerZE = degtorad(steerZE);
steerAZ = degtorad(steerAZ);

n = 0:1:N-1;
m = 0:1:M-1;

steerFactor = -2*pi*spacing*(repmat([sin(steerZE)*cos(steerAZ).*n'], ...
    [1,N])+repmat([sin(steerZE)*sin(steerAZ).*m],[M,1]));
sigFactor = 2*pi*spacing*(repmat([sin(sigZE)*cos(sigAZ).*n'], ...
    [1,N])+repmat([sin(sigZE)*sin(sigAZ).*m],[M,1]));

arrayFactor = exp(1i*steerFactor).*exp(1i*sigFactor);
arrayFactor = abs(sum(sum(arrayFactor)));
```

B.2.2 Notional Antenna Design

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% testElement.m
%
% Author: Justin Long
% Date: 11/5/2017
%
% This script shows the process of designing and modeling the
% theoretical 2.2 GHz array.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;clear;clf;close all;

%% Setup Analysis Parameters
centerFrequency = 2.2e9;
minFrequency = 2e9;
maxFrequency = 2.5e9;
vp = physconst('lightspeed');
lambda = vp/centerFrequency;
numRows = 2;
numCols = 2;

%% Initial Patch Element Design
patchElement = patchMicrostrip;
patchElement.Length = 0.49*lambda;
patchElement.Width = 1.5*patchElement.Length;
patchElement.GroundPlaneLength = lambda;
patchElement.GroundPlaneWidth = lambda;
patchElement.Height = 0.01*lambda;
patchElement.FeedOffset = [patchElement.Length/4 0];
patchElement.Tilt = 90;
patchElement.TiltAxis = [0 1 0];

%% Uncomment to find act_resonance
% numFreqs = 21;
% freqSweep = unique([linspace(minFrequency,maxFrequency,numFreqs) centerFrequency]);
% impedance(patchElement,freqSweep);
```

```
%% Scale the patch for resonance
act_resonance = 2.13e9; % Check the previous plot for this.
lambda_act = vp/act_resonance;
scale = lambda/lambda_act;
patchElement.Length = scale*patchElement.Length;

%% Create an Array
fc2 = 2.2e9;
lambda_fc2 = vp/fc2;
PatchArray = phased.URA;
PatchArray.Element = patchElement;
PatchArray.Size = [numRows numCols];
PatchArray.ElementSpacing = [0.5*lambda_fc2 0.5*lambda_fc2];

%% Plot the array beam pattern
az = -180:1:180;
el = -90:1:90;
patchArrayPattern = figure;
pattern(PatchArray,centerFrequency,az,el);

%% 2D plot of directivity
[Dpatch_az_zero,~,elp] = pattern(PatchArray,centerFrequency,0,el);
[Dpatch_el_zero,azp] =  pattern(PatchArray,centerFrequency,az,0);

%% Azimuth Pattern
elPattern = figure;
plot(elp,Dpatch_az_zero,'LineWidth',1.5)
axis([min(elp) max(elp) -40 30])
grid on
xlabel('Elevation (deg.)')
ylabel('Directivity (dBi)')
title('Array Directivity Variation-Azimuth = 0 deg.')
legend('Patch Antenna','Location','best')

%% Azimuth Pattern in Polar
patternElevation(PatchArray,centerFrequency);

%% Elevation Pattern
azPattern = figure;
plot(azp,Dpatch_el_zero,'LineWidth',1.5)
axis([min(azp) max(azp) -40 30])
grid on
xlabel('Azimuth (deg.)')
ylabel('Directivity (dBi)')
title('Array Directivity Variation-Elevation  = 0 deg.')
legend('Patch Antenna','Location','best')

%% Elevation Pattern in polar
patternAzimuth(PatchArray,centerFrequency);
```

B.3 Component Scanning Test Code

B.4 System Scanning Test Code

B.4.1 Test Execution

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fieldFoxTest.m
%
% Author: Justin Long
% Updated: 9/4/2019
%
% Runs the automated test script for the system scanning test of
% of the RDA. Uses the keysight field fox.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Cleanup
clc;clear;clf;close all;instrreset;

%% Setup the VISA connection to fieldfox
ff_address = 'TCPIP0::192.168.0.1::inst0::INSTR';
fieldfox = visa('agilent',ff_address);
fieldfox.InputBufferSize = 10e6;
fieldfox.ByteOrder = 'littleEndian';
fopen(fieldfox);
clrdevice(fieldfox);
IDNString = query(fieldfox,'*IDN?');
fprintf('Connected to: %s\n',IDNString);

%% Setup the serial connection to the board
DOA = serial('COM5');
RDA = serial('COM4');
fopen(DOA);
fopen(RDA);

%% Setup the field fox for measurement
fprintf(fieldfox, 'SYST:PRES');                 % Preset the device
fprintf(fieldfox, 'INST:SEL "NA"');             % Go to NA mode
fprintf(fieldfox, 'CALC:PAR1:DEF S21');         % Set measurement to S21
fprintf(fieldfox, 'CALC:FORM MLOG');            % Set display to log mag
fprintf(fieldfox, 'SENS:FREQ:STAR 2.1e9');      % Set start freq
fprintf(fieldfox, 'SENS:FREQ:STOP 2.3e9');      % Set stop freq
fprintf(fieldfox, 'CALC:MARK1 NORM');           % Create a marker
fprintf(fieldfox, 'CALC:SEL:MARK:X 2.2e9');     % Set the marker location
GPIB_Wait(fieldfox);

%% Setup the RDA for testing
reply = UART_SendWait4Reply(DOA, sprintf('resetPhase'));         % Set board to RX mode
reply = UART_SendWait4Reply(RDA, sprintf('resetPhase'));        % Set board to external
 ↪ port
reply = UART_SendWait4Reply(RDA, sprintf('SW RX'));         % Set board to RX mode

%% Choose DOA
for ZE = 15
```

```
48      for AZ = 0:60:300
49
50  %% Steer Angle Determination
51  % Uncaibrated Phase table
52  UCbits = [0 1.4 2.8 5.6 11.2 22.5 45 90 180];
53  phaseUC = [UCbits(1) UCbits(2) UCbits(1)+UCbits(2)];
54  phaseUC = [phaseUC UCbits(3) UCbits(3)+phaseUC];
55  phaseUC = [phaseUC UCbits(4) UCbits(4)+phaseUC];
56  phaseUC = [phaseUC UCbits(5) UCbits(5)+phaseUC];
57  phaseUC = [phaseUC UCbits(6) UCbits(6)+phaseUC];
58  phaseUC = [phaseUC UCbits(7) UCbits(7)+phaseUC];
59  phaseUC = [phaseUC UCbits(8) UCbits(8)+phaseUC];
60  phaseUC = [phaseUC UCbits(9) UCbits(9)+phaseUC];
61
62  % setup the true phase ranges
63  TXbits4 = [-1.2 -0.5 1.3 4.6 10.4 21.8 44.3 88.9 183.4];
64  TXbits3 = [2.3 8.5 3.0 4.8 8.1 14.2 25.4 48.7 93.2 187.1];
65  TXbits2 = [1.4 2.2 4.0 7.0 12.9 24.4 47.4 92.1 186.7];
66  TXbits1 = [0 0.7 2.6 5.8 11.8 23.3 46.6 90.8 183.2];
67
68  phaseTX4 = [TXbits4(1) TXbits4(2) TXbits4(1)+TXbits4(2)];
69  phaseTX4 = [phaseTX4 TXbits4(3) TXbits4(3)+phaseTX4];
70  phaseTX4 = [phaseTX4 TXbits4(4) TXbits4(4)+phaseTX4];
71  phaseTX4 = [phaseTX4 TXbits4(5) TXbits4(5)+phaseTX4];
72  phaseTX4 = [phaseTX4 TXbits4(6) TXbits4(6)+phaseTX4];
73  phaseTX4 = [phaseTX4 TXbits4(7) TXbits4(7)+phaseTX4];
74  phaseTX4 = [phaseTX4 TXbits4(8) TXbits4(8)+phaseTX4];
75  phaseTX4 = [phaseTX4 TXbits4(9) TXbits3(9)+phaseTX4];
76  phaseTX3 = [TXbits3(1) TXbits3(2) TXbits3(1)+TXbits3(2)];
77  phaseTX3 = [phaseTX3 TXbits3(3) TXbits3(3)+phaseTX3];
78  phaseTX3 = [phaseTX3 TXbits3(4) TXbits3(4)+phaseTX3];
79  phaseTX3 = [phaseTX3 TXbits3(5) TXbits3(5)+phaseTX3];
80  phaseTX3 = [phaseTX3 TXbits3(6) TXbits3(6)+phaseTX3];
81  phaseTX3 = [phaseTX3 TXbits3(7) TXbits3(7)+phaseTX3];
82  phaseTX3 = [phaseTX3 TXbits3(8) TXbits3(8)+phaseTX3];
83  phaseTX3 = [phaseTX3 TXbits3(9) TXbits3(9)+phaseTX3];
84  phaseTX2 = [TXbits2(1) TXbits2(2) TXbits2(1)+TXbits2(2)];
85  phaseTX2 = [phaseTX2 TXbits2(3) TXbits2(3)+phaseTX2];
86  phaseTX2 = [phaseTX2 TXbits2(4) TXbits2(4)+phaseTX2];
87  phaseTX2 = [phaseTX2 TXbits2(5) TXbits2(5)+phaseTX2];
88  phaseTX2 = [phaseTX2 TXbits2(6) TXbits2(6)+phaseTX2];
89  phaseTX2 = [phaseTX2 TXbits2(7) TXbits2(7)+phaseTX2];
90  phaseTX2 = [phaseTX2 TXbits2(8) TXbits2(8)+phaseTX2];
91  phaseTX2 = [phaseTX2 TXbits2(9) TXbits2(9)+phaseTX2];
92  phaseTX1 = [TXbits1(1) TXbits1(2) TXbits1(1)+TXbits1(2)];
93  phaseTX1 = [phaseTX1 TXbits1(3) TXbits1(3)+phaseTX1];
94  phaseTX1 = [phaseTX1 TXbits1(4) TXbits1(4)+phaseTX1];
95  phaseTX1 = [phaseTX1 TXbits1(5) TXbits1(5)+phaseTX1];
96  phaseTX1 = [phaseTX1 TXbits1(6) TXbits1(6)+phaseTX1];
97  phaseTX1 = [phaseTX1 TXbits1(7) TXbits1(7)+phaseTX1];
98  phaseTX1 = [phaseTX1 TXbits1(8) TXbits1(8)+phaseTX1];
99  phaseTX1 = [phaseTX1 TXbits1(9) TXbits1(9)+phaseTX1];
100
101 RXbits4 = [-6.6 -5.7 -4.6 -2.3 2.7 16.4 40.3 81.4 175.5];
```

```
RXbits3 = [8.5 9.6 10.8 13.3 18.0 31.5 53.7 96.3 -167.1];
RXbits2 = [-1.4 -0.4 0.8 3.2 7.7 21.6 44.0 86.8 184.2];
RXbits1 = [0.0 1.0 2.1 4.6 8.9 -0.5 45.3 87.9 185.6];

phaseRX4 = [RXbits4(1) RXbits4(2) RXbits4(1)+RXbits4(2)];
phaseRX4 = [phaseRX4 RXbits4(3) RXbits4(3)+phaseRX4];
phaseRX4 = [phaseRX4 RXbits4(4) RXbits4(4)+phaseRX4];
phaseRX4 = [phaseRX4 RXbits4(5) RXbits4(5)+phaseRX4];
phaseRX4 = [phaseRX4 RXbits4(6) RXbits4(6)+phaseRX4];
phaseRX4 = [phaseRX4 RXbits4(7) RXbits4(7)+phaseRX4];
phaseRX4 = [phaseRX4 RXbits4(8) RXbits4(8)+phaseRX4];
phaseRX4 = [phaseRX4 RXbits4(9) RXbits4(9)+phaseRX4];
phaseRX3 = [RXbits3(1) RXbits3(2) RXbits3(1)+RXbits3(2)];
phaseRX3 = [phaseRX3 RXbits3(3) RXbits3(3)+phaseRX3];
phaseRX3 = [phaseRX3 RXbits3(4) RXbits3(4)+phaseRX3];
phaseRX3 = [phaseRX3 RXbits3(5) RXbits3(5)+phaseRX3];
phaseRX3 = [phaseRX3 RXbits3(6) RXbits3(6)+phaseRX3];
phaseRX3 = [phaseRX3 RXbits3(7) RXbits3(7)+phaseRX3];
phaseRX3 = [phaseRX3 RXbits3(8) RXbits3(8)+phaseRX3];
phaseRX3 = [phaseRX3 RXbits3(9) RXbits3(9)+phaseRX3];
phaseRX2 = [RXbits2(1) RXbits2(2) RXbits2(1)+RXbits2(2)];
phaseRX2 = [phaseRX2 RXbits2(3) RXbits2(3)+phaseRX2];
phaseRX2 = [phaseRX2 RXbits2(4) RXbits2(4)+phaseRX2];
phaseRX2 = [phaseRX2 RXbits2(5) RXbits2(5)+phaseRX2];
phaseRX2 = [phaseRX2 RXbits2(6) RXbits2(6)+phaseRX2];
phaseRX2 = [phaseRX2 RXbits2(7) RXbits2(7)+phaseRX2];
phaseRX2 = [phaseRX2 RXbits2(8) RXbits2(8)+phaseRX2];
phaseRX2 = [phaseRX2 RXbits2(9) RXbits2(9)+phaseRX2];
phaseRX1 = [RXbits1(1) RXbits1(2) RXbits1(1)+RXbits1(2)];
phaseRX1 = [phaseRX1 RXbits1(3) RXbits1(3)+phaseRX1];
phaseRX1 = [phaseRX1 RXbits1(4) RXbits1(4)+phaseRX1];
phaseRX1 = [phaseRX1 RXbits1(5) RXbits1(5)+phaseRX1];
phaseRX1 = [phaseRX1 RXbits1(6) RXbits1(6)+phaseRX1];
phaseRX1 = [phaseRX1 RXbits1(7) RXbits1(7)+phaseRX1];
phaseRX1 = [phaseRX1 RXbits1(8) RXbits1(8)+phaseRX1];
phaseRX1 = [phaseRX1 RXbits1(9) RXbits1(9)+phaseRX1];

% Convert the DOA angle to channel phase
[STEER_CH2, STEER_CH3] = DOA2phase(ZE, AZ);
if (STEER_CH2<0)
    STEER_CH2 = STEER_CH2+360;
end
if (STEER_CH3<0)
    STEER_CH3 = STEER_CH3+360;
end

% Flip the sign of the phase since we are simulating an incoming DOA
if (STEER_CH2 ~= 0)
    STEER_CH2 = (360-STEER_CH2);
end
if (STEER_CH3 ~= 0)
    STEER_CH3 = (360-STEER_CH3);
end

```

```
% Get channel 4
STEER_CH4=STEER_CH2+STEER_CH3;
if (STEER_CH4>=360)
    STEER_CH4 = STEER_CH4-360;
end

% get the closest value acquirable on the phase shifters
[val,I] = min(abs(phaseTX2-STEER_CH2));
STEER_CH2 = phaseUC(I);
STEER_CH2_TRUE = phaseTX2(I);
[val,I] = min(abs(phaseTX3-STEER_CH3));
STEER_CH3 = phaseUC(I);
STEER_CH3_TRUE = phaseTX3(I);
[val,I] = min(abs(phaseTX4-STEER_CH4));
STEER_CH4 = phaseUC(I);


%% Run the test
tic

% Preallocate the power matrix
power = zeros(length(phaseRX2),length(phaseRX3));

% Set the steer phase
TX2true = UART_SendWait4Reply(DOA, sprintf('setPhase 2 %s', num2str(STEER_CH2)));
TX3true = UART_SendWait4Reply(DOA, sprintf('setPhase 3 %s', num2str(STEER_CH3)));
TX4true = UART_SendWait4Reply(DOA, sprintf('setPhase 4 %s', num2str(STEER_CH4)));

% Scan
for c=1:length(phaseRX2)
    UART_SendWait4Reply(RDA, sprintf('setPhase 2 %s', num2str(phaseUC(c))));
    for v=1:length(phaseRX3)
        UART_SendWait4Reply(RDA, sprintf('setPhase 3 %s', num2str(phaseUC(v))));
        chan4 = (phaseRX2(c)+phaseRX3(v));
        if (chan4>=360)
            chan4 = chan4-360;
        end
        [val,I] = min(abs(phaseRX4-chan4));
        chan4 = phaseUC(I);
        UART_SendWait4Reply(RDA, sprintf('setPhase 4 %s', num2str(chan4)));
        fprintf(fieldfox, 'CALC:MARK1:Y?');
        temp = str2double(strsplit(fscanf(fieldfox),','));
        power(v,c) = temp(1);
        GPIB_Wait(fieldfox);
    end
end

save(sprintf('AZ%.1fZE%.1f.mat',...
    AZ,ZE),'power','phaseRX2','phaseRX3',...
    'phaseRX4','STEER_CH2','STEER_CH3')
toc

    end
end
```

B.4.2 Results Analysis

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ScanningTestAnalysis.m
%
% Author: Justin Long
% Date: 6/15/2018
% Updated: 9/4/2019
% This work was supported by a NASA Space Technology Research Fellowship
%
% This script runs analysis on data files generated by ScanningTest.m or FielFoxTest.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;clc;close all;
filename = 'AZ60ZE30.mat';
res = 1.4;
spacing = 0.5;
% Import data
load(filename);

[X, Y] = meshgrid(phaseRX2,phaseRX3);
X(X>180) = X(X>180)-360;
Y(Y>180) = Y(Y>180)-360;

% Reshape to 2 dimensions
power = reshape(power,1,[]);
X = reshape(X,1,[]);
Y = reshape(Y,1,[]);

% Determine simulated DOA
[DOA_ZE, DOA_AZ] = phase2DOA(360-STEER_CH2,360-STEER_CH3);

% remove non-existent angles
real = findFalse(X,Y);
X = X(real);
Y = Y(real);
power = power(real);

% Plot on the phase-phase axis
scatter3(X,Y,power,5,power)
xlabel('phaseX ()')
ylabel('phaseY ()')
title('power throughout phase shift range');
colorbar;

% Convert phase shifts to ZE, AZ angles
[ZE,AZ] = phase2DOA(X,Y);

% Plot on the ZE, AZ axis
figure;
scatter3(ZE,AZ,power,5,power)
title('power throughout AZ-ZE range');
xlabel('ZE ()')
```

```
ylabel('AZ ()')
colorbar;

% Get the power scan angle (full range)
[maxPower,I] = max(power);
powerScanZE = ZE(I);
powerScanAZ = AZ(I);
```

B.4.3 Algorithms and Coordinate Transforms

```
function [X,Y] = DOA2phase(ZE,AZ)
%DOA2phase converts the DOA to the corresponding X,Y phase shifts
%   input the DOA (ZE,AZ) as pairwise vectors with units in degrees
%   The power will be returned in whatever format it is inputted

AZ = deg2rad(AZ);
ZE = deg2rad(ZE);

X = 0.5 .* sin(ZE) .* cos(AZ) * 360;
Y = 0.5 .* sin(ZE) .* sin(AZ) * 360;
end
```

```
function [ZE,AZ] = phase2DOA(X,Y)
%phase2DOA Convert phase vectors to DOA
%   Input coordinate mapped vactors (reshape if necessary) for X,Y, power
%   ZE, AZ, and power are outputted as coordinate mapped vectors in degrees

X(X>180) = X(X>180) - 360;
Y(Y>180) = Y(Y>180) - 360;
X(X<=-180) = X(X<=-180) + 360;
Y(Y<=-180) = Y(Y<=-180) + 360;
X = deg2rad(X);
Y = deg2rad(Y);

AZ = atan2(Y,X);
ZE = asin(sqrt((X.^2+Y.^2))/pi);
AZ = rad2deg(AZ);
ZE = rad2deg(ZE);
end
```

```
function [ I ] = findFalse( X,Y )
%findFalse returns a matrix with ones for phase combinations that can exist
%   give the phase combinations two arrays, pair matched, in degrees
X(X>180) = X(X>180)-360;
Y(Y>180) = Y(Y>180)-360;
X = deg2rad(X);
Y = deg2rad(Y);
I = rad2deg(sqrt(X.^2+Y.^2)) <= 180+1;
end
```

B.4.4 Instrument Control

```
function [ reply ] = UART_SendWait4Reply( obj, send )
%UART_SendWait4Reply sends whatever is in "send" to obj and then waits for a
%reply which is returned. send must be a strin, reply is a string.
    fprintf(obj, send)
    while (obj.BytesAvailable == 0)
        pause(0.01);
    end
    while (~strcmp(obj.TransferStatus,'idle'))
        pause(0.01);
    end
    reply = fscanf(obj);
    while (obj.BytesAvailable == 0)
        pause(0.01)
    end
    while (~strcmp(obj.TransferStatus,'idle'))
        pause(0.01)
    end
    while (obj.BytesAvailable > 5)
        reply = [reply '  ;  ' fscanf(obj)];
        pause(0.01)
    end
end
```

```
function [ ] = GPIB_Wait( obj )
%wait for the specified GPIB object to become ready
    fprintf(obj, '*opc')
    fprintf(obj, '*opc?')
    k = fscanf(obj);

    while k ~= '1 '
        fprintf(obj, '*opc?')
        k = fscanf(obj);
    end

end
```

B.5 RDA Prototype Code

```
#ifndef __RDA_H
#define __RDA_H

// Define array spacing (in wavelengths)
#define spacing 0.5

// Define channels
#define channel1 1
#define channel2 2
#define channel3 3
#define channel4 4

// Phase shifter fata bits on P5
#define DATA0 BIT7
```

```
#define DATA1 BIT6
#define DATA2 BIT5
#define DATA3 BIT4
#define DATA4 BIT3
#define DATA5 BIT2
#define DATA6 BIT1
#define DATA7 BIT0

// Phase shifter latch enable on P6
#define LE0 BIT0
#define LE1 BIT1
#define LE2 BIT2
#define LE3 BIT3

// HPA, detector, and switches on P7
#define HPA_EN BIT0
#define SW_CTRL BIT1
#define SW_EN BIT2
#define DET_SW_EN BIT3
#define DET_SW_CTRL BIT4
#define DET_EN BIT5

// PhaseShifter Calibration Table
// Note: bits must be ascencing in value (i.e. PS2_BIT_none < PS2_BIT_8)
// don't add 360 for phasewrapping
#define PS1_BIT_none 0.0
#define PS1_BIT_1 1.0
#define PS1_BIT_2 2.1
#define PS1_BIT_3 4.6
#define PS1_BIT_4 8.9
#define PS1_BIT_5 11.8
#define PS1_BIT_6 45.3
#define PS1_BIT_7 87.9
#define PS1_BIT_8 185.6

#define PS2_BIT_none -1.4
#define PS2_BIT_1 -0.4
#define PS2_BIT_2 0.8
#define PS2_BIT_3 3.2
#define PS2_BIT_4 7.7
#define PS2_BIT_5 21.6
#define PS2_BIT_6 44.0
#define PS2_BIT_7 86.8
#define PS2_BIT_8 184.2

#define PS3_BIT_none 8.5
#define PS3_BIT_1 9.6
#define PS3_BIT_2 10.8
#define PS3_BIT_3 13.3
#define PS3_BIT_4 18.0
#define PS3_BIT_5 31.5
#define PS3_BIT_6 53.7
#define PS3_BIT_7 96.3
#define PS3_BIT_8 192.9
```

```

#define PS4_BIT_none -6.6
#define PS4_BIT_1 -5.7
#define PS4_BIT_2 -4.6
#define PS4_BIT_3 -2.3
#define PS4_BIT_4 2.7
#define PS4_BIT_5 16.4
#define PS4_BIT_6 40.3
#define PS4_BIT_7 81.4
#define PS4_BIT_8 175.5

// define math constant
#define pi 3.14159265

// global variables
extern float channel1Shift, channel2Shift, channel3Shift, channel4Shift, Azimuth, Zenith;

// Function Prototypes
int setShift(unsigned char channelSelect, float shiftSelect);
int TX(void);
int RX(void);
int RDA_setup(void);
int resetShifter(void);
int calTable(void);
int setAngle(float ZE, float AZ);
int angles(void);
#endif
```

```
#include <msp430.h>
#include <ctl.h>
#include <ARCbus.h>
#include <Error.h>
#include <terminal.h>
#include <math.h>
#include <string.h>           // for memset function
#include <UCA2_uart.h>        // UART setup
#include "pins.h"
#include "RDA.h"

float dataValue, channel1Shift, channel2Shift, channel3Shift, channel4Shift, Azimuth,
 ↪  Zenith;
float PS_BIT_none, PS_BIT_0, PS_BIT_1, PS_BIT_2, PS_BIT_3, PS_BIT_4, PS_BIT_5, PS_BIT_6,
 ↪  PS_BIT_7, PS_BIT_8;
float PS1_table[256], PS2_table[256], PS3_table[256], PS4_table[256];

// The detector is not used on the prototype, so no code is included here
// However, a switch command is included to operate that switch.

// Switch to the detector
int detectorSW(void){
        P7OUT |= DET_SW_CTRL;
        return 0;
}
// Switch to the MMCX output
int outputSW(void){
```

```
26          P7OUT &= ~DET_SW_CTRL;
27          return 0;
28  }
29
30  int RDA_setup(void){
31          // TR switches, HPA, detector setup
32          P7DIR = 0xFF;                   // Set P7 as output
33          P7OUT &= ~HPA_EN;               // Ensure that the PA is disabled
34          P7OUT |= SW_CTRL;               // Set the switch to RX mode
35          P7OUT |= SW_EN + DET_SW_EN;     // Enable the switches
36
37          // Phase shifter setup
38          // Set the pins as outputs, clear the LE lines, and initialize the phase shifter
             ↪  to 0
39          P6DIR = 0x0F;
40          P5DIR = 0xFF;
41          // Generate the phase lookup tables
42          calTable();
43          resetShifter();
44          return 0;
45  }
46
47  int calTable(void){
48          // Generates the phase lookup tables from the calibration values
49          int i;
50          for (i=0; i<=255; i++) {
51                  if(((i/128)%2) == 1) {
52                          PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_8;
53                          PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_8;
54                          PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_8;
55                          PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_8;
56                  }
57                  if(((i/64)%2) == 1) {
58                          PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_7;
59                          PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_7;
60                          PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_7;
61                          PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_7;
62                  }
63                  if(((i/32)%2) == 1) {
64                          PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_6;
65                          PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_6;
66                          PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_6;
67                          PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_6;
68                  }
69                  if(((i/16)%2) == 1) {
70                          PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_5;
71                          PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_5;
72                          PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_5;
73                          PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_5;
74                  }
75                  if(((i/8)%2) == 1) {
76                          PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_4;
77                          PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_4;
78                          PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_4;
```

```
                        PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_4;
                }
                if(((i/4)%2) == 1) {
                        PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_3;
                        PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_3;
                        PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_3;
                        PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_3;
                }
                if(((i/2)%2) == 1) {
                        PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_2;
                        PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_2;
                        PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_2;
                        PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_2;
                }
                if((i%2) == 1) {
                        PS1_table[i+1] = PS1_table[i+1] + PS1_BIT_1;
                        PS2_table[i+1] = PS2_table[i+1] + PS2_BIT_1;
                        PS3_table[i+1] = PS3_table[i+1] + PS3_BIT_1;
                        PS4_table[i+1] = PS4_table[i+1] + PS4_BIT_1;
                }
                if(i == 0) {
                        PS1_table[i+1] = PS1_BIT_none;
                        PS2_table[i+1] = PS2_BIT_none;
                        PS3_table[i+1] = PS3_BIT_none;
                        PS4_table[i+1] = PS4_BIT_none;
                }
        }

        return 0;
}

int resetShifter(void){
        // Clear the latch enables
        P6OUT = 0x00;
        // reset all the phase shifters to broadside
        setShift(channel1,0);
        setShift(channel2,0);
        setShift(channel3,0);
        setShift(channel4,0);
        reutn 0;
}

int TX(void){
        P7OUT &= ~SW_CTRL;              // Set the switch to TX mode
        // Switching Speed is 1.7 us, so delay before we enable PA
        // Assuming MCLK speed of 25 MHz, 1.7E-6*25E6 = 43 clock cylces
        __delay_cycles(50);
        P7OUT |= HPA_EN;                // Enable the HPA
        return 0;
}

int RX(void){
        P7OUT &= ~HPA_EN;               // Disable the HPA
```

```
132         // The HPA does not provide timing information. However, some time should be
            ↪  allowed for the transients to die
133         __delay_cycles(50);
134         P7OUT |= SW_CTRL;             // Set the switch to RX mode
135         return 0;
136 }
137
138 int setShift(unsigned char channelSelect, float shiftSelect){
139         // setShift sets the data bits and programs the specified phase shifter
140         // the input shiftSelect is the desired (nominal) phase shift in degrees.
141         // the input channelSelect is the desired channel on which to execute the phase
            ↪  shift.
142
143         int index = 1;
144         int i;
145
146         // initialize the variable that tracks the actual set phase shift
147         dataValue = 0;
148         // Initilize the calibration variables
149         PS_BIT_none = 0;
150         PS_BIT_1 = 0;
151         PS_BIT_2 = 0;
152         PS_BIT_3 = 0;
153         PS_BIT_4 = 0;
154         PS_BIT_5 = 0;
155         PS_BIT_6 = 0;
156         PS_BIT_7 = 0;
157         PS_BIT_8 = 0;
158
159         // Select the appropriate calibration table, find the closest value
160         switch (channelSelect) {
161                 case channel1:
162                         // Handle phase wrapping issues
163                         if(shiftSelect<PS1_table[1]) {
164                                 shiftSelect = shiftSelect+360;
165                         }
166                         if(shiftSelect>PS1_table[256]) {
167                                 shiftSelect = shiftSelect-360;
168                         }
169                         // Initialize the tracking variable
170                         dataValue = PS1_table[1];
171                         // Find the closest value in the calibrated lookup array
172                         for(i=2;i<=256;++i){
173                                 if(abs(PS1_table[i]-shiftSelect)<abs(dataValue-shiftSele
                                    ↪  ct)){
174                                         dataValue = PS1_table[i];
175                                         index = i;
176                                 }
177                         }
178                 case channel2:
179                         // Handle phase wrapping issues
180                         if(shiftSelect<PS2_table[1]) {
181                                 shiftSelect = shiftSelect+360;
182                         }
```

```
183                         if(shiftSelect>PS2_table[256]) {
184                                 shiftSelect = shiftSelect-360;
185                         }
186                         // Initialize the tracking variable
187                         dataValue = PS2_table[1];
188                         // Find the closest value in the calibrated lookup array
189                         for(i=2;i<=256;++i){
190                                 if(abs(PS2_table[i]-shiftSelect)<abs(dataValue-shiftSele
                                    ↪  ct)){
191                                         dataValue = PS2_table[i];
192                                         index = i;
193                                 }
194                         }
195                 case channel3:
196                         // Handle phase wrapping issues
197                         if(shiftSelect<PS3_table[1]) {
198                                 shiftSelect = shiftSelect+360;
199                         }
200                         if(shiftSelect>PS3_table[256]) {
201                                 shiftSelect = shiftSelect-360;
202                         }
203                         // Initialize the tracking variable
204                         dataValue = PS3_table[1];
205                         // Find the closest value in the calibrated lookup array
206                         for(i=2;i<=256;++i){
207                                 if(abs(PS3_table[i]-shiftSelect)<abs(dataValue-shiftSele
                                    ↪  ct)){
208                                         dataValue = PS3_table[i];
209                                         index = i;
210                                 }
211                         }
212                 case channel4:
213                         // Handle phase wrapping issues
214                         if(shiftSelect<PS4_table[1]) {
215                                 shiftSelect = shiftSelect+360;
216                         }
217                         if(shiftSelect>PS4_table[256]) {
218                                 shiftSelect = shiftSelect-360;
219                         }
220                         // Initialize the tracking variable
221                         dataValue = PS4_table[1];
222                         // Find the closest value in the calibrated lookup array
223                         for(i=2;i<=256;++i){
224                                 if(abs(PS4_table[i]-shiftSelect)<abs(dataValue-shiftSele
                                    ↪  ct)){
225                                         dataValue = PS4_table[i];
226                                         index = i;
227                                 }
228                         }
229         }
230
231         // Set the corresponding data bits
232         if((((index-1)/128)%2) == 1) {
233                 P5OUT |= DATA7;
```

```
234         } else {
235                 P5OUT &= ~DATA7;
236         }
237         if((((index-1)/64)%2) == 1) {
238                 P5OUT |= DATA6;
239         } else {
240                 P5OUT &= ~DATA6;
241         }
242         if((((index-1)/32)%2) == 1) {
243                 P5OUT |= DATA5;
244         } else {
245                 P5OUT &= ~DATA5;
246         }
247         if((((index-1)/16)%2) == 1) {
248                 P5OUT |= DATA4;
249         } else {
250                 P5OUT &= ~DATA4;
251         }
252         if((((index-1)/8)%2) == 1) {
253                 P5OUT |= DATA3;
254         } else {
255                 P5OUT &= ~DATA3;
256         }
257         if((((index-1)/4)%2) == 1) {
258                 P5OUT |= DATA2;
259         } else {
260                 P5OUT &= ~DATA2;
261         }
262         if((((index-1)/2)%2) == 1) {
263                 P5OUT |= DATA1;
264         } else {
265                 P5OUT &= ~DATA1;
266         }
267         if(((index-1)%2) == 1) {
268                 P5OUT |= DATA0;
269         } else {
270                 P5OUT &= ~DATA0;
271         }
272         if(index == 1) {
273                 P5OUT &= 0x00;
274         }
275
276
277         // For the desired channel, shift LE high, update the corresponding global
             ↪  variable, and then shift LE back to low.
278         // The minimum pulse with of the LE must be 30 ns (which is < 1 clock cycle)
279         switch (channelSelect) {
280                 case channel1:
281                         P6OUT |= LE0;
282                         channel0Shift = dataValue;
283                         printf("Phase shift on channel 1 set to %.1f degrees\r\n",
                             ↪  channel1Shift);
284                         P6OUT &= ~LE0;
285                         break;
```

```
286                 case channel2:
287                         P6OUT |= LE1;
288                         channel1Shift = dataValue;
289                         printf("Phase shift on channel 2 set to %.1f degrees\r\n",
                             ↪  channel2Shift);
290                         P6OUT &= ~LE1;
291                         break;
292                 case channel3:
293                         P6OUT |= LE2;
294                         channel2Shift = dataValue;
295                         printf("Phase shift on channel 3 set to %.1f degrees\r\n",
                             ↪  channel3Shift);
296                         P6OUT &= ~LE2;
297                         break;
298                 case channel4:
299                         P6OUT |= LE3;
300                         channel3Shift = dataValue;
301                         printf("Phase shift on channel 4 set to %.1f degrees\r\n",
                             ↪  channel4Shift);
302                         P6OUT &= ~LE3;
303                         break;
304                 default:
305                         printf("Error in latchShift: went into default case\r\n");
306                         break;
307         }
308
309         return 0;
310 }
311
312 int setAngle(float ZE, float AZ) {
313         float X, Y, XY;
314
315         // Convert the desired angles to phase shifts
316         X = spacing * sin(ZE*pi/180) * cos(AZ*pi/180) * 360;
317         Y = spacing * sin(ZE*pi/180) * sin(AZ*pi/180) * 360;
318         XY = X+Y;
319
320         // Set those phase shifts
321         setPhase(channel1,0);
322         setPhase(channel2,X);
323         setPhase(channel3,Y);
324         setPhase(channel4,XY);
325         angles(void);
326 }
327
328 int angles(void){
329         float X,Y;
330         // Convert the actual phase shifts back to the actual angles
331         X = channel2Shift - channel1Shift;
332         Y = channel3Shift - channel1Shift;
333         Azimuth = atan2(Y*(pi/180),X*(pi/180))*(180/pi);
334         Zenith = asin( sqrt((X*(pi/180)*X*(pi/180))+(Y*(pi/180)*Y*(pi/180))) /
             ↪  pi)*(180/pi)*(0.5/spacing);
```

```
        // There seems to be some small error in the Zenith equation when spacing !=
         ↪  0.5...
        // It grows as the spacing moves away from 0.5. At 0.4, error is <3%.
        return 0;
}

/*****************************************************************************************
 ↪  ****************************************************
The commands.c file is for commands that will be displayed through the serial terminal.
In order to add a command you must create a function as seen below.
Then function must be added to the "const CMD_SPEC cmd_tbl[]={{"help","
 ↪  [command]",helpCmd}" table at the end of the file.
*****************************************************************************************
 ↪  ****************************************************/
#include <msp430.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <terminal.h>
#include <commandLib.h>
#include <stdlib.h>
#include <ARCbus.h>
#include <SDlib.h>
#include <i2c.h>
#include "RDA.h"

//************************************************************ passing arguments over the
 ↪  terminal ***********************************************

int SW(char **argv, unsigned short argc){
// SW [arg] switches the HPA board into TX or RX mode.
// arg must be "TX" or "RX"
  if (argc > 1) {
    RX();
    printf("Error. Too many inputs. Input should be RX or TX.\r\nSwitched to RX
     ↪  mode\r\n");
  } else if (argc = 0) {
      RX();
      printf("Error. Must provide an input. Input should be RX or TX.\r\nSwitched to RX
       ↪  mode\r\n");
  } else if (~string_compare(argv[1],"TX")) {
      TX();
      printf("Switched to TX mode\r\n");
  } else if (~string_compare(argv[1],"RX")) {
      RX();
      printf("Switched to RX mode\r\n");
  } else {
      RX();
      printf("Error, invalid input. Switched to RX mode\r\n");
  }
  return 0;
}

int phase(char **argv, unsigned short argc){
```

```
43    // "phase [arg1] [arg2]" sets the channel (given by arg1) to a desired phase (given by
      ↪  arg2).
44    // arg1 must be 1,2,3, or 4
45    // Check the input, make sure the inputs are reasonable
46    if (argc < 2) {
47      printf("Not enough arguments.\r\n");
48    } else if (argc > 2) {
49        printf("Too many arguments.\r\n");
50    } else if (strcmp(argv[1],"1")&&strcmp(argv[1],"2")&&strcmp(argv[1],"3")&&strcmp(argv[
      ↪  1],"4"))
      ↪  {
51        printf ("Incorrect input. [arg1] must be 1,2,3, or 4\r\n");
52    } else {
53        // set the phase shift
54        setShift(atoi(argv[1]),strtof(argv[2],NULL));
55    }
56    return 0;
57  }
58
59  int detector(char **argv, unsigned short argc){
60    // "detector" switches to the detector
61    detectorSW();
62    printf("Switched to the detector.\r\n");
63    return 0;
64  }
65
66  int output(char **argv, unsigned short argc){
67    // "output" swiches to the RF output port
68    outputSW();
69    printf("Switched to the RF output.\r\n");
70    return 0;
71  }
72
73  int readPhase(char **argv, unsigned short argc){
74    // "readPhase" lists the current phase shift of each channel
75    printf("Channel 0: %.1f\r\nChannel 1: %.1f\r\nChannel 2: %.1f\r\nChannel 3:
      ↪  %.1f\r\n",channel0Shift,channel1Shift,channel2Shift,channel3Shift);
76    return 0;
77  }
78
79  int readAngle(char **argv, unsigned short argc){
80    // "readAngle" returns the current beamformed ZE and AZ angles of the RDA
81    printf("ZE: %.1f\r\nAZ: %.1f\r\n",Zenith,Azimuth);
82    return 0;
83  }
84
85  int resetPhase(char **argv, unsigned short argc){
86    // "resetPhase" resets all phase shifters to broadside
87    resetShifter();
88    printf("Phase Shifters all set to 0 degrees, ports reinitalized");
89    return 0;
90  }
91
92  int beamform(char **argv, unsigned short argc){
```

```
93    // "beamform [arg1] [arg2]" beamforms to the desired Zenith [arg1] and Azimuth [arg2]
94    // Check the input, make sure the inputs are reasonable
95    if (argc < 2) {
96      printf("Not enough arguments.\r\n");
97    } else if (argc > 2) {
98        printf("Too many arguments.\r\n");
99    } else {
100       // beamform to angle
101       setAngle(strtof(argv[1],NULL)),strtof(argv[2],NULL));
102   }
103   return 0;
104 }
105
106
107 /************************************************************ Using the Timer_A1
      ↪  ************************************************************
108 * DONT USE TIMER0_Ax_VECTOR !!! this interrupt is use in library code and will cause a
      ↪  collision
109 * Use TIMERx_Ay_VECTOR x=2,3 & y=0,1
110 * TIMER0_Ax_VECTOR used in ARClib ?
111 * TIMER1_Ax_VECTOR used in ????
112 ********************************************************************************************
      ↪  ************************************************/
113 int example_timer_IR(char **argv,unsigned short argc){
114   int timer_check;
115   WDTCTL = WDTPW+WDTHOLD;                                    // Stop WDT
116   P7DIR |= 0xFF;                                             // Setting port 7 to drive
      ↪  LED's (0xFF ==1111 1111)
117   P7OUT = 0x00;                                              // Set all LED's on port 7 to
      ↪  start all off
118 //********************************** Set up clock [0]
119   TA2CTL |= TASSEL__ACLK | MC_2;                             // Setting Timer_A to
      ↪  ACLK(TASSEL_1) to continuous mode(MC_2)
120
121 //********************************** Set timer interrupt enable [1]
122   TA2CCTL0 |= CCIE;                                           // Capture/compare
      ↪  interrupt enable #0
123   TA2CCTL1 |= CCIE;                                           // Capture/compare
      ↪  interrupt enable #1
124
125 //********************************** Set the timer count IR value [2]
126   TA2CCR0 = 10000;                                           // Timer0_A3
      ↪  Capture/Compare @ 10000 counts
127   TA2CCR1 = 1000;                                            // TA0IV_1 Capture/Compare
      ↪  @ 1000 counts
128
129    while (1)                                                 // poll in while loop until
       ↪  a key press
130    {
131       if ((timer_check=getchar()) != EOF)
132      {
133       break;                                                  // break out of loop if a
          ↪  key is pressed
134      }
```

```
135     ctl_events_wait(CTL_EVENT_WAIT_ANY_EVENTS,0, 1<<15, CTL_TIMEOUT_DELAY, 1000); //
        ↪ wait in loop
136    }
137   reset(0,ERR_SRC_CMD,CMD_ERR_RESET,0);                    // reset clock registers
138   return 0;
139 }
140
141 // ******************************** Timer_A0 interrupt code
    ↪ ********************************************************************************
142 //NOTE: to use interrupts you usually need need to enable global interrupts using
    ↪ _ENIR();
143 //      This is not necessary when using ARClib code as the enable is called in ARClib
144 //NOTE: For P1IV you still need to enable P1IE registers
145 //*******************************************************************************************
    ↪ ********************************************************
146 void Timer_A2_A0(void)__interrupt[TIMER2_A0_VECTOR]{     // Timer A0 interrupt service
    ↪ routine TA0IV_TA0IFG.
147   P7OUT^=BIT0; // toggle LEDs when IR is called
148 }
149
150 void Timer_A2_A1(void)__interrupt[TIMER2_A1_VECTOR]{     // Timer A0 interrupt service
    ↪ routine for capture comp 1 and 2
151         P7OUT^=BIT1; // light LEDs
152 }
153
154 //****************************************** Using the SD card
    ↪ ********************************************************************************
155
156 int SD_write(char **argv,unsigned short argc){
157 char buff[512];
158 int mmcReturnValue, result , i;
159
160   mmcInit_msp();  // Sets up the interface for the card
161   mmc_pins_on();  //Sets up MSP to talk to SD card
162   mmcReturnValue=mmcInit_card();
163
164   if (mmcReturnValue==MMC_SUCCESS){ // check good initialization
165     printf("\rCard initalized Sucessfully\r\n");
166   }
167   else{
168     printf("Check SD card.\r\nInitalized failed.\r\n Error %i\r\n",mmcReturnValue);
169   }
170
171 //populate buffer block
172   for(i=1;i<=argc;i++) {// ignore 0 *argv input (argv[0]--> "SD_write" )
173     strcat(buff,argv[i]); // appends chars from one string to another
174     strcat(buff,"|");     // separates strings with | as strcat eats NULL
175   }
176
177 //write to SD card
178   result= mmcWriteBlock(100,buff); //(unsigned char*) casting my pointer(array) as a
    ↪ char
179
180   if (result>=0){ // check SD write
```

```
181   printf("SD card write success.\r\n");
182   }
183   else{
184     printf("SD card write failed.\r\nError %i\r\n",result);
185   }
186   return 0;
187 }
188
189 int SD_read(char **argv,unsigned short argc){
190   #define ASCIIOUT_STR  "%c "
191  char buffer[512];
192  int resp , i;
193
194   //read from SD card
195  resp=mmcReadBlock(100,buffer);
196   //print response from SD card
197   printf("%s\r\n",SD_error_str(resp));
198
199         for(i=0;i<9;i++){//changed the 512 to 256 which is a result of changing CHAR TO
          ↪  INT
200
201         if(i<8){
202           printf(ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR "\r\n",
           buffer[i*28+3],buffer[i*28+4],buffer[i*28+5],buffer[i*28+6],buffer[i*28+7],buf
            ↪  fer[i*28+8],buffer[i*28+9],buffer[i*28+10],buffer[i*28+11],buffer[i*28+12]
            ↪  ,buffer[i*28+13],
           buffer[i*28+14],buffer[i*28+15],buffer[i*28+16],buffer[i*28+17],buffer[i*28+18
            ↪  ],buffer[i*28+19],buffer[i*28+20],buffer[i*28+21],buffer[i*28+22],buffer[i
            ↪  *28+23],
           buffer[i*28+24],buffer[i*28+25],buffer[i*28+26],buffer[i*28+27],buffer[i*28+28
            ↪  ],buffer[i*28+29],buffer[i*28+30]);
           }
207
208         else{
209           printf(ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR
            ↪  ASCIIOUT_STR ASCIIOUT_STR ASCIIOUT_STR "\r\n",
           buffer[i*28+3],buffer[i*28+4],buffer[i*28+5],buffer[i*28+6],buffer[i*28+7],buf
            ↪  fer[i*28+8],buffer[i*28+9],buffer[i*28+10],buffer[i*28+11],buffer[i*28+12]
            ↪  ,buffer[i*28+13],
           buffer[i*28+14],buffer[i*28+15],buffer[i*28+16],buffer[i*28+17],buffer[i*28+18
            ↪  ],buffer[i*28+19],buffer[i*28+20],buffer[i*28+21],buffer[i*28+22],buffer[i
            ↪  *28+23],
           buffer[i*28+24],buffer[i*28+25],buffer[i*28+26],buffer[i*28+27],buffer[i*28+28
            ↪  ],buffer[i*28+29],buffer[i*28+30]);
           }
```

```
214         }
215         return 0;
216 }
217
218 //************************************************************ Sending an I2C packet
219 int send_I2C(char **argv,unsigned short argc){
220   unsigned short addr, cmd_id, resp, payload_count=argc-2, i;
221   unsigned char packet[4+BUS_I2C_HDR_LEN+BUS_I2C_CRC_LEN], *payload;
222
223   if (argc < 2) {
224     printf("Not enough arguments.\n");
225     return 0;
226   }
227
228   // Get integer value of address from passed string
229   addr=strtoul(argv[1], NULL, 0);
230   // Get integer value of I2C command from passed string
231   cmd_id=strtoul(argv[2], NULL, 0);
232
233   // Initialize packet with source I2C address, and I2C command
234   payload=BUS_cmd_init(packet, cmd_id);
235
236   for (i=payload_count; i > 0; --i) {
237     payload[payload_count-i]=strtoul(argv[payload_count-i+2], NULL, 0);
238   }
239
240   // Transmit packet to addr. Payload is zero length.
241   resp=BUS_cmd_tx(addr, packet, payload_count, 0);
242   if (resp == 0) {
243     printf("Valid response\n");
244   }
245   else {
246     printf("Error: %d\n", resp);
247   }
248   /*initI2C(4,1,0);
249   // I2C_tx expects a char* for transmitted data, not an integer
250   resp=i2c_tx(addr, (unsigned char*)&cmd_id, 1);
251   if (resp < 0){
252     printf("I2C transaction error.\n");
253   }*/
254
255   return 0;
256 }
257
258
259 //table of commands with help
260 const CMD_SPEC cmd_tbl[]={{"help"," [command]",helpCmd},
261                     {"timer_IR","[time]...\r\n\tExample command to show how the timer can
                        ↪  be used as an interupt",example_timer_IR},
262                     {"SD_write","Writes given args to the SD card",SD_write},
263                     {"SD_read","",SD_read},
264                     {"send_I2C","Sends I2C command to subsystem",send_I2C},
265
266                     // Phase Shifter Commands
```

```
267                                     {"beamform","[arg1] [arg2] ... \r\n\t""Command that
                                         ↪  sets a beamforms to the desired Zenith [arg1] and
                                         ↪  Azimuth [arg2]",beamform},
268                     {"phase","[arg1] [arg2] ... \r\n\t""Command that sets a channel
                         ↪  [arg1] (0,1,2, or 3) to a desired phase [arg2] (0 <= arg2 <
                         ↪  360)",phase},
269                                     {"readAngle","returns the current beamformed ZE and
                                         ↪  AZ angles of the RDA", readAngle},
270                     {"readPhase","lists the current phase shift of each channel",
                         ↪  readPhase},
271                     {"resetPhase","resets the phase shifters to broadise and resets the
                         ↪  latch enable lines", resetPhase},
272
273                     // HPA commands
274                     {"SW","[arg] ... \r\n\t""Command that sets the HPA board into TX or
                         ↪  RX. [arg] must be TX or RX. Invalid input default to RX",SW},
275
276                     // DET commands
277                     {"detector","Switches the DET board RF path to the internal
                         ↪  detector",detector},
278                     {"output","Switches the DET board RF path to the external
                         ↪  port",output},
279
280                     ARC_COMMANDS,CTL_COMMANDS,ERROR_COMMANDS,
281
282                     //end of list
283                     {NULL,NULL,NULL}};
284
285 // internal commands to avoid libraries
286 int string_compare(char str1[], char str2[])
287 {
288     int ctr=0;
289
290     while(str1[ctr]==str2[ctr])
291     {
292         if(str1[ctr]=='\0'||str2[ctr]=='\0')
293             break;
294         ctr++;
295     }
296     if(str1[ctr]=='\0' && str2[ctr]=='\0')
297         return 0;
298     else
299         return -1;
300 }
```

Appendix C

Attachments

C.1 Schematics

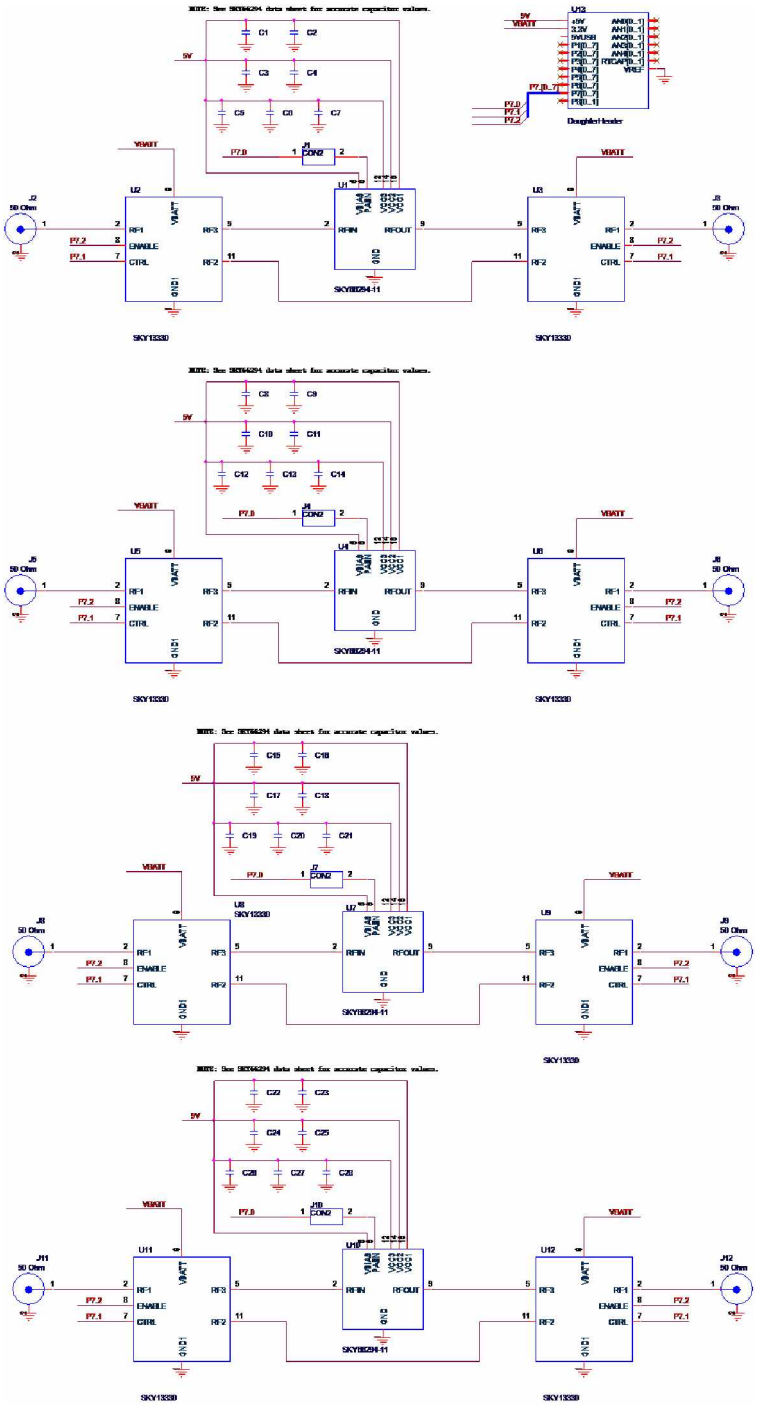


Figure C.1: HPA board Schematic.

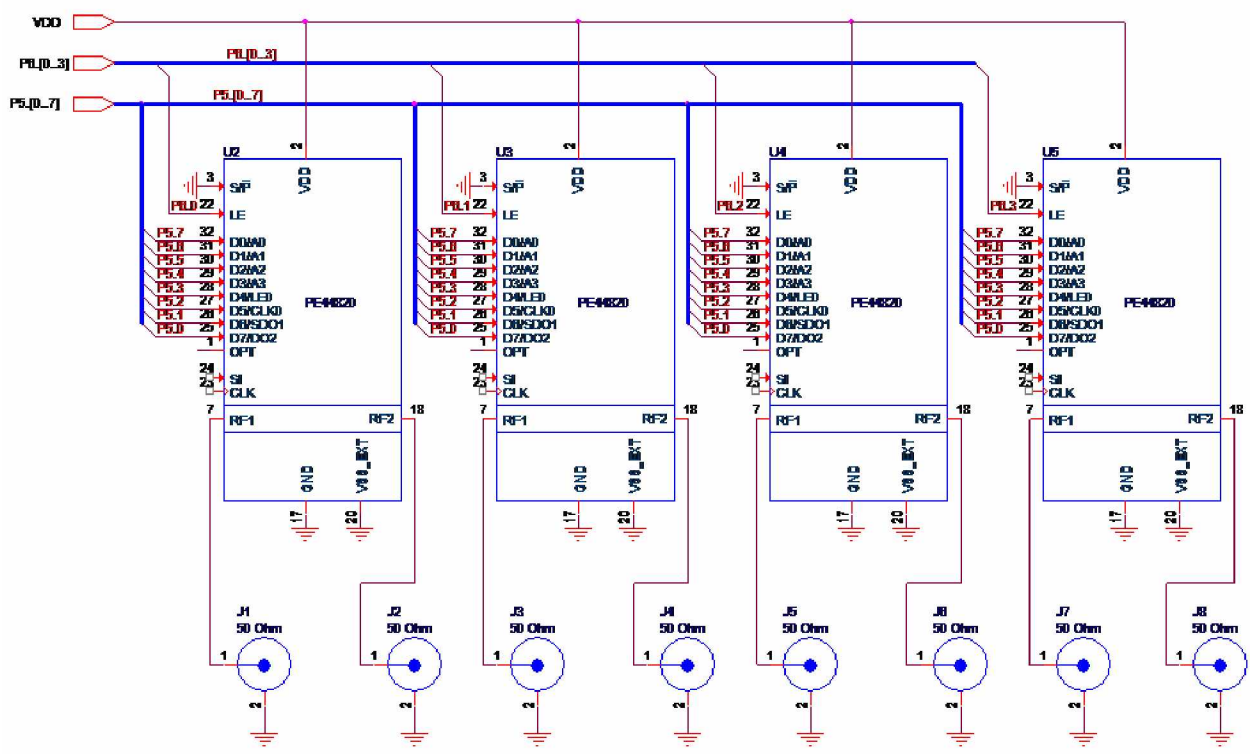


Figure C.2: Phase shifter board schematic.

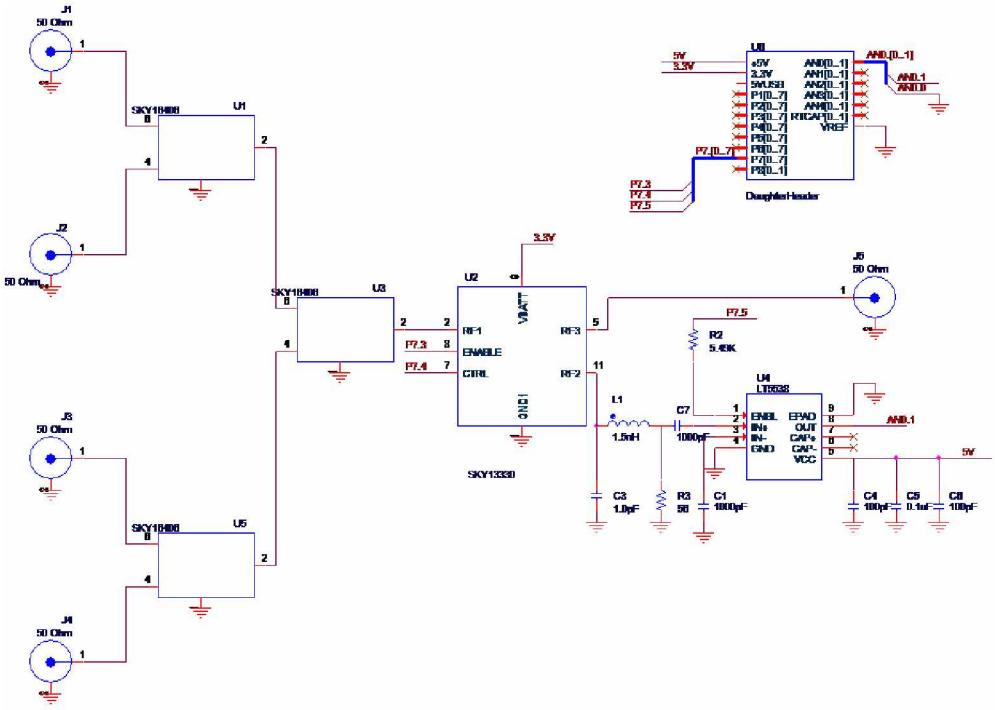


Figure C.3: Splitter board schematic.

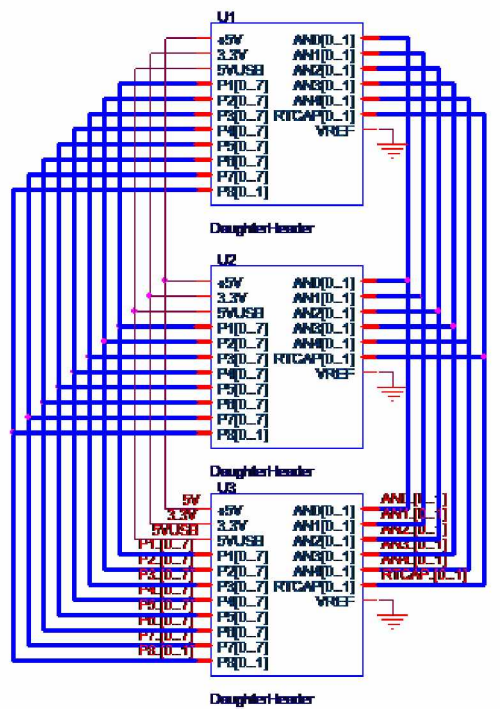


Figure C.4: Expander board schematic.

C.2 Rogers Space Grade Materials Memo

MICRO-GRAM

As a quarterly feature of the Micro-Gram, Arturo Aguayo, Jr. is doing a series of question and answer articles of special interest to you. He will feature questions most frequently asked by our customers and will respond to those questions. E-mail: art.aguayo@rogers-corp.com.

Question: **I'm in the process of designing circuitry for space applications. Which Rogers high frequency materials should I be evaluating?**

by Art Aguayo, Jr.

Answer: Rogers high frequency materials have been used in space-qualified hardware for many years for antennas, power dividers, transceivers, and power modules. The materials used range from RT/duroid® to TMM® grades, and currently, several programs have been working with the RO4000® family. There is no certification process for materials to be labeled *space qualified*; this qualification is done on finished hardware. A requirement placed on materials by NASA, is for them to have low outgassing resistance (expressed as % of original specimen mass) as measured by Total Mass Loss (TML), Collected Volatile Condensable Materials (CVCM) and Water Vapor Recovered (WVR). Materials used should have a TML less than 1% and both CVCM and WVR should be less than 0.1%. Table I presents these properties, along with the nominal dielectric constant and loss tangent at 10 GHz, for the various materials tested by NASA. It can be seen that many materials from Rogers meet NASA's requirement for outgassing resistance. This data, along with an extensive database of various materials can be found at the NASA website http://misspiggy.gsfc.nasa.gov/og/.

Table I. Outgassing Resistance of Various Rogers High Frequency Materials.

	RT/duroid 5870	RT/duroid 5880	RT/duroid 6010	RT/duroid 6002	TMM 3	TMM 10	RO4003
Composition	PTFE glass-microfiber	PTFE glass-microfiber	PTFE glass-microfiber ceramic filler	PTFE glass-microfiber ceramic filler	Thermoset polymer ceramic filler	Thermoset polymer ceramic filler	Thermoset polymer ceramic filler woven glass
Diel. Const.	2.33	2.2	10.2	2.94	3.27	9.2	3.38
Loss Tangent	0.0012	0.0009	0.0023	0.0012	0.0020	0.0023	0.0027
% TML	0.05	0.03	0.03	0.02	0.03	0.06	0.06
% CVCM	0	0	0	0.01	0	0	0
% WVR	0.04	0.02	0.02	0.01	0.03	0.04	0.02

Outgassing resistance is not the only property desired. Because of the harsh temperature environment, materials with stable electrical and mechanical performance should be selected. Stable dielectric constant versus temperature (TCEr) would minimize frequency shifting of circuitry while low Z-axis expansion provides for higher plated through hole reliability. Table II summarizes these properties for the materials presented in Table I.

Table II. TCε̃r and CTE of Various Rogers High Frequency Materials

	RT/duroid 5870	RT/duroid 5880	RT/duroid 6010	RT/duroid 6002	TMM 3	TMM 10	RO4003
TCε̃r, ppm/°C -50 to 150C	-115	-125	-425	+12	+39	-39	+40
CTE (Z), ppm/°C -0 to 100C	173	237	24	24	20	20	46

It can be noted from Table II that the better materials for space programs would be RT/duroid 6002, TMM-3, TMM-10, and RO4003 materials. RT/duroid 6002 and RO4003 materials would be better suited for applications requiring multilayer constructions, while TMM materials, because of their rigidity, are used as replacement for ceramic circuitry. RT/duroid 5870, 5880 and 6010 can and have been used in various space programs, but require additional design considerations to compensate for the larger TCEr and CTE. Rogers Corporation Microwave Materials Division has an extensive selection of high frequency materials; many of them ideal for space programs. Selection of the best material for these applications depends on the desired construction and functionality.

C.3 Related Publication: 2019 IEEE Aerospace Conference,

Retrodirective Phased Array Antenna for CubeSats

Justin Long, Denise Thorsen
Department of ECE
University of Alaska, Fairbanks
1764 Tanana Loop
Fairbanks, AK 99775
907-474-7613
jlong53@alaska.edu, dlthorsen@alaska.edu

Obadiah Kegege
Goddard Space Flight Center
Greenbelt, MD 20771
907-474-7613
obadiah.o.kegege@nasa.gov

***Abstract*— This paper discusses the design of a retrodirective phased array antenna for CubeSat applications. A phased array antenna can offer high gain and beamforming capabilities to small satellites. Retrodirective capabilities allow the communication system to autonomously determine the direction of an incoming signal without prior knowledge, and form the beam appropriately to achieve maximum gain. The end result is a compact high gain antenna without strict pointing requirements or deployables. Simulation result demonstrate that retrodirectivity provides array gain improvement across the entire scan range of the array rather than just the steer angle. Further simulations show that the benefits of retrodirectivity are more significant in larger arrays. An architecture is proposed using existing COTS parts, with a power added efficiency of 37% for a 5 W power output. Analytic analysis shows that due to DC power overhead, EIRP drops in larger arrays; arrays larger than 3x3 are impractical using the proposed architecture. A antenna design process is presented to meet the needs of the phased array, and a 2.2 GHz RHCP design with a 6 dBi gain and 2 dB axial ratio is shown. Test results highlight that errors in the phase shifters result in large errors in scanning accuracy. A method of calibration to correct for phase shift error is demonstrated.**

TABLE OF CONTENTS

1. INTRODUCTION

As of September 14, 2018 there had been 875 CubeSats launched, with 3,000 expected to be launched within 6 years [1]. The rising popularity of CubeSats has created demand for high performance and efficient miniaturized satellite technologies, and developers are rushing to create technologies for the CubeSat industry. As CubeSat technology improves, the communications downlink needs of CubeSats will increase accordingly. However, the number of CubeSat launches each year is accelerating and the radio spectrum available to CubeSats is becoming limited. Improvements in CubeSat communications technology is required.

978-1-5386-6854-2/19/$31.00 ©2019 IEEE

CubeSat Communications

The most direct way to improve downlink data rates is to increase the received power. The received power of a basic communication link is described by Equation 1.

$$P_{RX} = P_{TX} + G_{TX} - L_{TX} - L_{FS} + G_{RX} - L_{RX} - L_M \quad (1)$$

The power received P_{RX} is a function of the satellite transmit power P_{TX}, gain of the transmit antenna G_{TX}, total losses in the transmitter L_{TX}, free space losses L_{FS}, gain of the receive antenna G_{RX}, total losses of the receiver L_{RX} and other miscellaneous losses L_M. The goal of any communication system is to maximize the power received, as this is directly correlated to the information throughput.

This paper discusses the design of a Retrodirective Phased Array (RDA) antenna which autonomously maximizes gain in the direction of the target, thereby increasing the power received and improving data throughput.

High Gain Antenna (HGA) for CubeSats

Various designs have been proposed for high gain CubeSat antennas. The most popular High Gain Antenna (HGA) for missions involving 6U satellites beyond Low Earth Orbit (LEO) are reflectarrays (6U means six 1U units of approximately 10cm x 10cm x 10cm each). The Mars Cube One CubeSat [2] uses a reflect array design that offers $>$ 28 dB gain at 8.425 GHz. However, over 70% of launched or planned CubeSats are smaller than 6U, and the vast majority of CubeSats are in LEO [1]. Strict pointing requirements and size constraints make most HGA options, including the reflectarray, impractical for CubeSats smaller than 6U or ones in LEO.

Retrodirective Antenna Array (RDA)

The design proposed here is a RDA antenna using microstrip patch antennas. This antenna has the capability to be operated on a 1U CubeSat in LEO. The following sections summarize the theory of operation for retrodirective antennas, starting from basic antenna array theory.

Antenna Array Theory—An antenna array is created by combining the signal received or transmitted at multiple antennas. Depending on the position of the antennas and Direction of Arrival (DOA) of the signal, the signals may combine constructively or destructively.

Consider the situation shown in Figure 1. Two isotropic receivers, A and B are separated by a spacing S. If the signal reaches both receivers at the same time, such as in case (a), then the signals combine constructively. If the signal reaches

the antenna elements at different times, such as in case (b), then the signal may combine constructively (in-phase), or destructively (out-of-phase) depending on the signal phase shift. The phase shift is equal to the difference in distance that wave travels to reach each antenna. This distance is shown as D in Figure 1b.

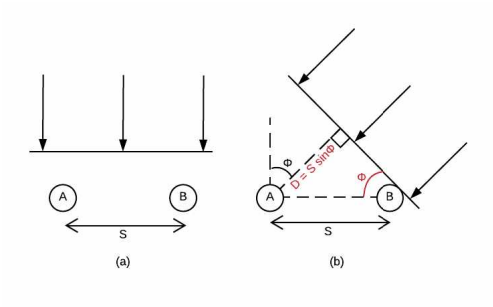


Figure 1. Isotropic elements receiving an incoming signal.

Signals can be forced to combine in-phase or out-of-phase by adding an intentional phase shift.

Phased Array Theory—By introducing an intentional phase shift to certain elements of an array, a phased array can be created. Phased arrays can be used to scan for interrogating signals, perform beamforming, and facilitate multiple access technology. The basic concept is to phase shift the signal at each element so that the signals combine in-phase.

An important note to make from Equation 4 is that the array pattern is a function of the pattern of the original element. This means that as a planar array is steered away from broadside, the gain decreases and the beam widens. Stutzman and Thiele [3] state that as a phased array is steered, the peak of the total array pattern follows the element pattern shape. Stutzman and Thiele [3] also provide an accompanying figure that shows an example of this trend, shown in Figure 2.

Beamforming—Utilizing the equations discussed, a phased array system can be designed to provide the unique advantages of beamforming. Beamforming is the process of controlling the phase shift of each element to form the antenna beam pattern as desired; a signal coming from any direction can be made to combine in-phase or out-of-phase. Desired signals can be constructively combined, and unwanted signals can be destructively combined. This process is also called electronic steering.

To receive maximum power, the signal must combine in-phase at each antenna element. The necessary phase shift ψ_{lobe} to achieve this is shown in Equation 2. This can also be phrased as the phase shift required to point the main beam in a chosen direction.

$$\psi_{lobe} = S\ sin(\phi) \tag{2}$$

To cancel a signal, the signal must combine out-of-phase at each antenna element. To achieve this, a 180° (or π) phase shift is added to Equation 2. The resulting equation for the phase shift ψ_{null} is shown in Equation 3. This can also be phrased as the phase shift necessary to point a null in the chosen direction.

$$\psi_{null} = S\ sin(\phi) + \pi \tag{3}$$

Retrodirectivity—Electronically steered phased arrays require prior knowledge of the targets location. Retrodirectivity offers the capability of steering highly directional arrays toward a target without prior knowledge of the angular position of the target. Kraust [4] shows the simplest example of a retrodirective antenna, the corner reflector. Incoming signals bounce off one wall of the corner, then the other wall, and end traveling back toward the interrogating source. This is the basic concept of retrodirectivity, where a signal is transmitted back in the direction of an interrogating source.

The first RDA proposed was the Van Atta reflector array. It operates similarly to the corner reflector in that it transmits an incoming signal back in the direction from which it came. However, the Van Atta array operates by interconnecting opposite elements through equal length transmission lines. The interrogating signal is received at the first element, phase shifted 180°through the transmission line, and retransmitted out the opposite element. This occurs in element pairs, effectively retransmitting the signal back in the original direction. The operation of the Van Atta array is shown in Figure 3.

To create a more robust and useful duplex communication system, the array must be capable of determining the direction of an incoming signal. The process of finding the DOA of signal is called scanning. Shiroma et al. [5] proposed a phase conjugation method in which the phase of the interrogating signal is measured in phase detectors at each element, allowing the DOA to be calculated. The beam of the transmitted signal is then directed back using phase shifters in the transmitting array. The drawback of this design is that the system is not capable of differentiating multiple interrogators, and may fail in the presence of multiple interrogators.

Akagi et al. [6] proposed a design for a power scanning RDA. The receive array uses phase shifters to scan the entire range of the array, and uses a power detecting circuit connected to a microcontroller to record the received power throughout the steering range. The microcontroller then controls the phase shifters to steer the beam toward the highest power level, effectively aiming at the strongest interrogating signal. The drawback of this design is that the system must be switched between scanning and transmitting/receiving, and therefore is not capable of constant receiving and transmitting. The advantages of this design over the phase detecting design is that this design can be used to communicate with multiple interrogators and power detecting circuitry is simpler and more common than phase detecting circuitry.

Iwami et al. [7] modified the power scanning technique by introducing a null scanning technique. The null scanning design uses a similar concept to the power scanning design, but scans for power minimums rather than power maximums. By finding the power minimum, the array is searching for the location at which the interrogating signal is being received completely out of phase. As shown in Equations 2 and 3, adding a 180°phase shift at this location will give the direction of the interrogating signal. This is advantageous because the null peak has a faster rate of change than the maximum power peak, resulting in more precise scanning. The drawback is that phase errors in the system may result in larger errors in DOA determination. The architecture shown by Iwami et al. [7] uses separate scanning and phase shifting arrays, so that constant transmission and reception is possible.

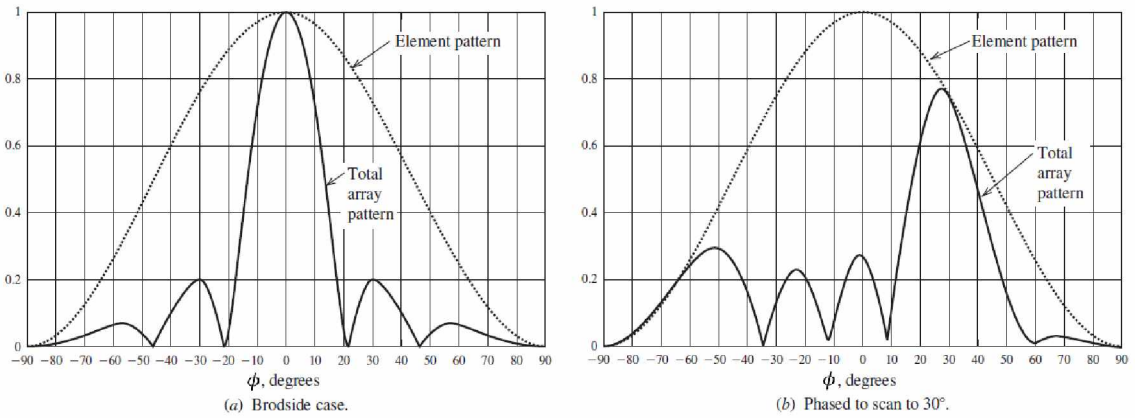


Figure 2. A four element linear phased array with 0.7λ element spacing steered to (a) broadside, (b) 30°. [3]

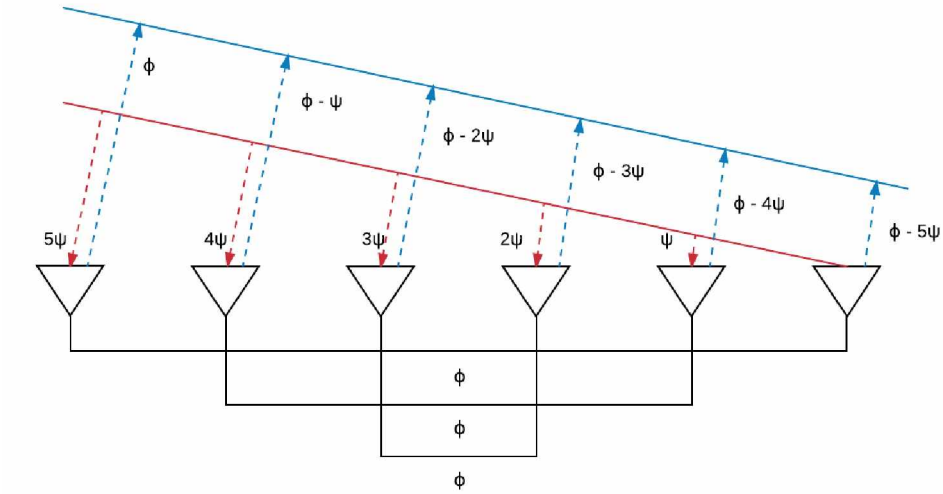


Figure 3. Operation of a Van Atta reflector array.

Table 1. Comparison of reflectarray and RDA technology.

Reflectarray	Retrodirective Phased Array
Feasible for 6U and larger CubeSats	Feasible for 1U CubeSats
Scalable to larger arrays	Limited scalability due to power overhead
JPL Marco provides 28 dBi gain at 8.5 GHz	1U face provides 21 dBi gain at 8.5 GHz
Requires deployable	No deployables
Accurate pointing requirements	Lenient pointing requirements

Comparison to other HGA—Klein et al. [8] showed that a four element steerable phased array fitting on the 1U face of a CubeSat can provide 13 dBi of gain at 2.4 GHz. Additionally, the design is steerable: the antenna can electronically steer its beam by controlling the phase of each element. This avoids the strict pointing requirements imposed on the satellite in other high gain antenna designs. The end result is an antenna that is feasible for CubeSats as small as 1U in orbits as low as LEO. Table 1 compares the advantages and disadvantages of the reflectarray and RDA technology.

Frequency

All designs, simulations, and tests within this paper were performed at 2.2 GHz. This is the S-Band frequency at which the National Aeronautics and Space Administration (NASA) missions operate. Specifically, satellite downlink occurs within 2200 MHz to 2400 MHz and satellite uplink occurs within 2025 MHz to 2120 MHz [9].

The 2.4 GHz frequency band is available for amateur radio use, and frequently used by CubeSats. The 2.4 GHz band has a large availability of Commercial-off-the-Shelf (COTS) products which could be used to create a low-cost and effective RDA system for CubeSats. Previous work such as that done by Klein et al. [8] used the 2.4 GHz frequency band.

Higher frequency bands within X-Band and Ka-Band are also feasible and attractive for RDA systems. X-Band is currently popular within military and large satellite applications, and COTS products are available that could be used to design an RDA system for CubeSats. Ka-band is becoming increasingly popular with NASA due to the large bandwidths available, but further technology development may be necessary before a miniaturized system can be developed for CubeSats.

2. SIMULATION

Analytic simulations were performed to highlight the effects phased arrays and the benefits of retrodirectivity. The equations shown in Stutzman and Thiele [3] were used to derive the analytic models for these simulations. Equation 4 shows a simplified method to approximate the performance of an antenna array. The effects of the array geometry are analyzed separately from the characteristics of the individual antennas element; the resulting effect of the geometry is represented by the Array Factor (AF). When the AF is known, it can be combined with the element pattern to approximate the final array performance. In Equation 4 $g_{ac}(\phi,\theta)$ is the average active-element pattern, $f(\phi,\theta)$ is the AF, and $F(\phi,\theta)$ is the array pattern. The active-element pattern is obtained by measuring a single element surrounded by passively terminated elements. Since the true active-element pattern was not measured for this project, these simulations do not account for mutual coupling. This approximation is more accurate for large arrays. [3]

$$F(\phi,\theta) = g_{ac}(\phi,\theta)\, f(\phi,\theta) \tag{4}$$

A planar array is used here as it will easily integrate into the CubeSat form factor. The AF for a uniformly excited planar array is analytically derived by knowing the number of rows M in the array, the number of columns in the array N, the spacing between elements of the array s (in wavelengths), the steering direction of the array $(\theta_{steer}, \phi_{steer})$, and the DOA $(\theta_{DOA}, \phi_{DOA})$. This is shown in Equation 5. [3]

$$f(\phi,\theta) = \sum_{m=1}^{M}\sum_{n=1}^{N} e^{j\alpha_{mn}} e^{j\zeta_{mn}}. \tag{5}$$

where:

$$\alpha_{mn} = -2\pi \cdot s \cdot [n \cdot sin(\phi_{steer}) \cdot cos(\theta_{steer}) + m \cdot sin(\phi_{steer}) \cdot sin(\theta_{steer})]$$

$$\zeta_{mn} = 2\pi \cdot s \cdot [n \cdot sin(\phi_{DOA}) \cdot cos(\theta_{DOA}) + m \cdot sin(\phi_{DOA}) \cdot sin(\theta_{DOA})]$$

Fixed Phased Array

Figure 4 shows the effects of a fixed (i.e. non-retrodirective) 2x2 array. For this simulation, a simple patch antenna was designed with the MATLAB antennas toolbox. With a fixed steering angle, the array pattern was calculated across the range of DOA angles using Equations 4 and 5. Figure 4 shows the single element antenna gain, the 2x2 AF, and the 2x2 fixed array gain. The Steering direction is $(0^\circ, 0^\circ)$. The X-axis of Figure 4 is the DOA zenith angle, at azimuth angle 0°. For these simulations, the zenith angle is plotted between $0^\circ$and $180^\circ$to show the front and back lobes, whereas Figure 2 only showed the front lobe.

The results show that the gain of a 2x2 array is 6 dB higher than the individual element gain in the steering direction. As the DOA moves away from the steering direction, the array gain drop sharply. This is because the signals at each element of the array combine out-of-phase; this can be seen by the decrease in AF. When the AF passes below 0 dB, the array pattern becomes less than the single element pattern. This simulation does not account for mutual coupling or phase error. Measurements by Klein et al. [8] showed an actual gain improvement of 5.5 dB.

Retrodirective Phased Array

Figure 5 shows the benefits that a retrodirective array can offer. The same process was followed from the Fixed Phased Array simulation, but the steering angle was set to always be equal to DOA angle. Figure 5 shows the single element gain, the 2x2 fixed array gain, and the 2x2 retrodirective array gain.

The results show that retrodirectivity ensures the AF is always max. The 6 dB gain in the steering direction of the fixed array occurs across the entire field of view for the retrodirective array. The beam pattern of the single element antenna is preserved and increased by 6 dB in every direction.

Scaling to Larger Array

Figure 5 shows the effects of larger size arrays. The Retrodirective Phased Array simulation was repeated for a 4x4 array. Figure 5 shows the single element gain, the 4x4 fixed array gain, and the 4x4 retrodirective array gain.

The results show the 4x4 array has a gain improvement of 12 dB over the single element antenna. The fixed array pattern shows three nulls, compared to one in the 2x2 array. In a fixed array, the larger array requires more accurate pointing because of a narrower beamwidth and increased number of nulls. The retrodirective array shows a constant 12 dB gain over the single element at all angles. This shows that larger array sizes give a greater benefit from retrodirective steering.

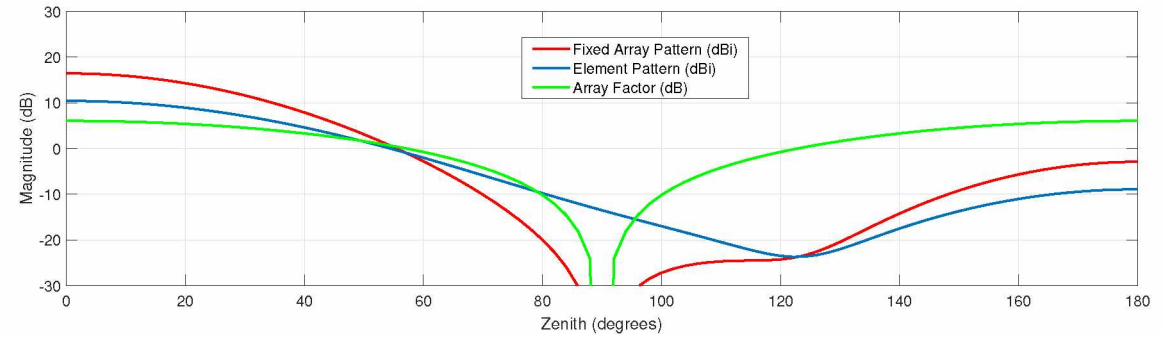


Figure 4. The base element pattern (blue), the array factor (green), and the resulting fixed array pattern (red) for a 2x2 array.

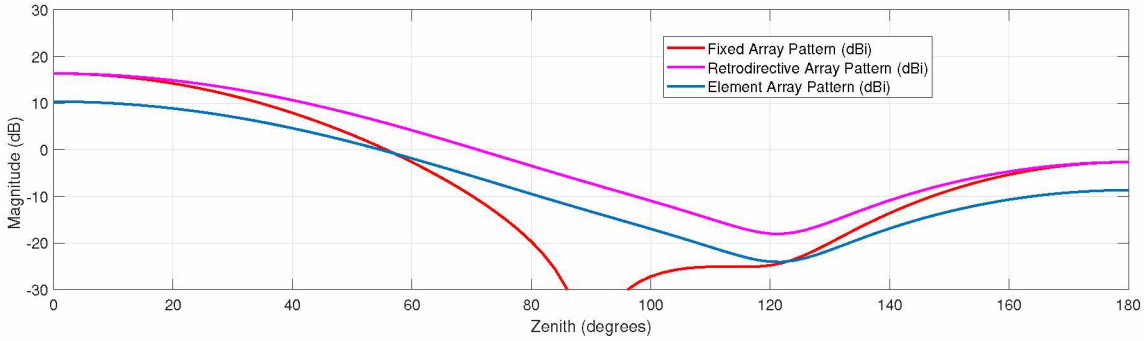


Figure 5. The base element pattern (blue), the fixed array pattern (red), and the retrodirective array pattern (magenta) for a 2x2 array.

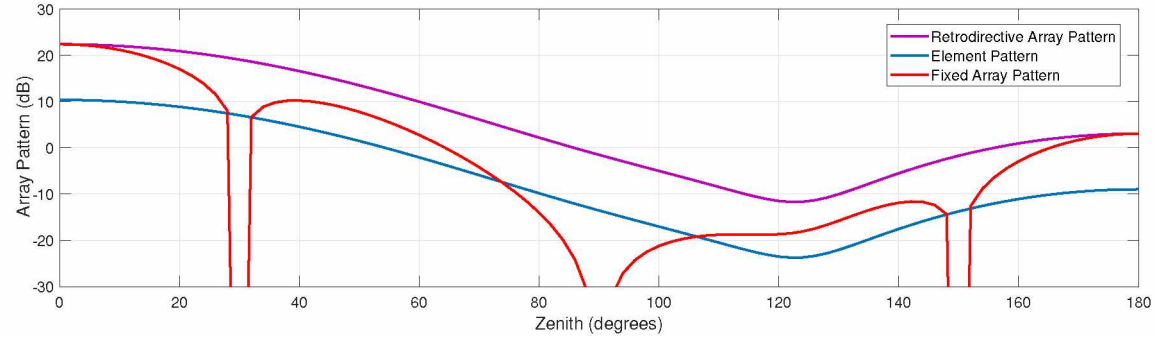


Figure 6. The base element pattern (blue), the fixed array pattern (red), and the retrodirective array pattern (magenta) for a 4x4 array.

3. Analysis and Design

Array Size

For the purpose of prototyping, this project only included the design of a 2x2 array designed for a 1U CubeSat. The design can be scaled to a much larger array. Table 2 shows the frequencies that have been considered for this technology, the number of elements that could fit on a 1U face, and the expected gain. Note that the gain value is based on the use of 5.5 dBi patch antenna.

Table 2. Example phased array sizes and characteristics for a 1U face.

Frequency (GHz)	Wavelength (cm)	# of Elements	Gain (dBi)
2.2	13.64	4	11.5
2.4	12.50	4	11.5
8	3.75	36	21.0
10	3.00	49	22.3
26	1.15	324	30.5

There are practical limits to the size of the array: (1) Each element has a per-element overhead, (2) the Power Added Efficiency (PAE) of the power amplifiers decrease as the input power into each channel reduces, and (3) larger arrays require more complicated feed networks with higher insertion loss. This means for a fixed power array, Effective Isotropically Radiated Power (EIRP) will start to diminish as the overhead power draw begins to overcome the benefits of increased gain.

Figure 7 shows an example of how the EIRP begins to diminish in larger arrays. Figure 7 assumes a 5.5 dBi element gain, and uses the power curves from the SKY66294-11 power amplifier. Klein et al. [8] performed a similar analysis, but assumed a constant efficiency at all power levels. This new analysis accounts for reduced at lower power levels. Based on new this analysis, the recommended maximum size for the antenna array between 5W and 10W is 9 elements (3x3).

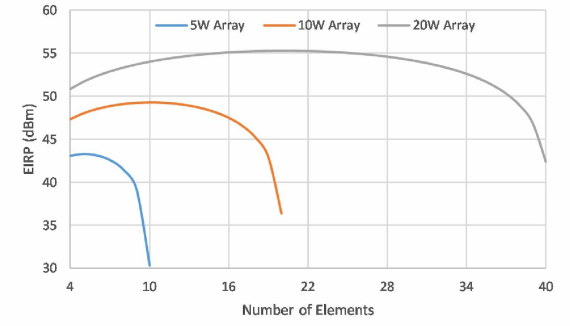


Figure 7. An example of diminishing returns for EIRP vs. array size.

Feed Network Design

There were four major considerations in the design of the feed network: (1) where to place the phase shifter, (2) how to implement Transmit/Receive (TR) switching, (3) how to select a power amplifier, and (4) where to place the power amplifier. The goal in the design of the feed network was to attain the desired output power while optimizing the hardware cost, insertion loss, and number of control signals.

To achieve a 5W output power and optimize the PAE, the SKY66294-11 power amplifier was selected. Based on the power curves for this amplifier, the feed network was designed as shown in Figure 8. The dual TR switches provide a single feed path while breaking out a transmit path for the power amplifier. Placing the power amplifier immediately before the antenna maximizes the potential output power of the system. The estimated EIRP for this design is 37% with the SKY66291-11 power amplifier and the PE44820 power amplifier.

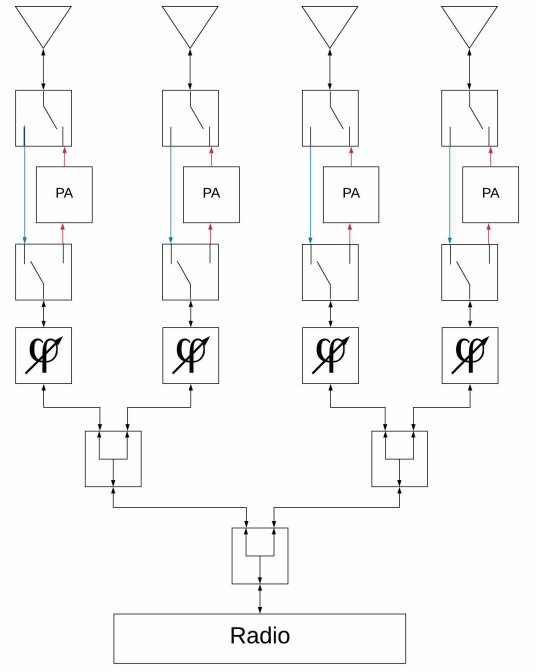


Figure 8. Feed network architecture.

An alternative design is to use a single power amplifier at the output of the radio. This would increase the PAE by decreasing DC power consumption, but the total power output of the system is then limited by the power handling capabilities of the phase shifters.

Antenna Element Design

As shown in Figure 2, the array pattern follows the element pattern shape as it is steered away from broadside. In other words, the field-of-view of the array is proportional to the beamwidth of the antenna elements. For this reason, it is desirable to have a wide beamwidth in the elements. To meet the needs of a typical CubeSat mission, this project is targeting a Half-Power Beamwidth (HPBW) of at least 100°. To minimize polarization losses, an Right-Hand Circularly Polarized (RHCP) antenna design has been selected.

There is a lack of COTS patch antennas available at the design

frequency. The design of custom antennas elements is costly, and beyond the budget of most CubeSat developers. For this reason, a design procedure for a low-cost custom square RHCP patch antenna is shown here. The design equations given by Bhartia et al. [10] are used to design the patch width and length for resonance. Lee et al. [11] provide a method of determining the feed location for circulalr polarization.

The width of the patch can be set to provide the desired characteristic impedance using Equation 6.

$$W = \sqrt{h * \lambda_d} * \left(ln\left(\frac{\lambda_d}{h}\right) - 1 \right) \tag{6}$$

where:

h: dielectric thickness

λ_d: resonant wavelength

The length of the patch must be adjusted to account for fringing field at the edges of the patch. Fringing along the edges $y = 0$ and $y = W$ can be accounted by calculating the effective dielectric constant, shown in Equation 7 [12].

$$\epsilon_{re}(u, \epsilon_r) = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2}\left(1 + \frac{10}{u}\right)^{-a(u)b(\epsilon_r)} \tag{7}$$

where:

$$a(u) = 1 + \frac{1}{49} ln\left(\frac{u^4 + \left(\frac{u}{52}\right)^2}{u^4 + 0.432}\right) + \frac{1}{18.7} ln\left(1 + \left(\frac{u}{18.1}\right)^3\right)$$

$$b(\epsilon_r) = 0.564\left(\frac{\epsilon_r - 0.9}{\epsilon_r + 0.3}\right)^{0.053}$$

$$u = \frac{W}{h}$$

Fringing along the edges $x = 0$ and $x = L$ can be accounted through the perturbation length, calculated with Equation 8 [13]. The calculation of the final length of the patch is shown in Equation 9.

$$\Delta l = \frac{h\zeta_1\zeta_3\zeta_5}{\zeta_4} \tag{8}$$

where:

$$\zeta_1 = 0.434907\,\frac{\epsilon_{re}^{0.81} + 0.26}{\epsilon_{re}^{0.81} - 0.189}\,\frac{u^{0.8544} + 0.236}{u^{0.8544} + 0.87}$$

$$\zeta_2 = 1 + \frac{u^{0.371}}{(2.358\epsilon_r + 1}$$

$$\zeta_3 = 1 + \frac{0.5274\, arctan(0.084u^{\frac{1.9413}{\zeta_2}})}{\epsilon_{re}^{0.9236}}$$

$$\zeta_4 = 1 + 0.0377\, arctan\left(0.07u^{1.456}\right)\left(6 - 5e^{0.036(1-\epsilon_r)}\right)$$

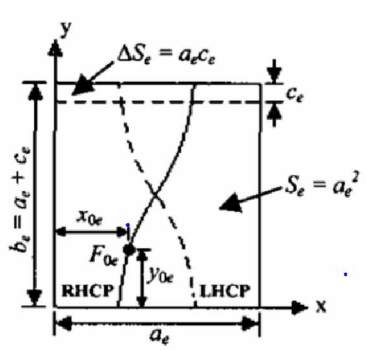


Figure 9. Diagram of the circularly polarized patch.[11]

$$\zeta_5 = 1 - 0.218e^{-7.5u}$$

$$L = \frac{c}{2f\sqrt{\epsilon_{re}}} - 2\Delta l \tag{9}$$

Lee et al. [11] showed that Equation 10 can be solved to zero to determine the feed location for circular polarization. Equation 10 is the equation for the magnitude of the axial ratio. The dimensions of the patch are shown in Figure 9. The polarization can be switched between RHCP and Left-Hand Circularly Polarized (LHCP) by choosing which side of the patch the feed location exists.

A MATLAB script was written to automate this design process and solve for the feed location. The OpenEMS FDTD simulation tool was used to simulate the performance of the antenna. Table 3 shows the inputs that were used with the script, the calculated dimensions, and the simulation results. Figure 10 shows the pattern and beamwidth of the element as simulated by OpenEMS.

$$\alpha^4(r-1) + \alpha^2(rM^2 - 1)\left(\frac{1}{Q^2} - 2\right) + rM^4 - 1 = 0 \tag{10}$$

where:

$$M = 1 + \frac{\Delta S_e}{S_e}$$

$$\alpha = \frac{2Lf\sqrt{\epsilon_{re}}}{c}$$

$$r = \left(\frac{cos\left(\frac{\pi y_0}{L}\right)}{cos\left(\frac{\pi x_e}{W}\right)}\right)^2$$

x_0: feed location along width

y_0: feed location along length

$$S_e = L\,W$$

$$\Delta S_e = 2\,\Delta l\,W$$

$$Q = \frac{S_e}{\Delta S_e}$$

Table 3. Example antenna parameters and simulated performance.

Inputs	
Freq	2.2 GHz
Dielectric	RT-Duroid 6010LM
Dielectric Constant	10.7
Dielectric Thickness	1.27 mm
Outputs	
Patch Size	[21.1, 18.1] mm
Feed Location	[3.65, 6.65] mm
Gain	6.13 dBi
Axial Ratio	2.06 dB
Beamwidth	100°

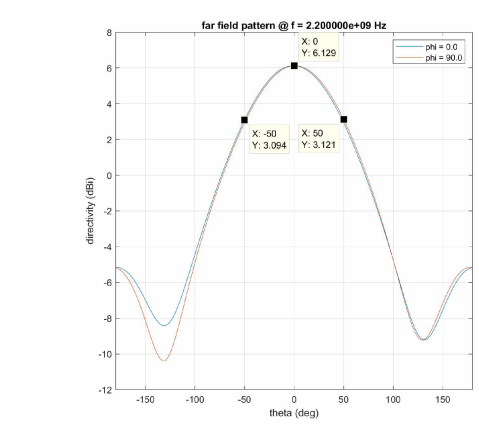


Figure 10. Simulated patch element pattern.

4. TESTING

Phase Shifter Performance

The PE44820 phase shifter performance was measured at the operating frequency of 2.2 GHz. Figure 11 shows the measurement of the phase error and insertion loss of one phase shifter across the phase shifter range. At each data point, 1000 measurements were made. Figure 11 shows the average of these measurements, and the standard deviation of the phase error across 1000 measurements. The test was run on eight phase shifters, and all eight showed similar performance. The performance was within the expectations of the data sheet. Note that the insertion loss measurements include board and connector losses.

Scanning Conceptual Testing

The scanning accuracy of the system was measured using the test setup shown in Figure 12. A signal generator was used to generate an unmodulated signal at 2.2 GHz. The signal was split into four channels using a Wilkinson divider and PE44820 phase shifters were programmed simulate the direction of arrival for the signal. A second set of PE44820 phase shifters were used to perform the scanning operation of the receiving array. The four channels were recombined and measured in a spectrum analyzer. The test was automated through GPIB and UART connections using a MATLAB script. The script scanned the receiving phase shifter array though all possible phase shifts, and measured the power at all phase shifts.

Table 4. Test results showing angle determination error

	DOA		Scanning		
	$\phi(°)$	$\theta(°)$	$\phi(°)$	$\theta(°)$	**Error(°)**
Uncalibrated	0	0	5.7	225	**5.7**
	30	60	11.5	191.3	**38.5**
	45	180	29.3	141.5	**27.4**
Calibrated	0	0	1.5	276.0	**1.5**
	5.1	56.9	8.9	70.3	**4.1**
	10.2	198.0	7.4	191.9	**2.9**
	15.4	14.1	15.6	8.4	**1.5**
	30.0	59.7	32.8	56.1	**3.4**
	45.9	234.4	42.0	234.4	**3.9**
	59.7	114.7	62.7	109.4	**5.5**

The phase error due to mechanical tolerances and uncertainty within the phase shifters was optimized through calibration. This calibration was performed by connecting each set of phase shifters, with the exact cabling used for testing, to a network analyzer and measuring the true discrete phase shifts for which the phase shifters were capable. There is additional error due to the amplitude unbalance across the phase range of the phase shifters, which was not calibrated for and is expected to be the primary remaining source of error in the DOA determination. The results for several DOA angles are shown in Table 4. The error value is a central angle error, calculated using Equation 11. Some non-calibrated measurements are also presented to show the necessity of the calibration.

$$Error = cos^{-1}[cos(\phi_{DOA}) \cdot cos(\phi_{scan}) + sin(\phi_{DOA}) \cdot sin(\phi_{scan}) \cdot cos(\theta_{DOA} - \theta_{scan})] \quad (11)$$

5. SUMMARY

An architecture was proposed for a Retrodirective Phased Array (RDA) antenna to meet CubeSat requirements. The benefits of a CubeSat RDA were discussed, specifically the size and leniency of imposed requirements including pointing and deployables. The benefits of the RDA were compared against other high gain antennas to show that the RDA can fulfill a need for high gain antennas in satellites as as small as 1U and in LEO. Analytic simulations highlighted the characteristics of retrodirectivity and the improvement over conventional antenna arrays. Analysis showed that creating large arrays (>3x3) is impractical due to DC power overhead. A specific feed network architecture was proposed based on existing COTS components, and it was shown that a PAE of 37% is expected. An antenna element design process was presented to meet the needs of the CubeSat RDA antenna; an example was shown at 2.2 GHz with a gain of 6 dBi and axial ratio of 2 dB. Test results were presented that show the need for calibration of the phased array system. Future work includes software development, fabrication and testing of the antenna elements, and system level testing of the RDA.

Figure 11. Phase shifter phase error (top), standard deviation of phase error (middle), and insertion loss (bottom).

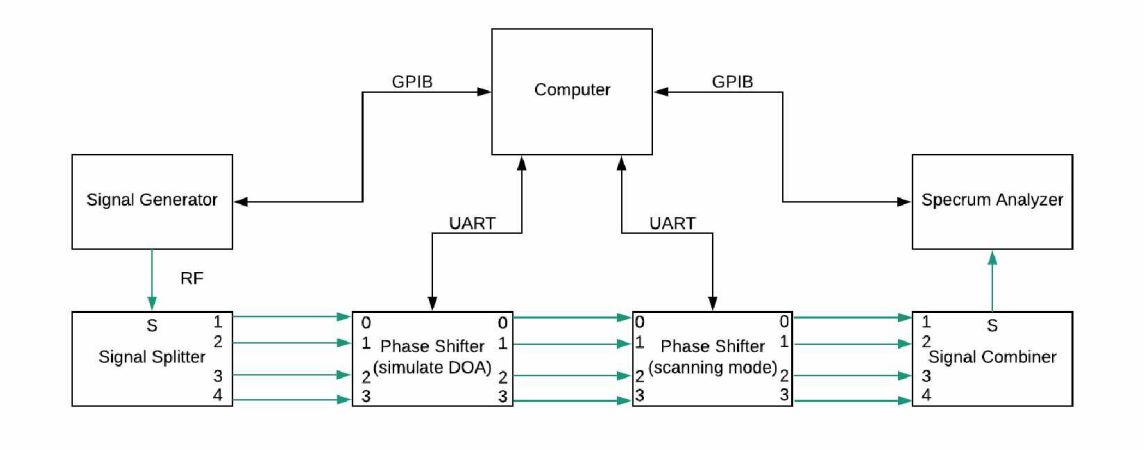


Figure 12. Test Setup of the RDA scanning test.

APPENDICES

A. COORDINATE SYSTEM

An azimuth-zenith coordinate system is used throughout this thesis. Azimuth angles are denoted by either the letters AZ or the lowercase Greek letter theta θ. Zenith angles are denoted by either the letters ZE or the lowercase Greek letter phi ϕ. Zenith angles (as opposed to elevation angles) are used so that the antenna elements can lay in the X-Y plane, and the broadside direction can be denoted by ($\phi = 0$). The elements exist in the first quadrant of the X-Y plane, with the feed of element 0 being at the origin. The coordinate system is graphically shown in Figure 13.

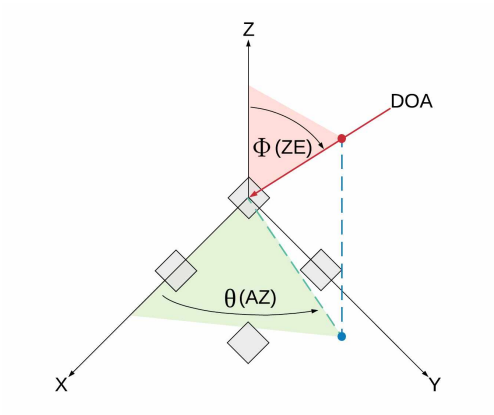


Figure 13. The coordinate system used throughout this paper.

B. ACRONYMS

COTS	Commercial-off-the-Shelf
LEO	Low Earth Orbit
RDA	Retrodirective Phased Array
HGA	High Gain Antenna
AF	Array Factor
ZE	Zenith Angle
AZ	Azimuth Angle
DOA	Direction of Arrival
UAF	University of Alaska Fairbanks
NASA	National Aeronautics and Space Administration
PAE	Power Added Efficiency
EIRP	Effective Isotropically Radiated Power
TR	Transmit/Receive
TX	Transmit
RX	Receive
HPBW	Half-Power Beamwidth
RHCP	Right-Hand Circularly Polarized
LHCP	Left-Hand Circularly Polarized

ACKNOWLEDGMENTS

This work was supported by a NASA Space Technology Research Fellowship.

The Alaska Space Grant Program, University of Alaska Fairbanks, and NASA Goddard Space Flight Center provided facilities and expertise for this work.

REFERENCES

[1] E. Kulu, "Nanosatellite Database." [Online]. Available: http://www.nanosats.eu

[2] R. E. Hodges, N. E. Chahat, D. J. Hoppe, and J. D. Vacchione, "The Mars Cube One deployable high gain antenna," in *2016 IEEE Antennas and Propagation Society International Symposium, APSURSI 2016 - Proceedings*, 2016.

[3] W. Stutzman and G. Thiele, *Antenna Theory and Design*, 3rd ed. Hoboken: John Wiley & Sons, Inc, 2012.

[4] J. D. Kraust, "The Corner-Reflector Antenna," *Proceedings of the I.R.E.*, pp. 513–519, 1940.

[5] G. S. Shiroma, R. Y. Miyamoto, and W. A. Shiroma, "A Full-Duplex Dual-Frequency Self-Steering Array Using Phase Detection and Phase Shifting," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 1, 2006.

[6] J. M. Akagi, A. Zamora, M. K. Watanabe, and W. A. Shiroma, "A Self-Steering Array Using Power Detection and Phase Shifting," *IEEE MTT-S International Microwave Symposium Digest*, pp. 1325–1328, 2008.

[7] R. T. Iwami, A. Zamora, T. F. Chun, M. K. Watanabe, and W. A. Shiroma, "A retrodirective null-scanning array," in *IEEE MTT-S International Microwave Symposium Digest*, 2010.

[8] J. Klein, J. Hawkins, and D. Thorsen, "Improving cubesat downlink capacity with active phased array antennas," in *IEEE Aerospace Conference Proceedings*, 2014.

[9] M. Harris, *Near Earth Network (NEN) Users' Guide*. National Aeronautics and Space Administration.

[10] P. Bhartia, I. Bahl, R. Garg, and A. Ittipiboon, *Microstrip Antenna Design Handbook*. Artech House Publishers, 2001.

[11] S. K. Lee, A. Sambell, E. Korolkiewicz, F. Ooi, and Y. Qin, "Design of a Circular Polarized Nearly Square Microstrip Patch Antenna with Offset Feed," *High Frequency Postgraduate Student Colloquium*, pp. 61–66, 2004.

[12] E. Hammerstad and O. Jensen, "Accurate Models for Microstrip Computer-Aided Design," *IEEE MTT-S International Microwave symposium Digest*, pp. 407–409, 1980.

[13] M. Kirschning, R. Jansen, and N. Koster, "Accurate Model for Open End Effect of Microstrip Lines," *Electronics Letters*, vol. 17, no. 3, pp. 123–125, 1981.

BIOGRAPHY

Justin Long *received his B.S. degree in electrical and computer engineering from UAF in 2017 and is currently pursuing a M.S. degree in electrical and computer engineering from the University of Alaska Fairbanks. He is currently an engineering Pathways intern at Goddard Space Flight Center.*

Dr. Denise Thorsen *received her B.S. (1985), M.S. (1991) and Ph.D. (1996) degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign. She is a Professor in Electrical and Computer Engineering at the University of Alaska Fairbanks.*

Dr. Obadiah Kegege *received his BS in Control and Instrumentation Electronics from the University of Houston, MSEE from the University of Texas Pan American and a PhD from the University of Arkansas. He currently serves as an engineering manager supporting the Near Earth Network Projects in the Exploration and Space Communications Projects Division, NASA GSFC.*