12-2019

Radio direction finding using pseudo-Doppler for UAV-based Animal Tracking

Anup Karki
Grand Valley State University

Radio direction finding using pseudo-Doppler for UAV-based Animal Tracking

Anup Karki

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Engineering

School of Engineering

December 2019

Acknowledgments

Thank you all my friends for supporting me throughout my thesis.

Abstract

Radio Direction Finding (RDF) is commonly used for low cost tracking and navigation systems. However, for a low cost application and mobility, the design constraints are highly limited. Pseudo Doppler (PD) can improve RDF capabilities without being cost prohibitive. This work entails the analysis of PD RDF and its potential use for Unmanned Aerial Vehicles (UAV) that are currently employed in wildlife research animal tracking. PD is based on the doppler effect or doppler shift. The doppler effect works like a frequency modulator that increases or decreases the observed frequency depending on whether a signal source is approaching or receding the detector. Noting this doppler shift can be used to improve the accuracy of the RDF algorithm. Normally, deploying the doppler method would require the physical rotation of an antenna at a relatively high frequency. Instead, PD employs a grid of electrically isolated antennas and then digitally samples each of the antennas in a sequential order, mimicking the action of mechanical rotation. Thus, PD removes the requirement to physically rotate the antenna yet provides the increased accuracy available with sensing the doppler effect.

PD can be implemented at low cost using an off-the-shelf Software Defined Radio (SDR) and simple monopole antennas. This work includes the design and implementation of a PD RDF on an SDR for deployment on an existing UAV platform. The expected improved performance of the PD system is to be tested on the existing UAV platform and compared against traditional tracking methods (such as a single Yagi antenna).

Keywords: Radio Direction Finding, Pseudo-Doppler, Software Defined Radio

Contents

List of Figures

List of Tables

Acronyms

ADC Analog to Digital Conversion. 20

AOA Angle of Arrival. i, ii, iv, 1, 3, 6, 8, 9, 13–16, 18, 25, 26, 28–34, 36, 38

DFT Discreted Fourier Transform. 14

ESD Energy Spectral Density. 33

FFT Fast Fourier Transform. i, 14–16, 34, 36

FIR Finite Impulse Response. 12, 24

FM Frequency Modulated. 12, 13

FPGA Field Programmable Gate Array. 20, 21, 36, 60

GPIO General Purpose Input Output. 20, 36

GPS Geo Positioning System. 2

GRC GNU Radio Companion. iii, iv, 23, 24, 45, 47

GUI Graphic User Interface. i, 23–25, 28, 47

IIR Infinite Impulse Response. ii, iv, 17, 18, 37

LNA Low Noise Amplifier. 22

MUSIC MUltiple SIgnal Classification. i, ii, 3, 31–33, 36

PLL Phase Locked Loop. 9

RDF Radio Direction Finding. ii, iv, 1, 2, 9–11, 18, 19, 26, 27, 29, 31–33, 35–37

RF Radio Frequency. 9, 60

RMS Root Mean Square. 10

SDR Software Defined Radio. iii, 9, 12, 19, 20, 22, 23, 26, 27, 32, 36, 60

SNR Signal to Noise Ratio. iv, 9, 15

UAV Unmanned Aerial Vehicle. 1, 2, 9–11, 36

UDP Uninterrupted Data Protocol. 23, 24, 47

USB Universal Serial Bus. 23

VHF Very High Frequency. iv, 1, 2, 11, 19

1 Introduction

A Radio Direction Finding (RDF) system determines the azimuth direction, also known as Angle of Arrival(AOA) of a remotely located transmitter. RDF is widely used in radio navigation system especially with boats and aircraft. RDF systems can be used with different radio sources; the system is limited by its antenna size. Radio sources with long wavelengths requires large antennas while short wavelengths permit an RDF system with smaller antenna sizes.

Since the first use of directional antennas, AOA estimation had been widely used. A direction-finding system typically includes multiple antennas forming an antenna array.The antenna array could form a directional antenna. When antenna array is implemented as directional antenna each element works to increase the directivity of the directional antenna. The whole antenna array acts as a single antenna and is studied as a single antenna. On the other hand, each antenna in the array can function individually i.e. each antenna has its own radiation pattern and directivity. If multiple antennas are processed at the same time, then it is multi-channel method, though each antenna has its own dedicated channel. If the signal received though a single channel then it is called single channel method. In single channel method, each antenna is selectively switched i.e. only one antenna is active at a time.

Both the multi-channel and single channel methods have been employed. Multi-channel techniques such as Estimation of Signal Parameters via Rotational Invariance Techniques(ESPRIT) [1] and MUltiple SIgnal Classification(MUSIC) [2] [3] utilize varying phases on the different antennas to estimate the direction. These techniques require at least three coherent receivers in which phases are compared. A local oscillator, matched filter and adjustable gain would be required for coherency of these receivers. The match must be maintained over frequency, temperature and time. Periodic calibration can be employed to maintain the match but the receivers

are often bulky and power consuming. Hence, these techniques are not suitable for mobile applications such as hand-held devices or UAV.

The single channel method is based on an interferometer [4]. The single channel method offers more simplicity, mobility and cost effectiveness. There are two distinct types of single channel methods: an amplitude-based system and a phase-based system. An amplitude-based system estimates the direction by analyzing the amplitudes of the output voltage of the antenna array elements. The Watson-Watt technique is such an example [5].

The phase-based system estimates the direction by analyzing the relative phases of the antenna array elements, which are unique for each angle of arrival. Phase-based systems employ a switching mechanism to switch between the antennas. The pseudo Doppler method is an example of the phased-based system [6] [7].

Pseudo Doppler offers simplicity, mobility and cost effectiveness. A pseudo Doppler utilizes antenna array structure in which the antennas are sequentially switched i.e. only one antenna is active at a time. Thus, making it useful for low cost application. This thesis aims at developing a low cost pseudo Doppler based RDF system, that can be deployed on a UAV to track animal transmitters.

1.1 Wildlife radio tracking

Small radio transmitters are frequently used by researchers to study and observe migration patterns and other behaviors. Beacons used for animal tracking are extremely diverse and cover a large range of sizes and transmitter output power [8]. Figure 1 shows one example of such transmitter on a wild dog. The best option for attachment option depends on the body type, shape, size and lifestyle of the animal species.

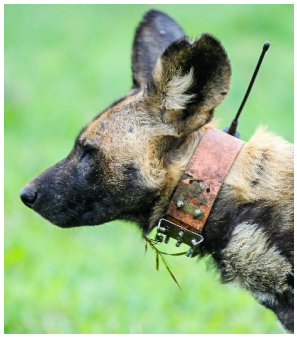


Figure 1: An external VHF collar on a wild dog [9]

Typical radio frequencies for wildlife tracking ranges from 148-152 MHz [10]. The VHF band is most commonly used for these applications and a typical receiver system uses a directional antenna.

The process of tracking VHF radio equipped animals usually includes a full ground search to locate the direction of strongest signal using triangulation methods. Over the past decade, UAVs have increasingly been used to extend the range of signal reception aiding in triangulation methods, and also provide a safer and cost-efficient alternative to full-scale aircraft for wildlife researchers.

1.2 Research goals

The research was done to meet the following objectives:

- Identify potential improvements in Radio Direction Finding capabilities through the use of pseudo-Doppler technique.
- Simulate the pseudo-Doppler Radio Direction Finding and implement on a hardware platform.
- Perform comparison of pseudo-Doppler Radio Direction evaluating performance against the performance of standard methods such as directional antennas.

1.3 Document Structure

This dissertation is organized as follows:

Chapter 1 introduces the context of the research and provides the goals that were set out for the research.

Chapter 2 discusses the theory behind the Doppler based RDF. The Doppler technique is mathematically expressed in this chapter. This chapter then explains the underlying principle behind the pseudo Doppler method.

Chapter 3 provides a literature review on the subject. Previous methods and review of those methods are included in this chapter.

Chapter 4 design for the implementation of pseudo Doppler RDF system. It discusses the design of the various components in the algorithm.

Chapter 5 describes the hardware and software implementation of the pseudo Doppler system. It includes the hardware description and the methodology to implement the system.

Chapter 6 describes validation of the system built and checks if the algorithm can be implemented for AOA estimation.

Chapter 7 discusses the result obtained from the system. It also compares the pseudo Doppler with systems such as MUSIC and directional antennas.

Chapter 8 concludes the dissertation and provides recommendations for the future work.

2 Doppler Radio Direction Finding System

A Doppler RDF system is based on the relative motion between transmitter and receiver. The relative motion can be produced by mounting the antennas on a rotating turntable (True Doppler) or by electronically switching between antennas (Pseudo Doppler).

2.1 True Doppler

2.1.1 Finding AOA

Doppler direction-finding systems were first introduced in 1947 [11]. The bearing angle in this system is obtained from phase comparison between a reference signal and a Doppler induced signal.

The Doppler direction finding system is based on the change of frequency between relatively moving transmitter and receiver, known as Doppler shift. Doppler shift is the frequency shift observed in the incoming signal due to the motion of the moving object and compression/expansion of the waveform. For example, such effect is observed when a moving vehicle engages a siren. Due to the motion of the vehicle, an increase or decrease in the siren's frequency is observed based on the vehicles approach or withdrawal, respectively. If the vehicle is approaching the observer, the siren appears to be at a higher frequency; conversely, if the vehicle is receding, it appears lower. The Doppler effect is shown in Figure 2.

A similar phenomenon is observed when an antenna is moving towards or away from locally stationery transmitting source. Signal received from the antenna moving towards the transmitter source appears to be at higher frequency than the actual transmission. Similarly, the signal received from the antenna moving away from

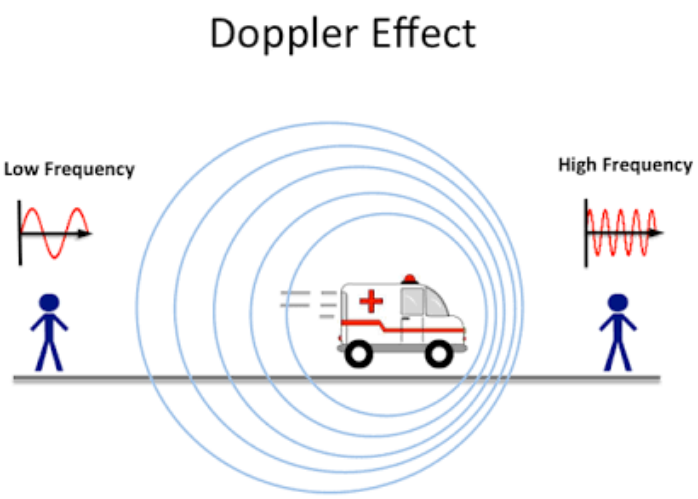
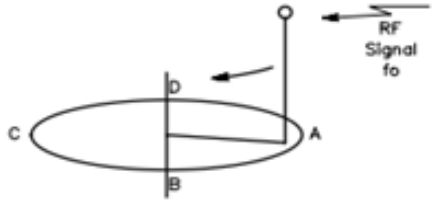


Figure 2: Change in frequency of the siren based on approaching or withdrawal end [12].

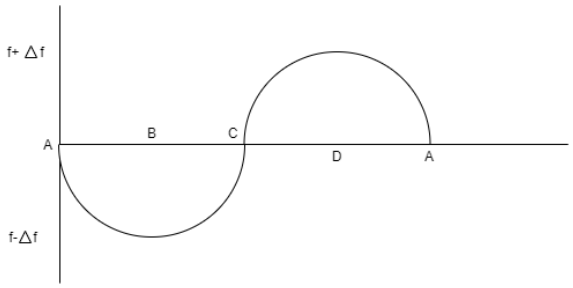
the transmitting source appears to be lower in frequency. If the antenna is moving in a circular pattern as shown in figure 3(a), the receiving signal would produce different shifts based on the position of antenna in the circle. As shown in the figure 3(a), the positions A and C are the nearest and farthest points. In the static case, frequency at A and C equals the frequency of transmission as the signal is not moving away or towards the antenna. Conversely, if the antenna is rotating in clockwise direction, the observed frequency of the received signal decreases as the antenna moves from point A to B and B to C. Maximum frequency deviation occurs as the antenna passes through point B. Then frequency will increase as the antenna moves from point C to D, maximum deviation occurring at D and then decreases from D to A. This process is shown in figure 3(b).

The signal received by a omnidirectional antenna which is rotating at ω rad/sec as shown in figure 4 on a turntable of radius r is given by [14]

$$s(t) = f(x, y, t) = Acos(2\pi f_c t + \beta r cos(\omega t - \theta))), \tag{1}$$



(a) A single antenna receiving an RF signal from the distance source. Antenna potions A,B,C,D are due to rotation of the turntable where the antenna is mounted [13]



(b) The resulting Doppler shift produced due to the rotation of the antenna [13]

Figure 3: Doppler effect due to rotation of the antenna

where $\beta = \frac{2\pi}{\lambda_c}$, f_c is the signal's frequency, λ_c is the signal wavelength and θ is the angle of arrival.

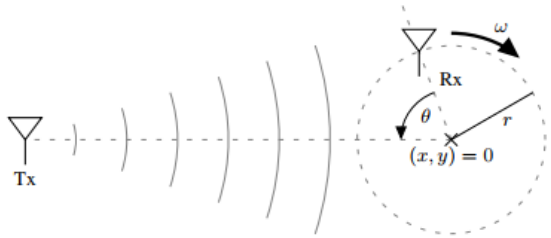


Figure 4: Doppler method with a receiver antenna rotating with angular velocity ω on a circle with radius r [14]

Equation (1) is similar to the standard PM modulated signal [15] as

$$x_{\text{pm}}(t) = A_c cos(2\pi f_c t + \hat{\beta} cos(2\pi f_m t)), \tag{2}$$

where $\hat{\beta}$ is the modulation index, f_m is the modulating sinusoid, Δf is the frequency deviation and f_c is the carrier frequency.

Comparing equations (1) with (2), it can be inferred that the signal received due to circular motion of the antenna undergoes single tone phase modulation. Thus, the modulation index is (βr) and the single tone sinusoid due to rotation of the antenna

has frequency $\omega/2\pi$. Here, the modulating signal $cos(\omega t - \theta)$ contains the AOA as phase shift θ. Comparing with a known reference $cos(\omega t)$ would give the AOA by inspection. The instantaneous frequency of (1) is given as

$$\omega_i(t) = \frac{d(\phi(t))}{dt} = 2\pi f_c - \beta\omega r sin(\omega t - \theta) \tag{3}$$

Filtering out the DC component in (3), the obtained Doppler signal is

$$S_D(t) = \beta r\omega sin(\omega t - \theta). \tag{4}$$

This signal can then be compared with the reference signal given as

$$S_r(t) = sin(\omega t) \tag{5}$$

whose frequency is ω in order to infer the phase.

Thus, the AOA (θ) can be calculated as the phase difference between Doppler Signal $S_D(t)$ and reference signal $S_r(t)$

2.1.2 Rotational Rate

Interpreting equation (1) as an FM signal, modulation index is given by $\hat{\beta}$. Here, from (1) and (2), the modulation index is given by ($\beta r = \frac{2\pi}{\lambda_c} r$). If Δf is the frequency deviation, then modulation index can be re-written as

$$\frac{2\pi}{\lambda_c} r = \frac{\Delta f}{\omega/2\pi}. \tag{6}$$

The expression (6) can be rearranged to give the angular frequency

$$\omega = \frac{\Delta f \lambda_c}{r}. \tag{7}$$

Equation 7 can be re-written in terms of the carrier frequency as

$$\omega = \frac{\Delta f c}{r f_c}, \tag{8}$$

where c is the velocity of radio signal in vacuum ($c = 3 \times 10^8 m/s$).

In present case, for example ($f_c = 150.580 MHz$) , to generate a frequency shift of 500 Hz, a turntable of 10 cm needs to have angular velocity of 9961.5 revolutions per second. A rotation of such high rate is nearly impossible with mechanically rotating antenna turntable. Hence, true Doppler shift method is impractical.

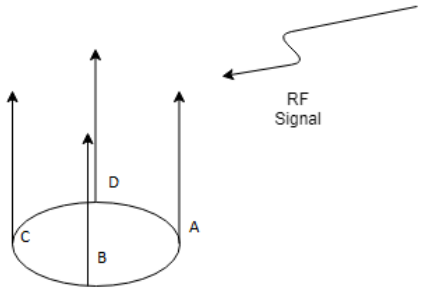
An alternative to the true Doppler method is the Pseudo Doppler method. Here, instead of rotating the antenna mechanically, the array of antenna elements are electronically switched to mimic rotation about fixed antenna positions.

2.2 Pseudo Doppler Method

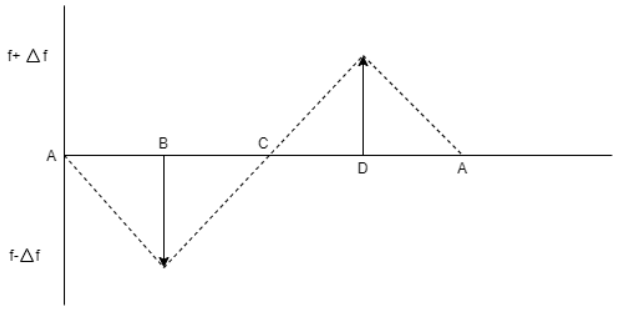
In 1978 a method to electrically spin the antenna was devised [16]. This project, denoted DoppleScAnt used quarter-wavelength vertical antennas arranged in a circular pattern. Only one antenna was electrically selected. By controlling the order in which the antennas are selected, the DoppleScAnt was able to emulate the single antenna rotating in a circle. Over the years, many modifications to the original design have been made. Generally a set of 4 or 8 antennas are used [17].

The Pseudo Doppler method exhibits a similar response to that of the true Doppler method. Due to antenna switching, only one antenna is activated for a given time. Assume, the antennas are switched in a clockwise fashion (A-B-C-D-A) every T_s time per step. Hence, instead of continuous curve as shown in Figure 3(b), abrupt changes at switching time is observed. Similar to the Doppler shift, the antenna positions A,B,C,D would detect the approaching or withdrawal signal. A signal com-

ing from the direction A would produce no shift at position A, while B produces minimum shift for B being the withdrawal side for the signal and D would produce the maximum shift due to approaching position of the signal as shown in 5(b). The phase jumps due to the switching occur every time in the receiver antennas.



(a) A possible arrangement of the stationery antennas for pseudo Doppler effect



(b) The output pulses received due to pseudo Doppler

Figure 5: Pseudo Doppler arrangement and resulting Doppler shifts

Assume there are N_a monopole antenna on a circular array of radius 'r', if the receiver switches from ith antenna to the $(i+1)th$ antenna every T_s seconds then phase imposed by each antenna is given by [4]

$$\frac{2\pi r}{\lambda_c} cos(\frac{2\pi i}{N_a} - \theta). \tag{9}$$

Hence, the time-varying phase shift imposed

$$\Theta_s(t) = \frac{2\pi r}{\lambda_c} cos(\frac{2\pi}{N_a} \sum_{n=1}^{\infty} (u(t - nT_s) - \theta)), \tag{10}$$

where $u(t)$ is the unit step function and θ is the AOA. Thus, the received signal is

$$s(t) = Acos(2\pi f_c t + \frac{2\pi r}{\lambda} cos(\frac{2\pi}{N_a} \sum_{n=0}^{\infty} u(t - nT_s) - \theta)), \tag{11}$$

The instantaneous frequency of (11) is given by [4]

$$f(t) = \omega_c - \frac{2\pi r}{\lambda_c}\frac{2}{N_a}\sum_{n=0}^{\infty} sin(\frac{2\pi}{N_a}(n-\frac{1}{2}) - \theta)\delta(t - nT_s) \tag{12}$$

It can be observed that (12) represents a sampled signal with sampling period T_s. After filtering out the DC component and sampling the discrete-time version of (12) is written as

$$f[n] = -\frac{2\pi r}{\lambda_c}\frac{2}{N_a}\sin(\frac{2\pi}{N_a}(n-\frac{1}{2}) - \theta) \tag{13}$$

From (13), clearly the differentiated signal is not continuous in nature in comparison to its true Doppler counterpart (4). As the antenna switches from one antenna to another it will be sampled $T_s * N_a$ times per rotation. Hence, the frequency of the pseudo Doppler signal is $\frac{1}{T_s * N_a}$. Thus, the reference signal must also use the frequency of $\frac{1}{T_s * N_a}$. The phase comparison of equation (13) with this reference signal gives the AOA (θ). Pseudo Doppler differs from true Doppler as:

- Signal is received by an antenna array rather than a single antenna
- Antennas in the array are switched periodically by an external switch.

3 Literature Review

Pseudo-Doppler RDF have gained popularity due to its simplicity and low-cost application. Initially used for "fox-hunting" or T-hunting [18], it has found usage in HAM radio projects [16], wildlife tracking [19], vehicle applications [20] [21] and network design applications [22]. While pseudo Doppler offers simplicity for some good accuracy, it suffers from several drawbacks [7] such as multi-path, geometrical errors and group delay due to the receiver. Despite the drawbacks it offers significant benefits in terms of weight, cost, complexity and power consumption.

Pseudo Doppler was initially popularized by Amateur Ham radio operators. It has found some usage in wildlife tracking as well [19]. However, the actual accuracy of the these systems and nature of the transmitter signal is missing. Directional antennas are popular choice for wildlife tracking due to their simplicity but have limited accuracy due to their scan range and directivity. Directional antennas are usually equipped with external compass and requires scan (either manually or through external hardware). Pseudo Doppler on the other hand is omnidirectional and can provide an angle estimate within 5 degrees under good working conditions [23]. This literature review looks at the different approaches and implementations of the pseudo Doppler and its potential use in wildlife tracking.

Since its introduction in [16], several methods have been developed to increase the accuracy of the system. In [6] a four antenna based system is presented where adaptive algorithm is used to estimate antenna phases over each iterations. These antenna phases were processed to give the AOA estimate. An adaptive approach to pseudo Doppler algorithm which can be implemented on a micro-controller, however accuracy of the system was not presented.

In [4] a Phase Locked Loop (PLL) (PLL) based algorithm was discussed. An antenna array of 8 elements with equal number of PLLs was designed. Under labora-

tory condition, at Transmitter(Tx)-Receiver(Rx) distance of 1.1 m, the system provided an accuracy within 0.3°-4°. It is shown that the PLL algorithm is better than the pseudo Doppler algorithm. The algorithm was performed better than pseudo Doppler under low SNRs and multi-path signals. However, the system was based on multiple PLLs and 8 antenna array elements. Such PLL are needed to be synchronized and thus increasing the complexity to implement. An array of 8 antennas would increase the radius of the array making it unsuitable to mount on a UAV.

A software based approach is presented by [24]. The pseudo Doppler algorithm is implemented with GNU radio. GNU radio provides software interface to interact with SDRs. Since the algorithm is implemented on software, it reduces the errors due to hardware components. A software based implementation reduces the complexity and provides ease of debugging the system. Hence, a software based approach to implement the algorithm was taken.

OperaCake platform was introduced by [25]. OperaCake is an RF switch that can interface with HackRF. It is capable of switching the antenna at much higher rate 625kHz, thus providing a much higher sampling frequency. OperaCake's hardware description is open source.

Pseudo Doppler's performance is not just based on its algorithm but also on antenna size and number of antenna elements. A typical pseudo Doppler antenna array has a radius size of less than the half the wavelength to avoid multi-path phase ambiguity. [26] presents a comparative study of three different array sizes of 0.2λ, 0.14λ and 0.25λ where at 0.2λ yields an accuracy of 7°at 2.5GHz. [26] recommends 0.2λ spacing.

In [27], an analysis of different numbers of antennas in an array is presented. In comparing the accuracy, with antennas 4,5,6,7 and 8, greater accuracy was observed when the number of antenna was 8. Although increasing the number of antennas

increase the performance of the system, an increase in antenna array size makes it bulkier.

A comparative study of different direction finding algorithms is presented by [28][17]. Some of the drawbacks suffered by pseudo Doppler are:

- At least one antenna cycle scanning is required to calculate the bearing;
- Needs longer duty cycle (>50%) to perform properly.

Pseudo Doppler has been successfully implemented both on hardware based application and software. Despite its drawbacks it can be a good alternative in wildlife tracking. Traditionally, it is implemented only with long duty cycle signals. However, a different approach to signal identification can make the algorithm work on lower duty cycle signals. This approach is described in upcoming section 4.

The thesis aims at implementing the pseudo-Doppler RDF that can work with low duty cycle signals ($< 2\%$). The implementation includes signal detection method that would provide fast and confident signal validation method. Another aim of this thesis is to reduce the surface dimension of the antenna array (0.05 λ) so that it can be mounted on a UAV. The thesis analyzes the performance of the signal validation method with antenna array of 0.05 λ.

4 Design

The thesis aimed at developing a pseudo Doppler based RDF system that can be mounted on a UAV for animal tracking. Typically, wildlife tracking uses VHF frequencies of 148-152 MHz [10]. Often in the field of animal tracking, the radio signals are pulsed in time with low duty cycles (1%-2%). Hence, the RDF should be able to work on the desired signal and have a small radius to be mounted on the UAV. Thus, pseudo Doppler RDF system was designed with following parameters:

- Signal Frequency $'f_c'$ 150.580 MHz
- Signal Duty Cycle (<2%)
- Antenna Radius 'r' 10cm ($r/\lambda_c = 0.05$)
- Number of antenna 'N_a' 4
- Sampling Frequency 'fs' 250 kHz
- Rotation rate 'f_m' 4882.8215 Hz

Though the target of the system was to make a working prototype for signal duty cycle of less than 2%, the system was initially tested with 100% to validate the algorithm. Subsequently, the system's performance was tested for signals of different duty cycles. The performance of the systems was tested for duty cycle from 80% 1%. These tests showed the algorithm's accuracy at incoming signal of different duty cycles as shown in section 4.5.1 and 7.1. Although, the system is implemented for small radius antenna array, it can be used also for larger antenna array but the radius has to be smaller than half the wavelength of the incoming frequency.

The block diagram of the pulsed pseudo Doppler RDF system is shown in Figure 6. The signal is received by the antenna array. The antenna array is controlled by RF

switch. Thus, only one antenna is active at a time. The switching of the antenna is controlled through a circuit whose control signals are derived from the receiver. It is necessary to derive the control signals from the receiver so that receiver and antenna switching would be coherent. Otherwise, the received signal would drift with respect to the reference signal. The received signal is then decimated by a factor of 'D'. The signal is FM demodulated yielding the single tone sinusoid signal. The band pass filter then removes interference to generate a smooth signal. The filtered signal is then compared with a known reference signal. The angle estimate is then corrected with respect to the calibrated angle estimate.

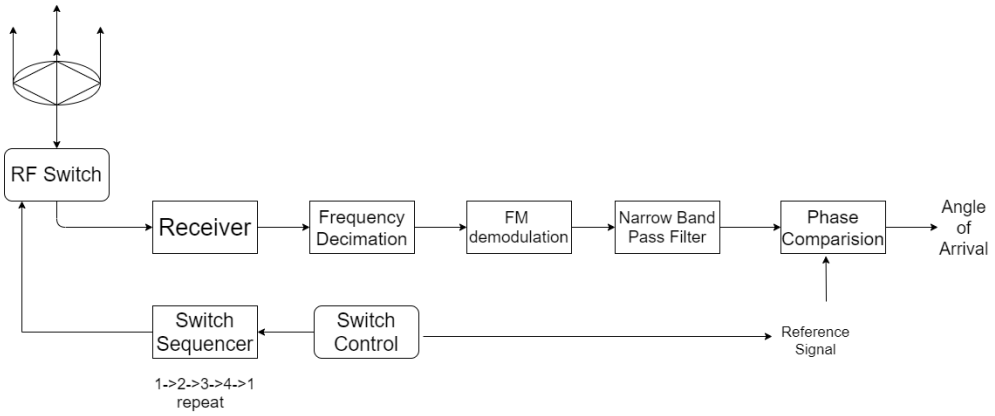


Figure 6: Pseudo Doppler Radio Direction Finding System

4.1 Signal Acquisition

The antennas are placed regularly on a circle of radius 'r'. Thus, on a complex plane the normalized positions of the antenna can be represented as [1, 0+1j, -1, 0-1j] as shown in the Figure 7. The antennas are sampled at a regular interval of T_s seconds. Thus, signal would emulate the Doppler signal of rotation time period of $4T_s$. The frequency spectrum of the simulated pseudo Doppler signal is shown in Figure 8. Note that due to complex representation only one "copy" of the spectrum is generated as the signal is no longer conjugate symmetric. Further, the FM spectrum is shifted 9kHz off DC to avoid the SDR DC-bias issue.

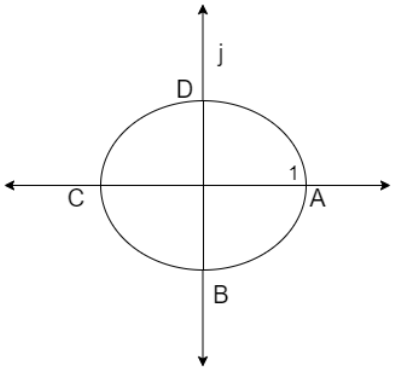


Figure 7: Antenna positions separated at 90°in complex plane

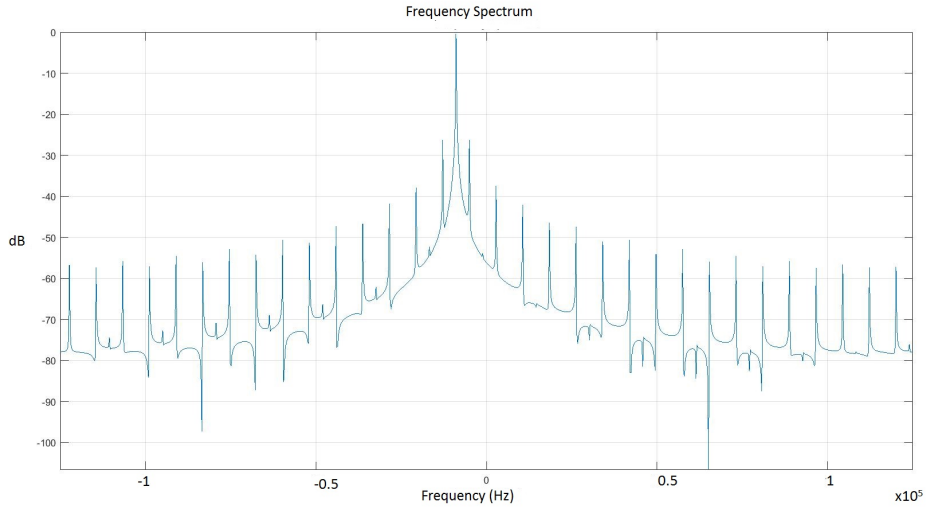


Figure 8: Frequency Spectrum of the pseudo Doppler signal

4.2 Decimation

Since the sampling rate is very high compared to the rotational rate, it is desirable to perform frequency decimation prior to demodulation. The pseudo Doppler signal is then decimated by a factor 'D' and then demodulated. The factor 'D' should be such that the signal would not suffer aliasing. The decimation is performed with Finite Impulse Response (FIR) filters. The filter employs a narrow low pass which allows only the modulating signal i.e. the rotational signal. The FIR filters have a cutoff frequency 10% greater than that of the rotation frequency. The choice for the order and transition width of the filter is based on observation and is dependent on

the quality of signal received. The FIR filter used in this application has order of 60 and has a transition width 1/10th of the rotational frequency. The decimation factor should be such that the resultant sampling frequency is greater than the width of the low pass filter.

The low pass filter was designed as follow:

- Type of filter: Type II FIR
- Order: 60
- Cutoff-frequency: 1.1*rotational frequency
- transition width: (rotational frequency) /10
- window type: Hamming

For example, a cutoff frequency of the FIR low pass filter is 4.8812 kHz*1.1 = 5369.2 kHz. Thus, the decimation factor 'D' should not be greater than $\frac{f_s}{cutoff\_frequency*2} = \frac{250000}{5369.2*2} \approx 23$. Here, a value of $D = 10$ is chosen, making the new sampling frequency 25kHz.

4.3 Demodulation

FM demodulation is performed via Quadrature demodulation, chosen for its simplicity. A quadrature demodulation was implemented by multiplying the signal with its one sampled delayed conjugate version of itself. Mathematically, it can be expressed as

$$y[n] = angle(x[n]\bar{x}[n-1]). \tag{14}$$

The signal received by an antenna is given by [6]:

$$x[n] = Ae^{j\frac{2\pi r}{\lambda}cos(\frac{2\pi n}{N_a}-AOA)}. \tag{15}$$

The quadrature demodulation for the pseudo Doppler equation can be expressed as

$$\begin{aligned} y[n] &= angle(Ae^{j\frac{2\pi r}{\lambda}cos(\frac{2\pi n}{N_a}-AOA)}\overline{Ae^{j\frac{2\pi r}{\lambda}cos(\frac{2\pi (n-1)}{N_a}-AOA)}}) \\ &= angle(Ae^{j\frac{2\pi r}{\lambda}cos(\frac{2\pi n}{N_a}-AOA)}Ae^{-j\frac{2\pi r}{\lambda}cos(\frac{2\pi (n-1)}{N_a}-AOA)}) \end{aligned} \tag{16}$$

The expression in (16) yields,

$$\hat{y}[n] = \frac{4\pi r}{\lambda}sin(\frac{\pi}{N_a})cos(\frac{2\pi n}{N_a} - AOA) \tag{17}$$

The frequency spectrum of the demodulated signal is shown in Figure 9. From the spectrum, it is observed that a strong signal at given rotational rate is observed.

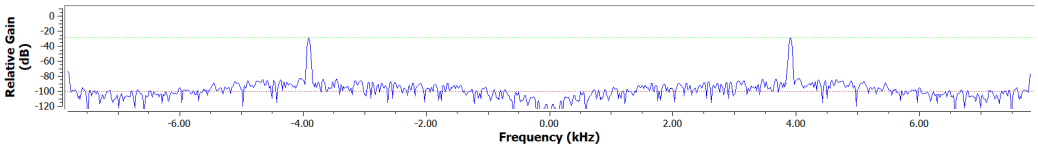


Figure 9: Frequency Display of the demodulated Signal (rotational rate = 3.9 kHz)

4.4 Band Pass filter

The demodulated signal is then passed through a narrow bandpass filter to smoothen the signal as well as to filter out any interference. The band width of the signal is 1% of the rotational rate. Thus the upper cutoff frequency is 0.95* rotation rate and lower cutoff frequency is 1.05* rotation rate. An FIR bandpass filter of order 20 is used in this application. The choice of the order of the filter is based on the observation and quality of received signal. The magnitude response of the band pass

filter is shown in Figure 10. The sinusoid received from the bandpass filter is then compared with the reference signal for phase measurement. The design specifications of the bandpass filter are as follows:

- Type of filter: Type II FIR filter
- Order: 20
- Low Cut off frequency: 0.95* rotational frequency
- High Cut off frequency: 1.05 *rotational frequency
- Transition width: rotation frequency /10
- Window: Hamming

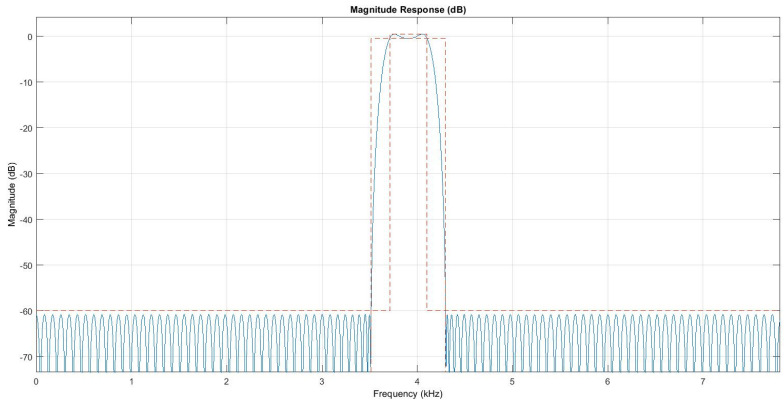


Figure 10: Magnitude response of the Narrow Band pass filter

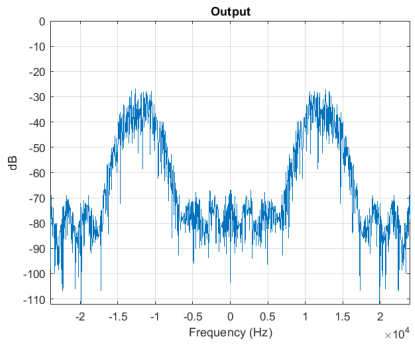
4.5 Phase Measurement

The phase measurement of the filtered signal with respect to the reference signal gives the AOA. There are several methods to give the phase difference between two sinusoid signals. Since the incoming signal has low duty cycle, the method must be able to differentiate between presence of signal and absence of signal.

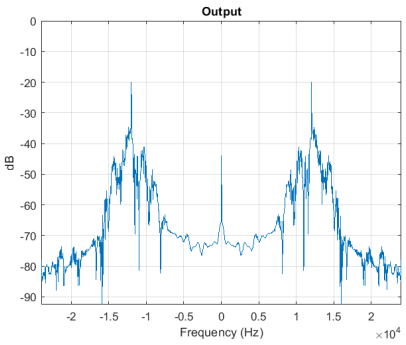
Initially, Discrete Fourier Transform (DFT) was implemented for phase measurement but later was abandoned for "statistical" method as Fast Fourier transform (FFT) was limited when implemented for duty cycle less than 5%.

4.5.1 FFT based phase measurement

Initially, FFT was implemented because the same frame of data could be used for both phase measurement and signal validation. The presence of the signal is identified by strong tone at rotational frequency as shown in Figure 11(b), while absence of strong tone suggests signal is not present (Figure 11(a)).



(a) Frequency spectrum when the signal is not present



(b) Frequency spectrum when the signal is present

Figure 11: Received signals when the signal is present and not present (SNR of 5dB)

This method was able to provide good estimates when the duty cycle of the incoming signal was greater than 3%. As the duty cycle decreased, the output became unstable and angle estimate could not determined. Figure 12 shows the result of AOA estimates against True AOA for different duty cycle signals. The observation for AOA of 45 degree with deviation at different duty cycle is shown in Table 1. Figure 13 shows the AOA estimates for duty cycle of 40% and 5%. From the observed that, it can be inferred that as the duty cycle decreased the performance of this method also decreased.

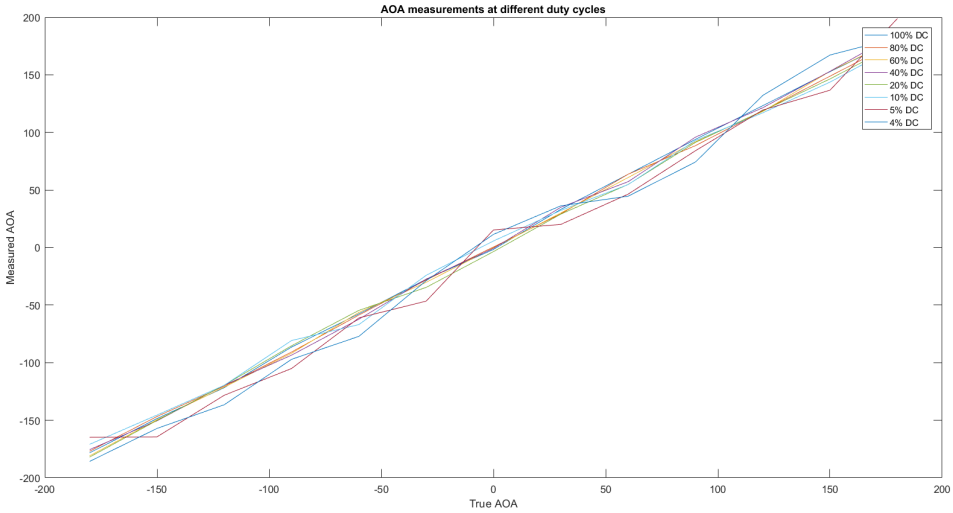


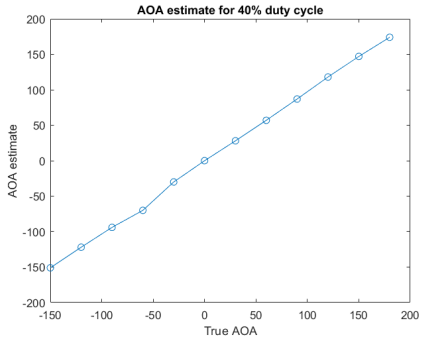
Figure 12: True AOA vs Measured AOA at different duty cycles

Table 1: AOA measurement at pulse width signals.

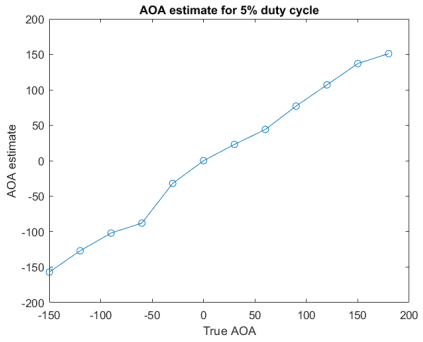
Angle of Arrival	**Duty Cycle %**	**Deviation in measured Angle**
45	100	±1
	80	±3
	60	±3
	40	±5
	20	±7
	10	±10
	5	±18
	4	±20
	3	Unstable
	2	Unstable

The instability of measurements on low duty cycle is due to the fact that the length of the FFT is longer than that of signal present. After decimating the incoming signal by 10, the resulting sampling frequency is 25 kHz. Thus, a frame of 1024 is of length $= \frac{1}{25000} * 1024 = 41ms$ which is much longer than the signal width at low cycle. Hence, it is not possible to get a stable result with an FFT of size 1024.

The RF switching is not instantaneous as in simulation.It takes some time (here $1\mu s$) to switch from one antenna to another. Thus, the frequency of the filtered signal varies slightly with the rotation rate. Hence, it is not possible to provide exact



(a) True AOA vs Measured AOA at 40% duty cycles.



(b) True AOA vs Measured AOA at 5% duty cycles.

Figure 13: AOA estimates at 40 % and 5% duty cycle

index for the rotation rate leading to spectral leakage. Spectral leakage is more pronounced as the resolution of the FFT is reduced. The implementation of FFT of shorter length was thus abandoned..

4.5.2 Single Pole IIR filter

Another method to filter out the valid measurements is using Low-pass Single IIR filter. A single pole Infinite Impulse Response (IIR) filter is defined as

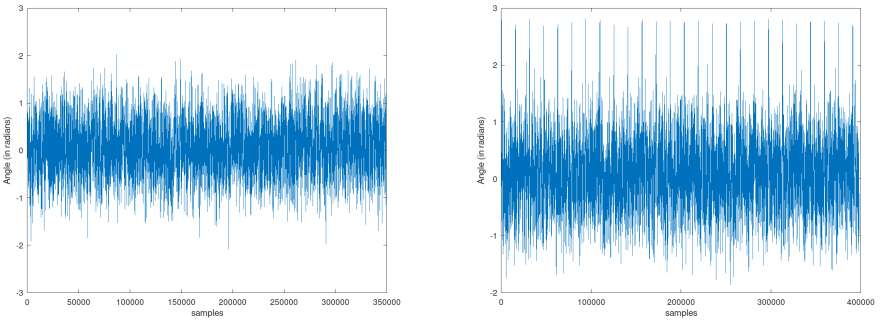
$$y[n] = bx[n] + ay[n-1]. \tag{18}$$

The parameters can be defined in terms of decay 'G'. The decay parameter 'G' describes how fast or slow the exponent response is approaching to zero. This parameter can be tuned as required in the operation of the filter. Consider, $a = G$ and $b = 1 - G$. Thus, (18) can be re-written as

$$y[n] = y[n-1] + G(x[n] - y[n-1]). \tag{19}$$

The single pole IIR filter is a moving average filter that applies weighing factors exponentially. Thus, the filter would suppress the values which are not random in na-

ture, i.e. different than that of its neighbors. Since, noise are random in nature, this filter can be used to find out the valid angle measurements from the invalid ones. However, for small angles (< 1 radian), it is unable to find the valid angles because the angles were too small. Figure 14 shows the result of the filter on the angle estimates. As shown in the Figure14, the filter is good at suppressing the noise when incoming angles are more than 1.5 radians while for angles less than 1.5 radians it cannot separate the valid angles from invalid ones (note flat peaks in 14(b)).



(a) Single Pole IIR filter on low angle measurements

(b) Single Pole IIR filter on high angle measurements

Figure 14: Single Pole IIR filter on angle measured

In order to utilize the filter for the full 2π radians, attempts such as signal shifting and unwrapping were used but were not successful. Hence, this method was abandoned in favor of the statistical method.

4.5.3 Phase Measurement based on Statistical method

The phase between two signals of the same frequency can be calculated via direct conjugate multiplication. The filtered signal is multiplied with the reference signal to give the phase (where the reference signal is conjugate of sinusoid of frequency equal to that of rotational frequency). The filtered signal is a complex signal that can be represented as $x(t) = A_1 e^{j(\omega t+\phi)}$ where ω is the rotational frequency and ϕ is the AOA. Hence, the reference signal would be $ref(t) = A_2 e^{-j(\omega t+\phi_2)}$ where ϕ_2

is some phase. This phase offset (ϕ_2 can be corrected with calibration and will be discussed in 4.6.1) The two signals are multiplied to give the AOA as,

$$\begin{aligned} AOA = angle(x(t) * ref(t)) &= angle(A_1 e^{j(\omega t+\phi)} * A_2 e^{-j(\omega t-\phi_2)}) \\ &= angle(A_1 * A_2 e^{j(\omega t+\phi-\omega-\phi_2)}) \\ &= angle(A_1 * A_2 e^{j(\phi-\phi_2)}) \\ &= \phi - \phi_2. \end{aligned} \tag{20}$$

Due to the extremely low duty cycles, an ad hoc method must be established to determine if the signal is present. To that end the presence of signal can be identified by observing the angle estimates given by (20). If the signal is present, then the angle measured will be consistent over its portion of the frame. The absence of signal can be observed through the random non-consistent phase estimates. Figure 15 shows the angle measurements over a period of time. It can be observed that when signal is present, the measurements are consistent while estimates are random when signal is not present.

The mean and standard deviation of the AOA estimate is calculated for each frame of data. Under the normal distribution, 68% of the data falls within one standard deviation. A low standard deviation suggests that the signal was present in the frame and the mean value of the estimation is then taken as valid angle.

4.6 Calibration

The AOA estimate is the phase difference measurement between the Doppler signal and reference signal. As the incoming signal gets processed via different processes to generate the Doppler signal, additional phase is introduced. Since, the AOA is embedded as phase in the Doppler signal, the AOA estimate is offsetted.

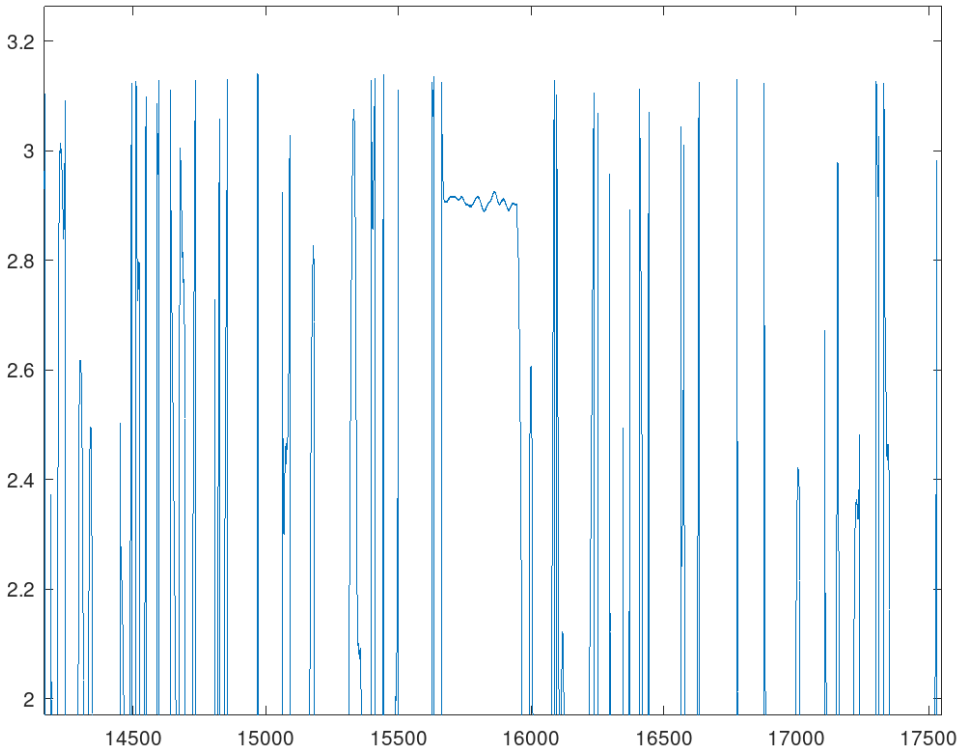


Figure 15: Zoomed region to show the consistent measurements

The other source of phase introduction comes from the lack of synchronization between the switch and program run. The switch starts operating as soon as the power is plugged into the SDR, while the program is run only after a while. This introduces additional phase to the Doppler signal. In order to get the accurate angle estimate, it is necessary to account for these additional phase shifts.

4.6.1 AOA Measurement

In order to get the accurate estimate, it is necessary to calibrate the pseudo Doppler RDF. As the signal passes though various components, it gains additional phase and thus some offset is present in the angle estimate. This offset is expressed in (20) as ϕ_2. In order to get the AOA estimate, the offset at 0 degree is measured first. This offset is then subtracted from measured phase to give the AOA estimate.

4.6.2 Threshold estimation

Although under normal distribution, 68% of the data fall within 1 standard deviation, this value has to be adjusted based on the surrounding environment. Thus, during calibration the threshold for standard deviation is adjusted. The threshold value is set when the observed angle estimate is stable i.e. threshold value is where consistent observations could be made as shown in Figure 15. The threshold estimation is based on observation and may need to be calibrated at every use.

5 System Implementation

The pseudo-Doppler RDF is implemented using LimeSDR-mini by Lime microsystems [29]. The PE423641 evaluation board is used as an antenna switch and standard VHF monopole antennas are used for the antenna array. This chapter describes the implementation of the system, hardware and software architecture.

5.1 Hardware architecture

The schematic of the RDF is shown in figure 16.

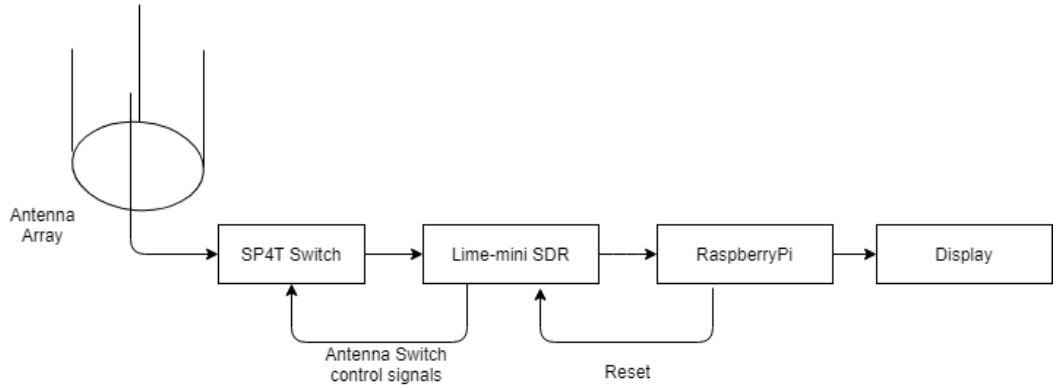


Figure 16: Schematic of the RDF system

The antennas are connected to the RF switch through the SP4T switch (PE423641). The antennas are connected to the ports RF1, RF2,RF3 RF4 respectively. The RF switch is controlled through the control signals V1 and V2. The output signal from RF switch is fetched to the LimeSDR mini though the RX1 port. The connection from RF switch to the SDR is done though the SMA cables. The schematic for the hardware components connection is shown in Appendix A.

A reset switch is interface with the GPIO 1 of the SDR. It is resets control signals to the RF switch and helps in angle calibration. The SDR is interfaced with the Raspberry Pi (any other similar platforms) through the USB cable.

5.2 Hardware Components

5.2.1 LimeSDR-Mini

The Lime-SDR mini is a low-cost SDR platform. It utilizes Intel's MAX 10 FPGA and Lime microsystems transceiver (LMS7002M) for prototyping and developing digital and RF designs. LimeSDR-Mini board covers RF frequency range from 10 MHz up to 3.5 GHz making it suitable to work on different wireless communication protocols. The mini performs 12-bit ADC, has 1x1 TX/RX channels and the maximum bandwidth of 30.72 MHz. The board is run by 40 MHz onboard clock. It utilities USB 3.0 for data connections. LimeSDR-mini is shown in Figure 17 (note the size of the USB connection).

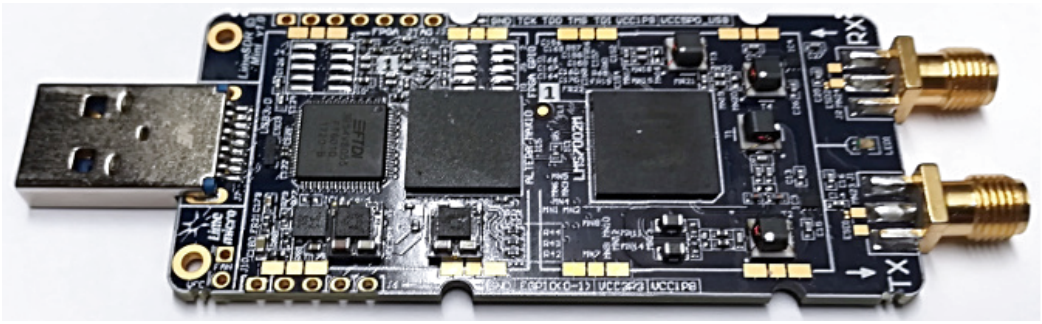


Figure 17: LimeSDR-mini

The LimeSDR-Mini has an on-board programmable FPGA (Intel Max 10) programmed through FPGA JTAG or Limesuite software interface. The advantage of using the LimeSDR-Mini over other available SDRs is that the FPGA GPIO are programmable as well as open sourced. Thus, it is possible to modify the FPGA code according to the need of the application. In this case, the control signals for the antenna switch and SDR need to be synchronized. The FPGA's program is modified to generate the control signals to control the antenna switch through the programmable GPIO pins. In so doing, this solves the synchronization problem discussed in section 4.

5.2.2 Monopole Antenna Array

Four monopole antennas are placed around a circle of radius of 10cm and 90 degrees apart. The antenna array is mounted on a copper plated board for signal reflection as shown in Figure 18. Each vertical whip of the antenna is of 50cm long. Additionally grounding is attained through the two horizontal whips of 25 cm added to each antenna.

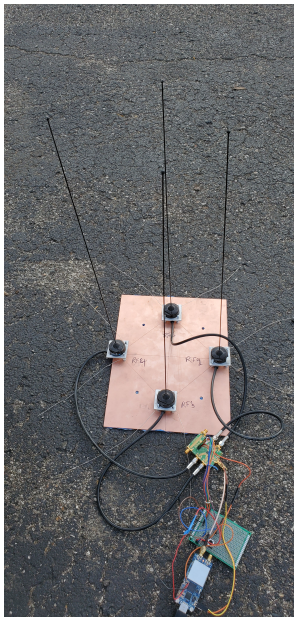
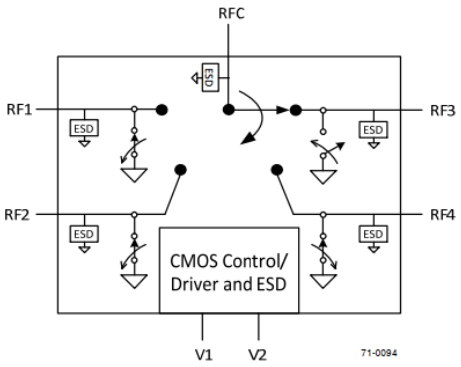


Figure 18: Monopole antenna mounted on a copper plate

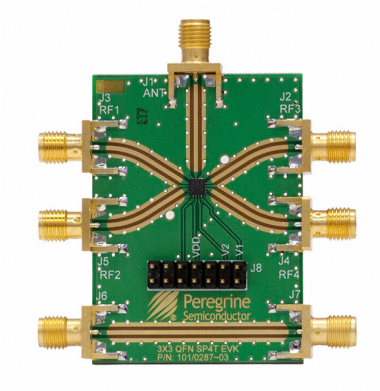
5.2.3 RF Switch PE423641

The RF Switch electrically selects the active antenna implemented via the PE423641 evaluation board. PE423641 is a SP4T RF switch. It is designed to cover a wide range of wireless applications from 50 MHz through 3 GHz. The functional diagram of PE423641 is shown in figure 19(a). The PE423641 evaluation board is shown in Figure 19(b). The RF common port is connected through a 50Ω transmission line via the J1 SMA connector. RF1, RF2, RF3, and RF4 are connected to through 50Ω transmission lines via SMA connectors to each antenna. The antenna switching is

provided though the control signals V1 and V2.



(a) Functional Diagram of PE423641



(b) Evaluation board for PE423641

Figure 19: RF Switch PE423641

The PE423641 has a maximum switching frequency of 25 kHz where switching frequency relates to the time duration between switching events. The truth table shown in the Table 2 defines the signal routing.

Table 2: Truth Table for the RF switch

Path	**V2**	**V1**
RFC-RF1	0	0
RFC-RF2	1	0
RFC-RF3	0	1
RFC-RF4	1	1

The control signals for antenna switching were generated by modifying the FPGA code of the LimeSDR-mini. A 13 bit counter was implemented on the FPGA to provide control signals V1 and V2 of PE423641 through 12th and 13th bit. The signal to the V1 and V2 is provided by the EGPIO pins EGPIO 1 and EGPIO 2 from the Lime SDR-mini. The Lime-SDR runs at 40 MHz, thus the frequency of V1 and V2 are:

$$V1 = \frac{40 \times 10^6}{2^{13}} = 4882.8125kHz \tag{21}$$

$$V2 = \frac{40 \times 10^6}{2^{12}} = 9765.625kHz \tag{22}$$

Hence, the rotation rate of the antenna is 4882.8125 kHz and switching frequency

between the antennas is 19,531.25 kHz. Binary counter is implemented in order to reduce any other extra circuity that would add processing overhead to the SDR. Figure 20 shows the control signals generated (Different amplitudes are due to different gain of the scope used).

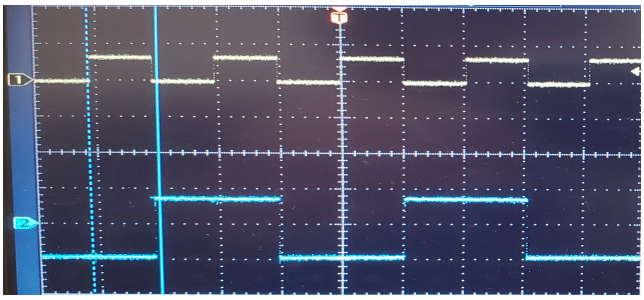


Figure 20: Control Signals for RF switch

5.2.4 HiLetgo 0.1-2000MHz RF WideBand Amplifier (Optional)

The HiLetgo RF WideBand Amplifier is a wide band high gain LNA. The amplifier provides a gain of 30dB. It was implemented to boost the signal level and is powered with 9V battery. The HiLetgo amplifier is shown in figure 21.

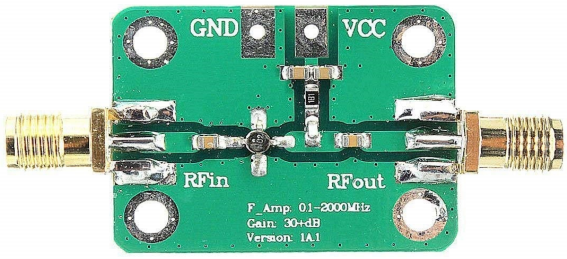


Figure 21: HiLetgo RF WideBand Amplifier

The measurement were performed at a distance of 20m. At such distance, the signal was strong enough. Hence, LNA was not used.

5.2.5 Raspberry Pi 4

As with most SDR systems, the majority of the signal processing is performed by an external processing engine. The Raspberry Pi 4 was chosen as the external processor to interface to the SDR and execute the direction finding process. The Raspberry Pi 4 is capable of running the GNU radio companion. It supports various communication protocols such as UART, SSH, USB and UDP allowing for the ease of interfacing. The Raspberry Pi's specifications are as follows:

- 1.5 GHz clock speed
- 2GB RAM
- Linux based Operating System
- Capable of handling Linux based Applications
- 40 GPIO pins
- UART, Ethernet, Bluetooth, USB and Wi-Fi support
- Light Weight
- Optional: supports battery powered application
- Low-cost($45 at this time)

5.3 System Specifications

The major components that draw power are the Raspberry Pi and limeSDR-mini. The raspberry pi 4 draws 3.4 watts at idle. Under normal working conditions, LimeSDR-mini draws 2 watts. Thus, at working condition the system draws 5.4 watts. The raspberry Pi requires an input voltage of 5V and 3A current for operation. The pe-

ripheral devices such as SDR, RF switch can be powered through Raspberry Pi.

Total Weight of the system (without battery) was measured as 1.89 lbs.

5.4 Software implementation

The majority of signal processing was performed on GNU Radio Companion (GRC) software. GRC provides signal processing blocks to implement software radios. The LimeSDR-Mini is interfaced into the GNU radio through gr-limesdr plugin and LimeSuite program. Additional programs were written in Python such as signal validation and Graphic User Interface(GUI). Python and GNU Radio communicated with each other through Uninterrupted Data Protocol (UDP). Additional programs such as angle evaluation and user friendly GUI are implemented on Python. PyQt is used to develop the GUI. Broadly the software architecture can be divided into two parts: GRC flow-graph and signal validation.

Figure 22 shows the signal flow through the different signal processing block. The **LMSuite Rx** block is the source block which interfaces with the Lime SDR-mini to receive the RF signal. The received signal is then down-sampled by the **Frequency Xlating FIR Filter** block which performs decimation by running a decimating FIR filter. The **FIR filter taps** are defined under Low Pass Filter Taps Block. After frequency decimation, **Quadrature Demod** block performs the demodulation. The demodulated signal is then passed though a narrow band pass filter which produces smooth filtered signal.

Figure 23 shows the method to calculate the phase difference using the statistical approach. The flow graph implements the method described in the 4.5.2. In the flow graph, the **Multiply** blocks takes performs one-to-one multiplication of filtered signal and reference signal. The resulting value is normalized to via **Complex To**

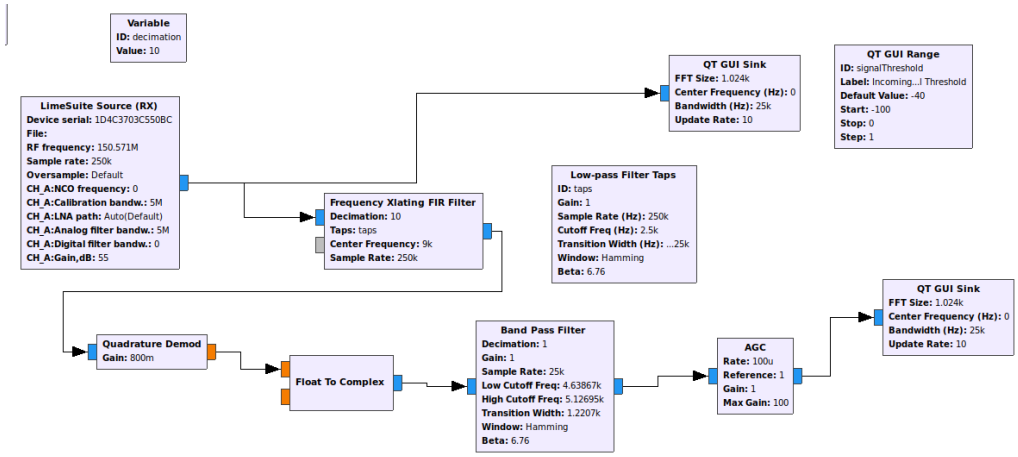


Figure 22: GRC flow graph

Mag and **Divide** blocks. The phase difference is then obtained with the **Complex to Arg** block. These angle measurements are then transmitted to the Python program for signal identification and display via UDP block.

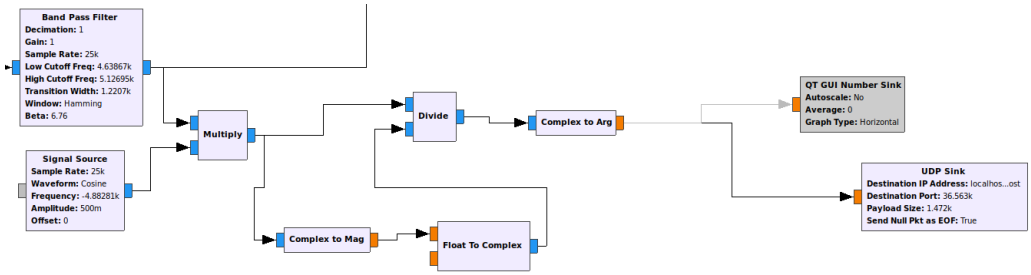


Figure 23: Flow graph to calculate phase by signal multiplication

The Python program 'pulsedSignal.py' implements the signal identification process as discussed in 4.5.2. The flowchart for this process is shown in figure 24.

The program is GUI interfaced with compass. The Figure 25 shows the GUI implemented using python and PyQt. The compass program is based on the open source program [30]. It is interfaced the GNU Radio Companion through UDP. The GUI performs calibration (as discussed in 4.6) and displays the valid angles estimates.

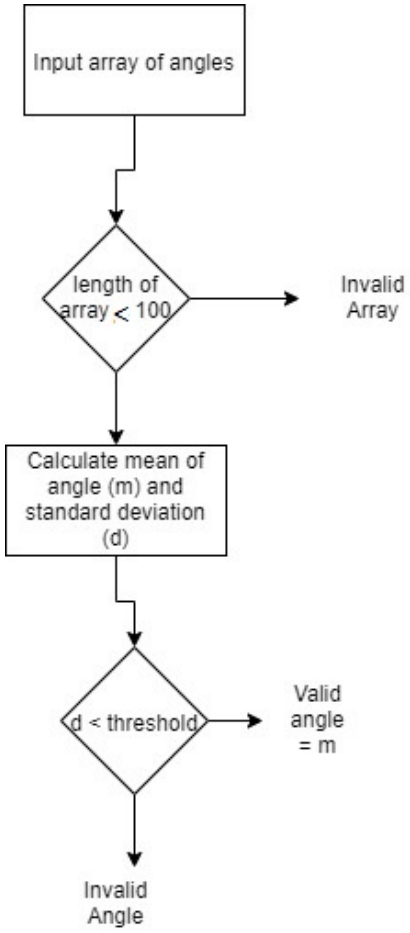


Figure 24: Flowchart to validate the signal from input block of angles

Figure 25: Output GUI to display the AOA

6 Validation

Before testing the system on low duty cycle signals, it is necessary to check if the algorithm is working properly. It is difficult to debug and check the working condition of the system for low duty cycle signal. The initial tests were performed on continuous signal. This would validate the if the algorithm is working properly or not.

6.1 Test Setup

A test setup was made in order to validate and analyze the performance of the pseudo Doppler RDF. The initial tests were performed inside the building of Innovation and Design Center, Grand Valley University. While the tests inside the building could provide verification for the system, the measurements were fluctuating. Hence, later tests were performed on the parking lot of the same building for improved results. The transmitter was provided with sufficient power such that the receiver receives the signal properly. The receiver was placed at a distance of 20 m away from the transmitter for bearing angle measurements. The measurements were taken when the effect due to the weather is minimal. Figure 26 shows the receiver setup with a PC. The PC was interfaced with the SDR to analyze received data in real-time.

6.1.1 Signal Generation

A meter antenna connected to the function generator was the source of radio signal for the experimentation. An operating frequency of 150.580 MHz was used to match the actual working frequency of a wildlife tracking transmitter. Tests were carried out for both continuous signal and low duty cycle signals. It was not possible to generate low duty cycle with a single function generator. Hence, in order to

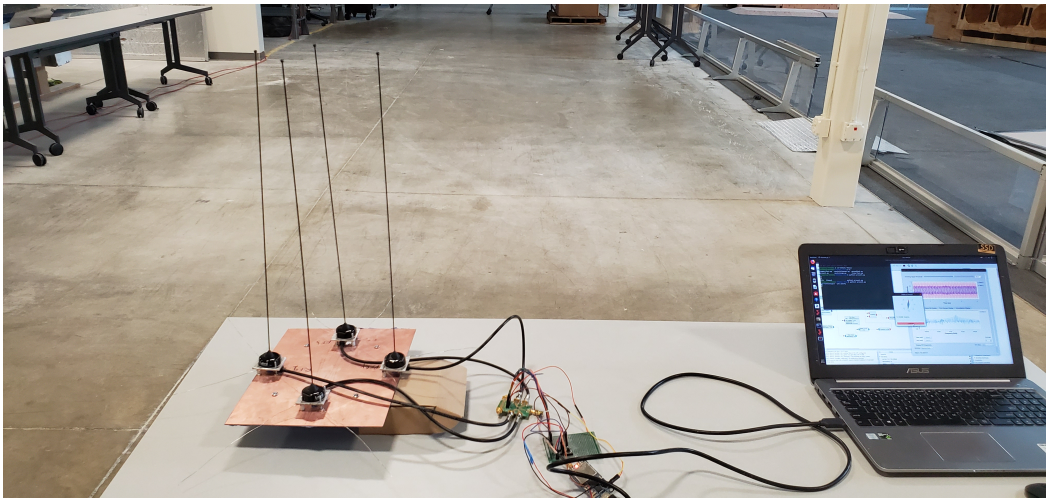


Figure 26: Setup for the observation of AOA with PC

generated low duty cycle signal, two function generators were cascaded. The source generator was operated under amplitude modulation mode with 100% depth. The second function generator then provided the external signal to produce output signal of low duty cycles. The second generator was run on pulse mode for this purpose. The pulse width of this generator was varied to generate output signal of different duty cycle. The generated pulse signal is shown in Figure 27 (Figure is not the actual signal for test purpose and is to show that the signal generated has duty cycle).

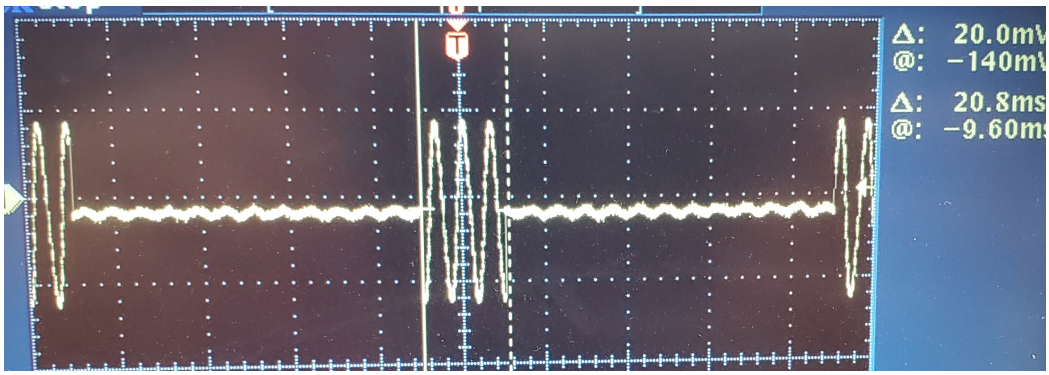


Figure 27: Pulse Generated

6.1.2 Test on Continuous Signal

In order to validate the pseudo Doppler Radio Direction Finding (RDF), it was tested on continuous signal. Figure 28 shows the real-time incoming signal received on the GNU radio software. The figure shows only the real part of the I/Q data received through the SDR (Both I/Q are present, the Q part is hidden only for display). The sharp changes on the signal received are due to the RF switching. Since all the antennas did not have same gain, received signal is observed to be asymmetrical. The difference in gain of the antenna will not effect the performance of the RDF. This was verified via simulation. It was observed that unless the gain of the antenna is zero or very small, the performance of the RDF will not degrade.

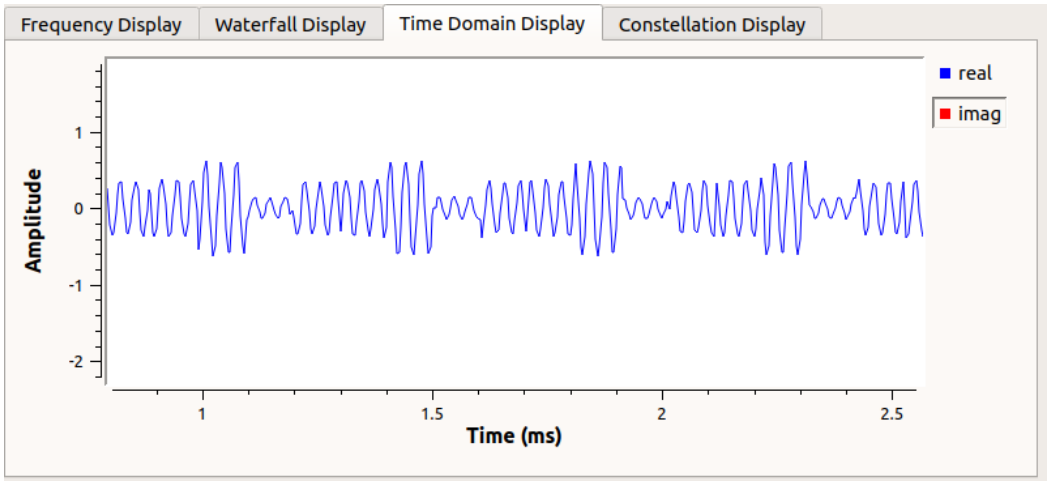


Figure 28: Signal Received on GUI of GNU Radio

Figure 29 shows the simulation for signals received at varying antenna gain for four different antenna. Figure 30 is the frequency display of the signal received from the band-pass filter which is the same compared to antenna array with same antenna gain. It shows that the algorithm is independent of the antenna gain. Since pseudo Doppler is phase based method, amplitude of received signal should not be a problem unless it is very small. This provides much more flexibility while choosing which antenna to use.

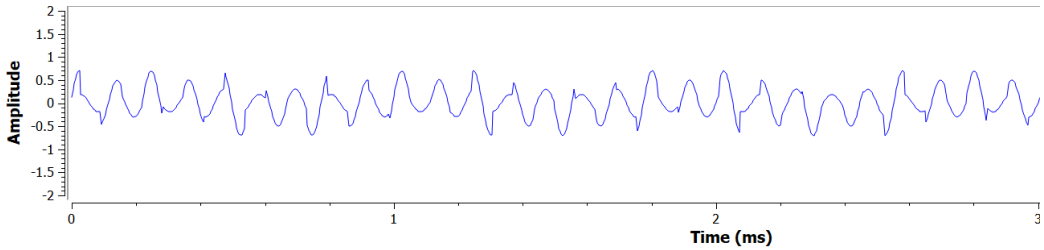


Figure 29: Signal simulated for uneven antenna gain

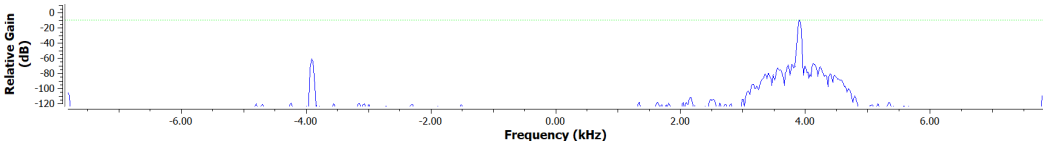


Figure 30: Uneven Signal filtered response

In order to test accuracy of the pseudo Doppler device, Angle of Arrival (AOA) calculation is done at every 15°. The measurements are made only after calibrating to the 0°. After multiple measurements, a mean absolute error of 4.0086 is observed. Figure 31 shows the corresponding graph of the measurements. Thus, it can be confirmed that the algorithm is functioning properly and can provide good AOA estimate.

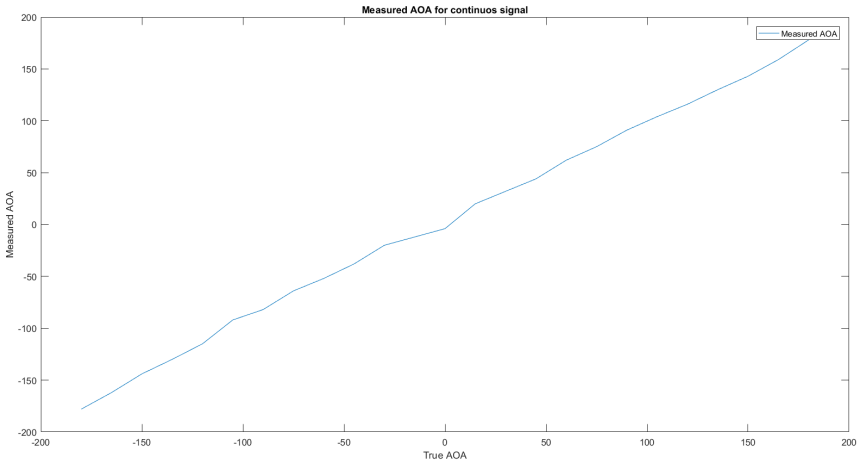


Figure 31: Measured AOA vs true AOA

7 Results and Discussion

7.1 Performance of the Pseudo Doppler Based RDF system

In order to test the performance of the system measurements were taken at 30°angle separation. Table 3 shows the AOA estimates observed for duty cycle of 2% to 1.2%. Figure 32 shows the corresponding data. It can be observed that the system provides acceptable results for the duty cycle approaching 1%. For duty cycle of 1.2% the system is capable of providing the estimate within 30-20 degrees while with 2% the accuracy is within 10 degrees.

Table 3: AOA estimates at different low duty cycle

True AOA	**2%**	**1.8%**	**1.6%**	**1.4%**	**1.2%**
0	0	0	0	0	0
30	31	33	26	29	19
60	62	58	57	63	71
90	93	93	94	95	81
120	119	123	122	126	121
150	153	152	149	147	158
180	178	-173	-176	-174	-171
-150	-147	-147	-150	-150	-162
-120	-120	-121	-117	-117	-134
-90	-87	-89	-92	-92	-89
-60	-58	-56	-57	-57	-40
-30	-30	-27	-33	-33	-35

Here, frame size of 256 is used to validate angle measured. The frame length is based on observation of the angles received at different directions. It was observed that at frame size of 256 most consistent result was obtained. Thus, a frame of 256 would be length $= \frac{1}{25000} * 256 = 10.2ms$. It is observed that for signals of 1.2% or lower a frame size of 256 was not sufficient enough to find the consistent set low deviation angles. An attempt was made with smaller frame sizes. Under small frame sizes, the standard deviation of the valid angles were not significant in compared to that of the non-valid signals. Hence, under smaller frame size it is not possible to

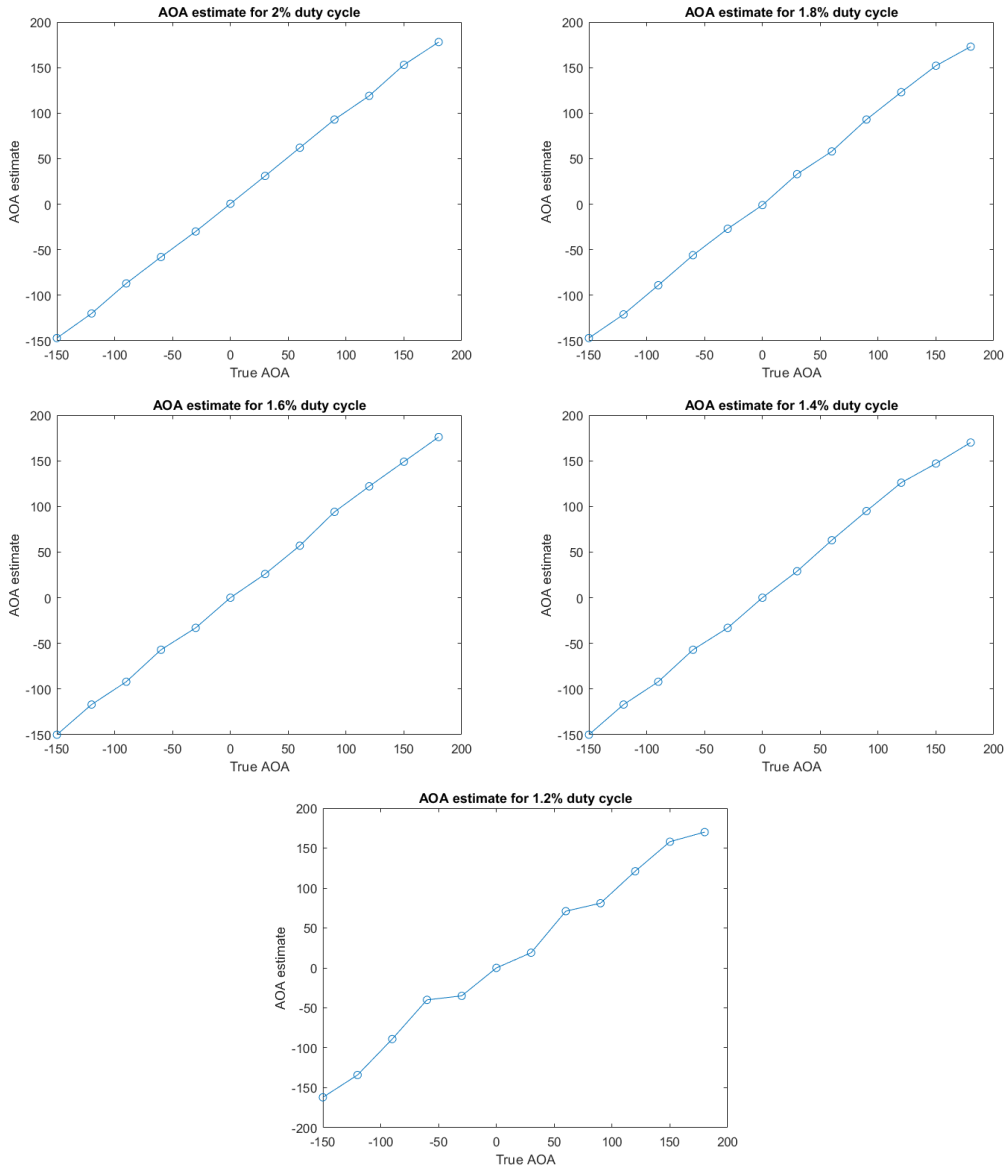
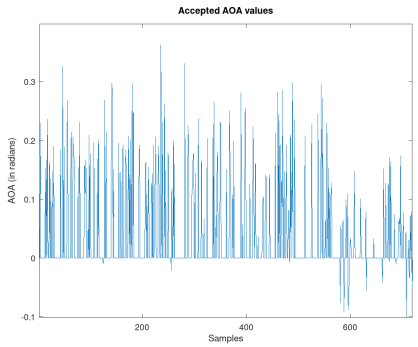
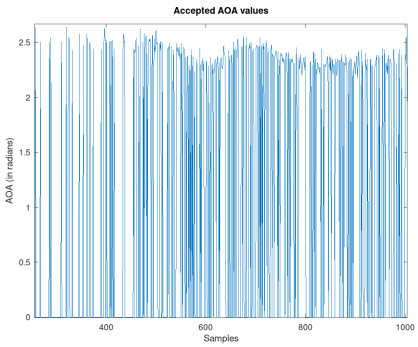


Figure 32: AOA estimates for low duty cycle signals

distinguish between valid and in-valid AOA measurements. The increase in frame size would make the frame width longer that the signal width, and thus increase in error. Figure 33 shows the valid angles measurements for duty cycle of 1.2% and 1.6%. Thus, for duty cycle of 1.2% the angle estimates had to averaged over some valid frames. Figure 33 are un-calibrated raw values, hence does not reflect the actual angle measurement.



(a) Measured AOA at duty cycle 1.2%



(b) Measured AOA at duty cycle 1.6%

Figure 33: AOA measurement on duty cycle 1.2% and 1.6%

7.2 Performance Summary

It was possible to measure the AOA for low duty cycles with statistical method. This method was found be more accurate and functioning at low duty cycle compared to other methods. Since, the frame size can be smaller it is possible to utilize the method for lower duty cycle signals. This method also suffers from the limitation of frame width, but not severe as other methods.

7.3 Performance comparison with MUSIC

The performance of the pseudo Doppler RDF was tested against the MUSIC algorithm. The MUSIC algorithm employs multiple antenna array system, that uses phase information from each antenna to estimate the AOA [31][2]. The MUSIC performs the eigen decomposition on the covariance matrix of signal data. The decomposition results into signal subspace orthogonal with noise subspace. These two orthogonal subspace are used to compute a spectrum function and through spectral peak search, AOA is calculated.

The tests were performed on KerberosSDR [32] which had MUSIC as one of the built in direction finding algorithm. KerberosSDR is 4 phase coherent RTL-SDRs.

The calibration and measurements instruction are based on KerberosSDR developers [32]. The measurements were taken from fixed transmitter at 20m. AOA measurements were taken taken at every 30°.

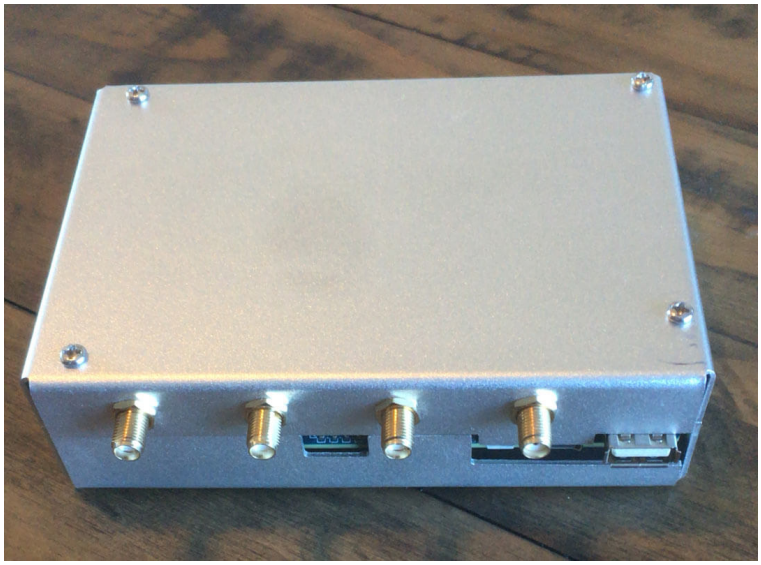


Figure 34: Kerberos SDR [33]

Initial tests were performed on the same antenna array where the pseudo Doppler was implemented. Thus, the test conditions were:

- Signal Frequency 150.580 MHz;
- Tuning Frequency 150.571 MHz;
- Array radius 'r' 0.05λ;
- Sampling frequency 1 MHz.

When the array size was less 10cm, angle estimates were highly varying and not consistent. Hence, the tests were carried out for larger antenna array (35cm). The performance of the system was better than of 10 cm. The AOA estimates observed on the radius of 30 cm is shown in Figure 35. It was observed that the AOA estimates were within 30 degrees. Figure 36 shows the antenna array and with KerberosSDR and RaspberryPi (note the size of array compared to the table top).

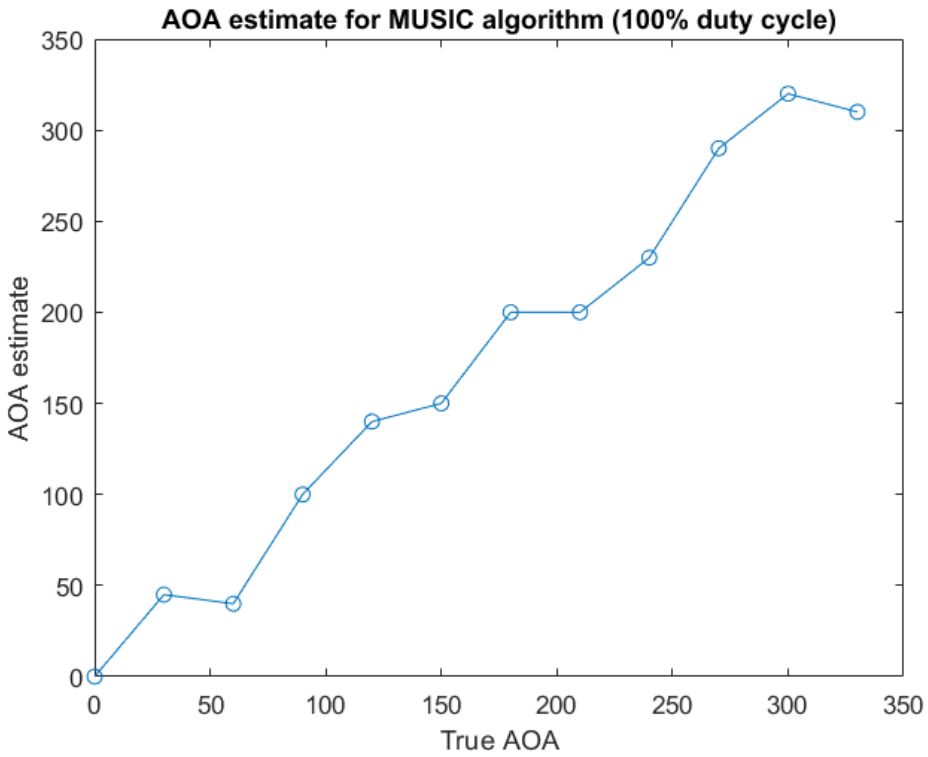


Figure 35: AOA estimates

Since, the update rate of the KerberosSDR was long (on average 5ms), compared to duty cycles, the performance of MUSIC was only tested on continuous signal.

Based on the observations following comparisons can be drawn.

- Pseudo-Doppler RDF can be implemented on smaller radius sizes compared to MUSIC RDF with more accuracy,
- MUSIC requires coherent antenna arrays which would require specific hardware whereas pseudo-Doppler antenna array need not to be coherent,
- MUSIC requires high computations and has slow update rate while pseudo-Doppler has low complexity and produce result at much faster rate,
- KerberosSDR has high power requirement (operating current 3A), whereas pseudo-Doppler can be implemented on low power consuming SDRs.

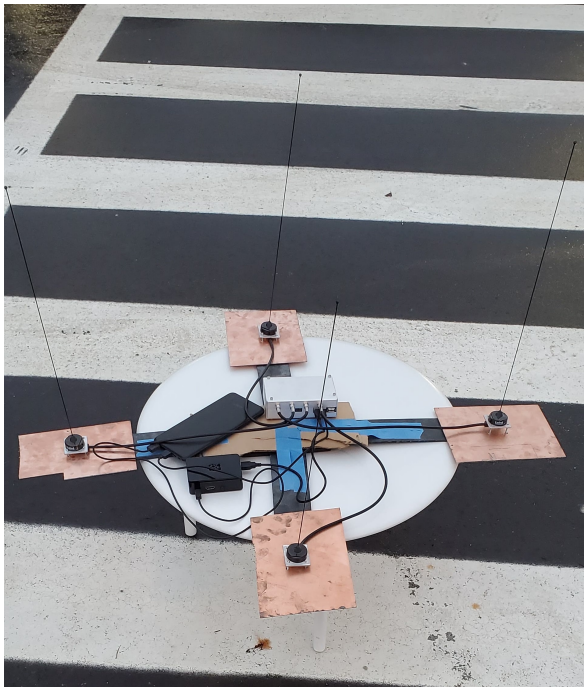


Figure 36: Antenna array of radius 35 cm for MUSIC implementation

7.4 Performance comparison with Directional antenna

Directional antenna receives greater power in specific direction. Yagi-Uda antenna system was implemented to compare with pseudo-Doppler RDF. Since, the directional antenna cannot provide angle by itself, external equipment (such as compass) is required to find the bearing of the incoming signal. Measurements were taken at every 45°. Measurements at smaller steps could not be done as the reflections due to surroundings affected the signal detection method.

Yagi-Uda antenna was employed to estimate angle of arrival for low duty cycle signals. A signal detect algorithm was implemented to check if the signal is present or not and based on the angular position from the transmitter. Angle measurements were taken by placing the transmitter at random positions and then directional antenna was used to find the angle estimate.

Directional antenna was employed for low duty cycle signal (here 1.4%). A signal detection algorithm was implemented. The algorithm calculates the Energy Spectral

Density (ESD) of given frame and check for peak for given signal of interest. The presence of sharp peak at signal of interest denoted the presence of signal. Here, the frequency of the transmitted signal is 150.580 MHz and the tuning frequency is 150.571 MHz. Thus, the signal detection algorithm calculates ESD for given frame and checks if a peak is present at 9kHz. The AOA estimates against True AOA is shown in Figure 37. It was observed that the directional antenna could provide an AOA estimate within 25-30 degree.

Test conditions

- Antenna length 1m
- Signal Frequency 150.580 MHz
- Tuning Frequency 150.571 MHz
- Sampling Frequency 250kHz
- Decimation factor 'D' 10
- FFT length 512

Thus, based on above condition the index for signal of interest is calculated as $ceil(\frac{512*9000}{250\times10^3/10}) = 185$.

Based on the observations following comparisons can be drawn.

- Directional antenna requires external equipment to estimate the AOA and is limited by the accuracy of the equipment;
- The accuracy of the system is limited by its directivity. Additional components can be added to increase the directivity of the system but would make it bulky;

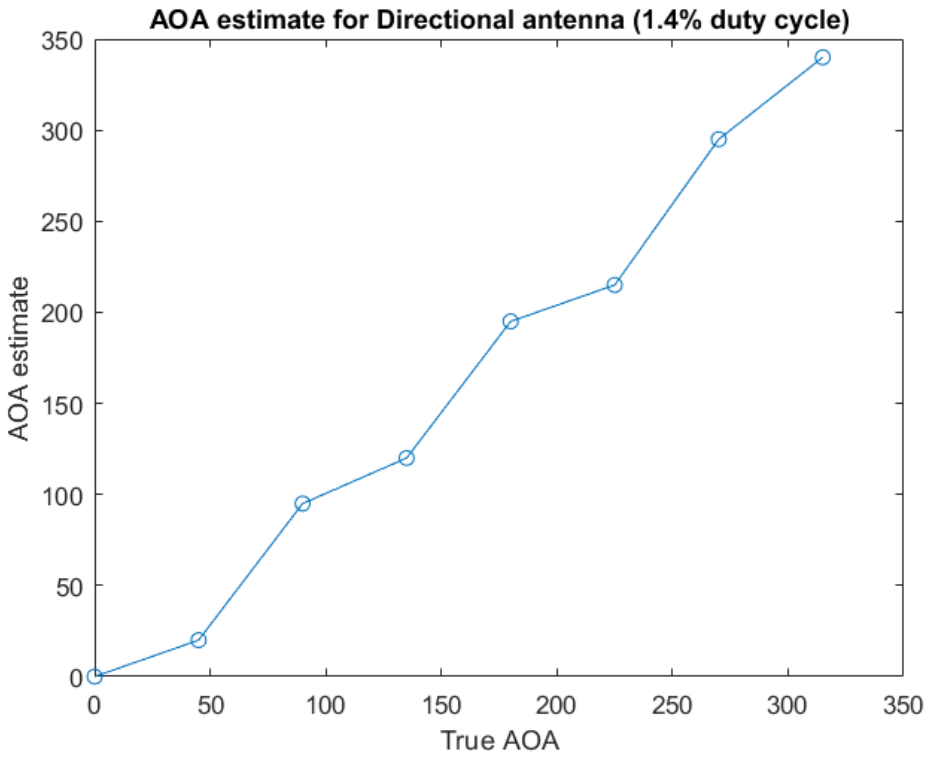


Figure 37: AOA estimates

- Directional antenna are more robust compared to pseudo-Doppler RDF, and can be used under different weather conditions.

8 Conclusions

The pseudo Doppler method was defined in Chapter 2 and designed in Chapter 4. Then, the system was implemented in Chapter 5. Two different methods were employed in an attempt to find AOA for signals with low duty cycle. The first attempt was made with FFT. FFT of length 1024 was capable of finding the AOA within 10 degrees for signal of 100 ms. As the duty cycle became shorter, the limitation of length of FFT hindered in accurate measurement.

"Statistical Method" was then implemented to identify the valid angles. Since the angle measurement is performed in time-domain, the frame size could be of much smaller than that of FFT. This method was capable of finding the AOA for duty cycle of 1.4% with 10 degrees accuracy. The accuracy could have been better if the GNU Radio and the angle validating program could interface with consistent data sizes.

The performance of the pseudo-Doppler RDF was compared with that of MUSIC for same array size and directional antenna. It was observed that pseudo-Doppler RDF provided better accuracy for small array size. When compared to directional antenna, pseudo Doppler provided more resolution in angle measurement. Thus, pseudo Doppler can be a good candidate that can be deployed on a UAV based animal tracking.

8.1 Obstacles and Lessons Learned

8.1.1 SDR selection

A great deal of time was consumed in figuring out the appropriate SDR to use. Initially, RTL-SDR as SDR as receiver and MSP-432 as control signal generator was

used. The problem with this method is that the control signals to RF switch and SDR were not synchronized. As a result, a drifting signal was received i.e. the filtered signal did not have a stationary phase difference with the reference signal. Hence, it was not possible to find a consistent AOA. Further SDRs such as Hack-RF and Kerberos SDR were tried. With these SDRs, the control signals to RF switch can be only provided through their main clock which runs at 10 MHz. Thus, to use their clock it has to be down-sampled by a huge factor. Implementing such large bit counters would be impractical.

LimeSDR-mini provided a low-cost SDR solution. The GPIO of the LimeSDR-mini can be programmed through its open source FPGA code. Hence, the FPGA code was studied to implement a bit counter that would provide control signal to the RF switch.

8.1.2 Validation and testing

In order to verify the accuracy of the algorithm, often large amount of data were dumped to a text file by GNU Radio. These files were analyzed in Octave to figure out possible errors.

Often the python program to validate the signal would crash. Thus, the calibration angle would be lost. This, problem was solved by re-calibrating the RDF for new measurements.

8.2 Future Work and Final Thoughts

The implementation of the system on Lime-SDR mini provides a lot of possibility for future projects. The LimeSDR can be programmed for signal identification at the receiver end thus removing the need for angle validation at the end. A more far-

fetched application would be implementing the whole pseudo Doppler algorithm on LimeSDR.

The duty cycle of the commonly available wildlife tracking transmitter can be 1%. The current system cannot be implemented for such low duty cycle signal. However, the current system can be improved to work on much smaller duty cycle signals. The major limitation current system faced was that of the rotation rate. The maximum switching frequency of RF switch was 25 kHz, thus the maximum rotation rate that can be achieved was 6.25 kHz. This imposed the limitation on decimation factor Thus with higher rotation rate, it might be possible to work on much smaller duty cycle signals.

Single pole low IIR filter was found to be very good at validating small duty cycle signals. However, it was limited by its working range as it could not work on small angles($<$1.5 radians). A study to overcome this limitation could help on developing methods that could be useful in very low duty cycle signals.

Another method to improve the signal validation method would be improving the Single pole IIR method. It could provide validation for much smaller frames but is limited by small angle measurement. An improvement in the method would allow the current configuration to perform better in much smaller duty cycle signals.

The thesis was mostly concentrated on Signal Processing and finding algorithm to find the angles for low duty cycle signals. Hence, much focus could not be provided to antenna array construction. Antenna array elements were not optimized for signal reception. Antennas with the same gain would provide more consistent output. An antenna array with bigger radius is less susceptible to errors due to reflection.

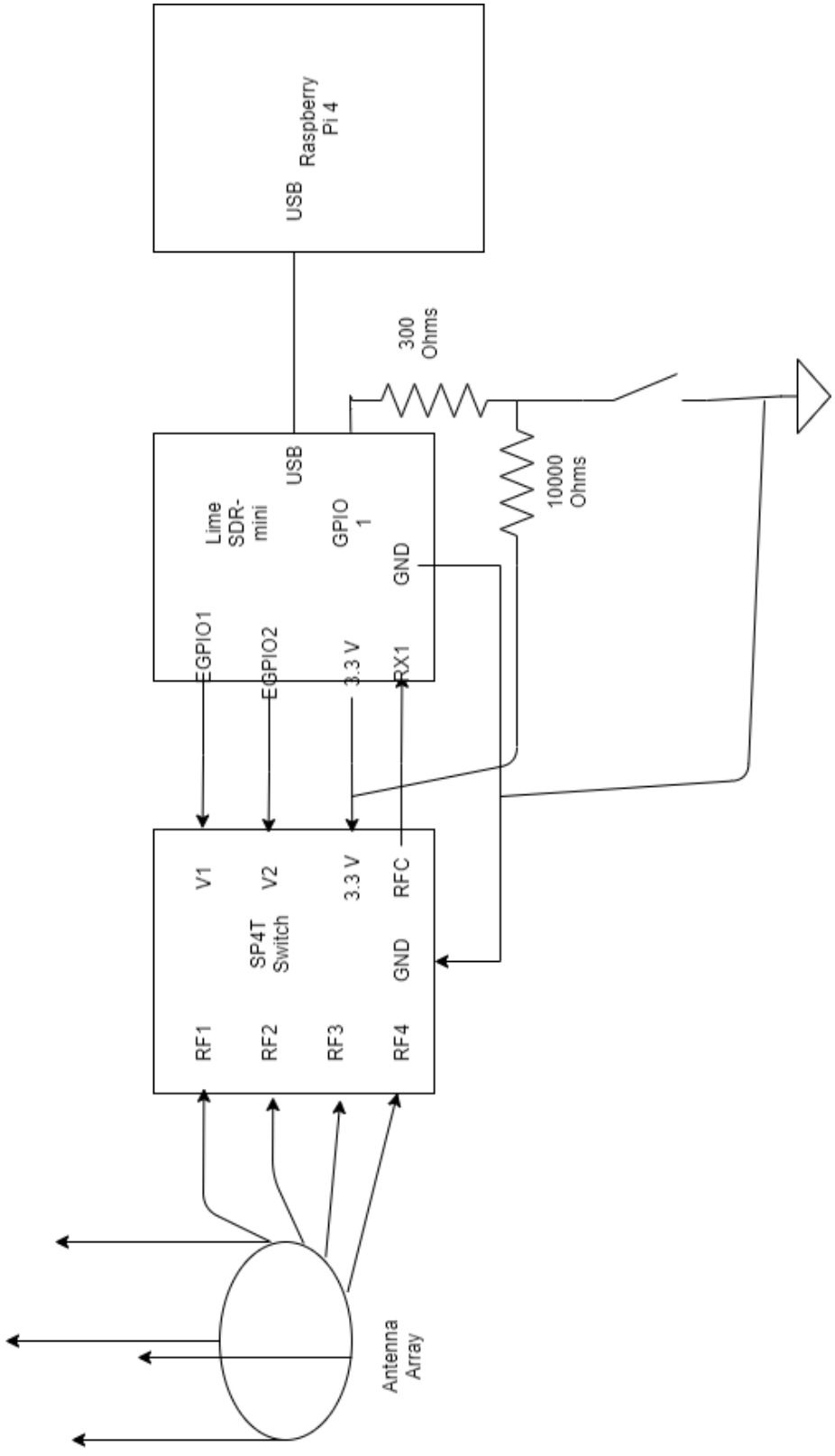
The future work will involve antenna array designs, RF switches and algorithms to find the valid angle more efficiently. Future work also includes improving the algorithm for both AOA measurement and localization.

This thesis proved to be a great experience. Not only it was a challenge but also platform to learn the theories behind previous direction finding systems.

Appendices

A Electrical Connection

The following circuit shows the electrical connections between various hardware components.

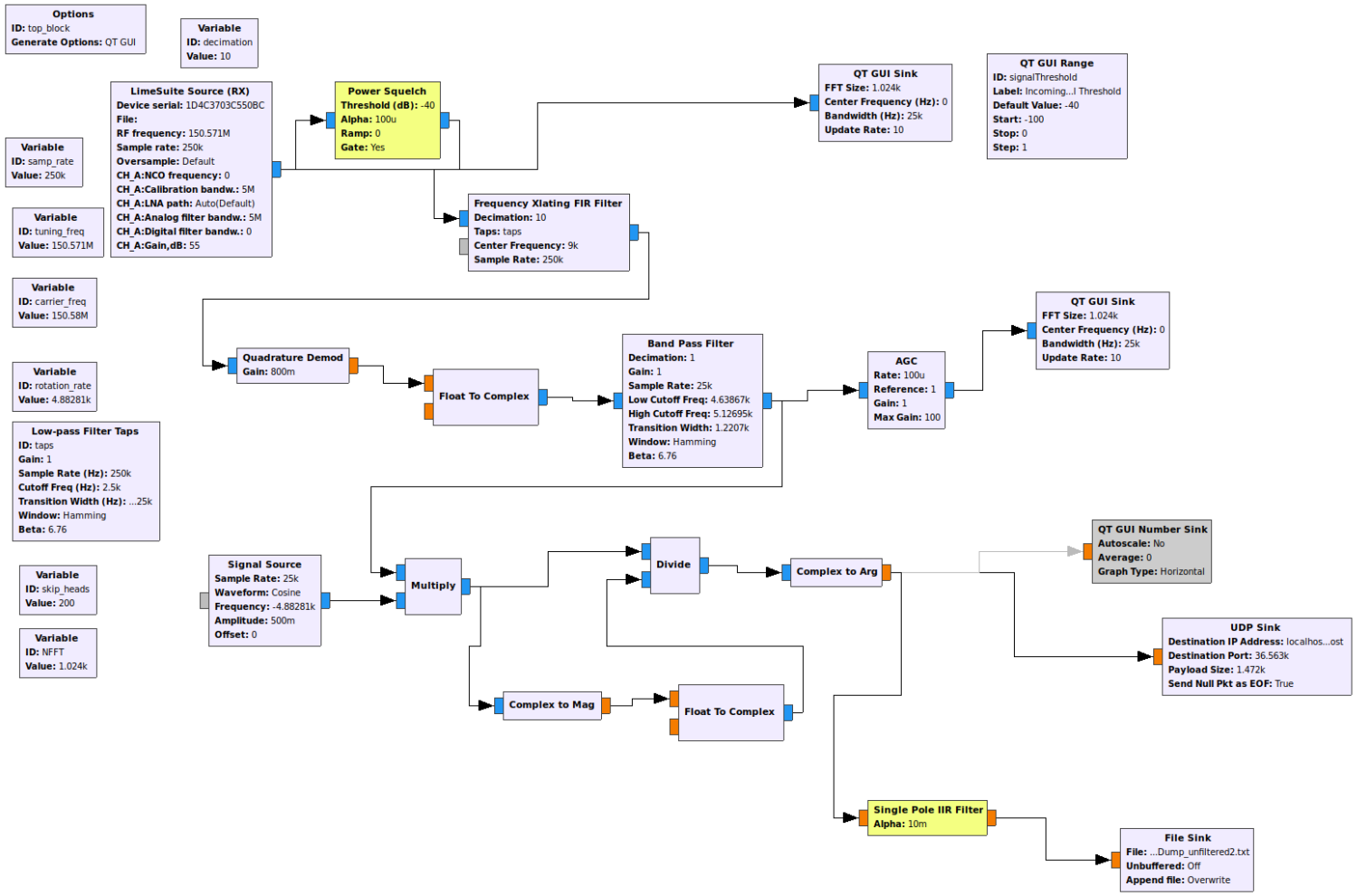

Raspberry Pi 4
USB
300 Ohms
10000 Ohms
USB
Lime SDR-mini
GPIO 1
GND
EGPIO1
EGPIO2
3.3 V
RX1
V1
V2
3.3 V
RFC
SP4T Switch
GND
RF1
RF2
RF3
RF4
Antenna Array

B Bill Of Materials

Item	Part/Part Number	Cost	Quantity	Total Cost	Manufacturer
1	LimeSDR-mini	$150.00	1	$150.00	Lime microsystems
2	Raspberry Pi 4	$45.00	1	$45.00	Raspberry Pi 4
3	Power Bank	$39.9	1	$39.9	AUKEY
4	Miscellaneous	$0.00	1	$0.00	GVSU

C GRC flow graph

The flow graph is the implementation of the pseudo Doppler algorithm on GRC software.


Options
ID: top_block
Generate Options: QT GUI
Variable
ID: decimation
Value: 10
LimeSuite Source (RX)
Device serial: 1D4C3703C550BC
File:
RF frequency: 150.571M
Sample rate: 250k
Oversample: Default
CH_A:NCO frequency: 0
CH_A:Calibration bandw.: 5M
CH_A:LNA path: Auto(Default)
CH_A:Analog filter bandw.: 5M
CH_A:Digital filter bandw.: 0
CH_A:Gain,dB: 55
Power Squelch
Threshold (dB): -40
Alpha: 100u
Ramp: 0
Gate: Yes
QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 25k
Update Rate: 10
QT GUI Range
ID: signalThreshold
Label: Incoming...l Threshold
Default Value: -40
Start: -100
Stop: 0
Step: 1
Variable
ID: samp_rate
Value: 250k
Variable
ID: tuning_freq
Value: 150.571M
Frequency Xlating FIR Filter
Decimation: 10
Taps: taps
Center Frequency: 9k
Sample Rate: 250k
Variable
ID: carrier_freq
Value: 150.58M
Quadrature Demod
Gain: 800m
Float To Complex
Band Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 25k
Low Cutoff Freq: 4.63867k
High Cutoff Freq: 5.12695k
Transition Width: 1.2207k
Window: Hamming
Beta: 6.76
AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 100
QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 25k
Update Rate: 10
Variable
ID: rotation_rate
Value: 4.88281k
Low-pass Filter Taps
ID: taps
Gain: 1
Sample Rate (Hz): 250k
Cutoff Freq (Hz): 2.5k
Transition Width (Hz): ...25k
Window: Hamming
Beta: 6.76
Signal Source
Sample Rate: 25k
Waveform: Cosine
Frequency: -4.88281k
Amplitude: 500m
Offset: 0
Multiply
Divide
Complex to Arg
QT GUI Number Sink
Autoscale: No
Average: 0
Graph Type: Horizontal
UDP Sink
Destination IP Address: localhos...ost
Destination Port: 36.563k
Payload Size: 1.472k
Send Null Pkt as EOF: True
Variable
ID: skip_heads
Value: 200
Variable
ID: NFFT
Value: 1.024k
Complex to Mag
Float To Complex
Single Pole IIR Filter
Alpha: 10m
File Sink
File: ...Dump_unfiltered2.txt
Unbuffered: Off
Append file: Overwrite

D Software

The following code interfaces with the GRC via UDP and validates the angle estimate. It interfaces a compass GUI to show the angle estimate. The compass widget is based on [30].

```
1
2  import sys
3  from PyQt4.QtCore import *
4  from PyQt4.QtGui import *
5  import socket
6  import numpy as np
7  from threading import Thread
8  threshold = 0.25
9
10
11
12
13 UDP_IP = "127.0.0.1"
14 UDP_PORT = 36563
15 PAYLOAD = 1472
16 global sock
17 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
18 sock.bind((UDP_IP,UDP_PORT))
19
20
21 class CompassWidget(QWidget):
22
23 angleChanged = pyqtSignal(float)
24
25 def __init__(self, parent = None):
26
27 QWidget.__init__(self, parent)
28
29 self._angle = 0.0
30 self._margins = 10
31 self._pointText = {0: "0", 45: "45", 90: "90", 135: "135",
      180: "180",
32 225: "-135", 270: "-90", 315: "-45"}
33
34 def paintEvent(self, event):
35
36 painter = QPainter()
37 painter.begin(self)
38 painter.setRenderHint(QPainter.Antialiasing)
39
```

```
40 painter.fillRect(event.rect(), self.palette().brush(QPalette.
       Window))
41 self.drawMarkings(painter)
42 self.drawNeedle(painter)
43
44 painter.end()
45
46 def drawMarkings(self, painter):
47
48 painter.save()
49 painter.translate(self.width()/2, self.height()/2)
50 scale = min((self.width() - self._margins)/120.0,
51 (self.height() - self._margins)/120.0)
52 painter.scale(scale, scale)
53
54 font = QFont(self.font())
55 font.setPixelSize(10)
56 metrics = QFontMetricsF(font)
57
58 painter.setFont(font)
59 painter.setPen(self.palette().color(QPalette.Shadow))
60
61 i = 0
62 while i < 360:
63
64 if i % 45 == 0:
65 painter.drawLine(0, -40, 0, -50)
66 painter.drawText(-metrics.width(self._pointText[i])/2.0, -52,
67 self._pointText[i])
68 else:
69 painter.drawLine(0, -45, 0, -50)
70
71 painter.rotate(15)
72 i += 15
73
74 painter.restore()
75
76 def drawNeedle(self, painter):
77
78 painter.save()
79 painter.translate(self.width()/2, self.height()/2)
80 painter.rotate(self._angle)
81 scale = min((self.width() - self._margins)/120.0,
82 (self.height() - self._margins)/120.0)
83 painter.scale(scale, scale)
84
```

```
85  painter.setPen(QPen(Qt.NoPen))
86  painter.setBrush(self.palette().brush(QPalette.Shadow))
87
88  painter.drawPolygon(
89  QPolygon([QPoint(-10, 0), QPoint(0, -45), QPoint(10, 0),
90  QPoint(0, 45), QPoint(-10, 0)])
91  )
92
93  painter.setBrush(self.palette().brush(QPalette.Highlight))
94
95  painter.drawPolygon(
96  QPolygon([QPoint(-5, -25), QPoint(0, -45), QPoint(5, -25),
97  QPoint(0, -30), QPoint(-5, -25)])
98  )
99
100 painter.restore()
101
102 def sizeHint(self):
103
104 return QSize(150, 150)
105
106 def angle(self):
107 return self._angle
108
109 @pyqtSlot(float)
110 def setAngle(self, angle):
111
112 if angle != self._angle:
113 self._angle = angle
114 self.angleChanged.emit(angle)
115 self.update()
116
117 angle = pyqtProperty(float, angle, setAngle)
118
119
120
121
122
123 class Window(QDialog):
124 def __init__(self):
125 super().__init__()
126 self.currentAngle = 0
127 self.correctionAngle = 0
128 self.compass = CompassWidget(self)
129 self.button = QPushButton('Calibrate',self)
130 self.button.clicked.connect(self.clicked)
```

```
131 self.label = QLabel(self)
132 self.layout = QVBoxLayout(self)
133 self.layout.addWidget(self.compass)
134 self.layout.addWidget(self.label)
135 self.layout.addWidget(self.button)
136 self.setWindowTitle("Angle of Arrival")
137 self.resize(300,300)
138
139
140
141 @pyqtSlot()
142 def clicked(self):
143 if self.currentAngle >0:
144 self.correctionAngle = -1 * self.currentAngle
145 else:
146 self.correctionAngle = self.currentAngle
147 print (f"correctionAngle= %f" %(self.correctionAngle))
148
149 def update(self):
150 actualAngle = self.currentAngle + self.correctionAngle
151 if actualAngle < 360:
152 actualAngle = actualAngle + 360
153 if actualAngle > 360:
154 actualAngle = actualAngle - 360
155 angleText = "%f  Degrees" %(actualAngle)
156 self.compass.setAngle(actualAngle)
157 self.label.setText(angleText)
158
159
160
161 class ClientThread(Thread):
162
163 def __init__(self,UDP_IP,port,PAYLOAD,window):
164 Thread.__init__(self)
165 self.UDP_IP = UDP_IP
166 self.port = port
167 self.payload = PAYLOAD
168 self.window = window
169
170
171 def run(self):
172 while True:
173 global sock
174 data,addr = sock.recvfrom(PAYLOAD)
175 numpyarray = np.fromstring(data, dtype=np.float32)
176 #print(len(numpyarray))
```

```
177 if len(numpyarray) > 100:
178 angle_mean = np.mean(numpyarray)
179 std_dev= np.std(numpyarray)
180 if std_dev < threshold:
181 currentAngle = angle_mean * 180/np.pi
182 self.window.currentAngle = currentAngle
183 self.window.update()
184
185 '''
186 def checkforSignal(self,fft):
187 fs = 500e3
188 width = 20
189 #N = 100
190 #print(len(fft))
191 if len(fft)>=512:
192 psd = (1/len(fft)*fs) * np.square(np.absolute(fft))
193 N = np.argmax(psd)
194 lowerRegionIndex = int(N-width/2)
195 upperRegionIndex = int(N+width/2)
196 lowerRegion = psd[lowerRegionIndex:N-1]
197 higherRegion = psd[N+1:upperRegionIndex]
198 regionOfInterest = np.mean(lowerRegion) + np.mean(
      higherRegion)
199 centralFreq = psd[N]
200 #print(centralFreq)
201 #print(psd[N-1])
202 #print(psd[N+1])
203 snr = centralFreq/(regionOfInterest/2)
204 #print(snr)
205
206 if snr > 25:
207 print(np.argmax(psd))
208 print (snr)
209 if N == 256:
210 return 1,N
211 else:
212 return 0,0
213 else:
214 return 0,0
215 else:
216 return 0,0
217 '''
218
219 if __name__ == "__main__":
220
221
```

```
222
223 app = QApplication(sys.argv)
224 window = Window()
225 clientThread = ClientThread(UDP_IP,UDP_PORT,PAYLOAD, window)
226 clientThread.start()
227 window.show()
228 sys.exit(app.exec_())
```

The simulation of pseudo Doppler implemented on MATLAB is the following codes.
Main.m

```
fc = 150.08e6;  %%% carrier frequency
fs = 250e3;    %%% sampling frequency
radius = 0.05;
lamda = 299792458/fc;
radiusToLamba = radius/lamda;
A = 2*pi*radiusToLamba;
Na = 4;  %% number of Antennas
%antenna_dwell = 25/1000;
samples_per_antenna = 16;
samples_per_rotation = Na * samples_per_antenna;
rotationalFrequency = fs/samples_per_rotation;

delF = A*rotationalFrequency ; %max frequency deviation
mod_idx = delF / rotationalFrequency;
signal_frequency = 9e3; %%% total signal bandwidth

decimation = 16;


%Signal Generation
total_length_of_signal = 1 ; %%% observations signal = 1s
totalNumOfSamples = total_length_of_signal * fs;

signal =  complexCarrier(totalNumOfSamples,fs,
    signal_frequency);
signal = signal';
pulse_length = 200/1000;
number_of_samples_in_a_pulse = fs * pulse_length;
pulse_signal = [1*signal(1:number_of_samples_in_a_pulse),0*
    signal(number_of_samples_in_a_pulse+1 :end)];
%pulse_signal = signal;

antenna_position = [1, 0+1j, -1, 0-1j];

%AngleOfArrival = 45;

%Window Sizes
window_size = 25/1000; %%% window size = 25ms
overlap_size = 10/1000; %%% overlapping window size = 10ms

num_samples_in_window = window_size * fs; %% Number of
    samples in a window
num_samples_overlapped = overlap_size * fs; %%% Number of
    samples overlapped
```

```
41
42
43
44 %%% Arrays for Analysis
45 phaseStep = 15 ;
46
47 phaseCounter = 0;
48
49 antennaSwitchOut = zeros(1,totalNumOfSamples);
50 switchOverlappedSamples = 2;
51
52 totalIterations = totalNumOfSamples/samples_per_rotation;
53
54
55 total_windows = totalNumOfSamples/num_samples_in_window;
56 phaseDiffer= zeros(1,total_windows);
57 filteredDiffer=zeros(total_windows,round(
       num_samples_in_window/decimation-1,0));
58 hilberts = zeros(total_windows,round(num_samples_in_window/
       decimation - 1));
59 filteredSignal = zeros(360/phaseStep,round(
       num_samples_in_window/decimation-1));
60
61 varianceInBW_array = zeros(1,total_windows);
62
63
64 Angles =(-180:phaseStep:180);
65 %Angles = 90;
66 phaseDependentAngles = zeros(1,length(Angles));
67 phaseDependentAngles2 = zeros(1,length(Angles));
68 Angles = 90;
69 for AngleOfArrival=Angles
70 AOA = deg2rad(AngleOfArrival);
71
72 %Antennas rotation
73 start = 1;  %% start and stop index for the signal generation
74 stop = samples_per_antenna;
75 for l=0:totalIterations
76 for k=1:Na
77 %amplitude_0*e**(1j*(phase_0+((tau*radius*antenna_position
       [0]*(e**(1j*angle_0))).real/wavelength)))
78 antennaSwitchOut(start:(stop-switchOverlappedSamples)) =
       pulse_signal(start:(stop-switchOverlappedSamples)) * exp(1
       i* real((2*pi*radius*antenna_position(k)*(exp(1i*AOA))))/
       lamda);
79 antennaSwitchOut((stop-switchOverlappedSamples):stop) =
```

```
    pulse_signal((stop-switchOverlappedSamples):stop) * (exp(1
    i* real((2*pi*radius*antenna_position(k)*(exp(1i*AOA))))/
    lamda) + exp(1i* real((2*pi*radius*antenna_position(mod(k
    +1,4)+1)*(exp(1i*AOA))))/lamda));
if start >= (totalNumOfSamples - samples_per_antenna)
break
else
start = stop;
stop = stop+samples_per_antenna;
end
end
end



noiseSignal = awgn(antennaSwitchOut,10,'measured');
%noiseSignal = antennaSwitchOut;
%noise = wgn(length(antennaSwitchOut),1,0,'complex');
%noiseSignal = noise;
startSig = 1;
stopSig = num_samples_in_window;
%{
for z=1:total_windows
[phaseD,filteredD] = getPhaseRedux(noiseSignal(startSig:
    stopSig),rotationalFrequency,signal_frequency,fs,
    samples_per_rotation);
phaseDiffer(z) =  phaseD;
filteredDiffer(z,:) = filteredD;
if startSig >= (totalNumOfSamples-num_samples_in_window)
break
else
startSig = stopSig+1;
stopSig = stopSig + num_samples_in_window;
end
end
%}
angleOfInterest = 0;
%phase difference calculation for each window
for z=1:total_windows

%signalMag = abs(noiseSignal(startSig:stopSig));
%figure()
%plot(signalMag(1:200));

[phaseD,filteredD,varianceInBW,signalAvailable] = getPhaseNew
    (noiseSignal(startSig:stopSig),rotationalFrequency,fs,
```

```
    samples_per_rotation,decimation);
if signalAvailable == 1
angleOfInterest = phaseD;
end
phaseDiffer(z) =  phaseD;
filteredDiffer(z,:) = filteredD;
varianceInBW_array(z) = varianceInBW;
if startSig >= (totalNumOfSamples-num_samples_in_window)
break
else
startSig = stopSig+1;
stopSig = stopSig + num_samples_in_window;
end
end
%plot(phaseDiffer)
%%{
%phase = phaseDiffer > -360;
phaseCounter = phaseCounter + 1;
phaseDependentAngles(phaseCounter) = wrapTo180(phaseDiffer(1)
    - 215);
phaseDependentAngles2(phaseCounter) = wrapTo180(phaseDiffer
    (2)-215);
filteredSignal(phaseCounter,:) = filteredD;
angleOfInterest = wrapTo180(angleOfInterest);
%{
if mod(phaseCounter-1,10) == 0
disp(AngleOfArrival)
disp(mean(phaseDiffer))
end
%}
end
plot(Angles,phaseDependentAngles);
xlabel('Measured Angle of Arrival')
ylabel('Angle of Arrival')
%figure()
%plot(Angles,phaseDependentAngles2);
%xlabel('Measured Angle of Arrival')
%ylabel('Angle of Arrival')

figure()
plotter = filteredSignal(1,:);
plot(plotter(1:300))
for z = 1:360/phaseStep

plotter = filteredSignal(z,:);
hold on
```

```
plot(2.5*samples_per_rotation,plotter(2.5*
    samples_per_rotation),'r*')
end
hold off
%%}
%{
for z=1:total_windows
figure()
%OutString = sprintf("Avg-energy: %f ; Max-energy = %f",
    avg_energies(z),max_energies(z));
plot_fft(filteredDiffer(z,:),-fs/8,fs/8,1024,1,"Output")
end
%}
plott(y,-fs/8,fs/8,1024,4,'Output');
%}
```

getPhaseNew.m

```
function [phaseDifference,filteredData,fdemod,
    varianceInBW,signalAvailable] = getPhaseNew(signal
    ,refSignalFreq,sampling_frequency,fullsampRot,
    decimation)

%signal = diff(signal);
decimated = decimate(signal,decimation,60,'fir');
%differed = diff(decimated);
fdemod = fm_quad(0.8,decimated,length(decimated)-1);
cmplx_demod = hilbert(fdemod);
fdemod_diff = diff(fdemod);
%filteredData = bandpass(fdemod_diff,[refSignalFreq
    *0.95 refSignalFreq*1.05],sampling_frequency/4);
bandpassfilter = designfilt('bandpassfir','
    FilterOrder',20,'CutoffFrequency1',refSignalFreq
    *0.95,'CutoffFrequency2',refSignalFreq*1.05,'
    SampleRate',sampling_frequency/decimation);
filteredData = filter(bandpassfilter,fdemod);
groupdelay = grpdelay(bandpassfilter,length(fdemod),
    sampling_frequency/decimation);
delay = mean(groupdelay);
delay_correction = 2*pi*delay/(sampling_frequency/
    decimation);
%filteredData = filtfilt(bandpassfilter,fdemod_diff);
%filteredData_diff = diff(filteredData);
numOfSamples = length(filteredData);
%t = (0: 1 : numOfSamples-1) * (sampling_frequency/4)
    ;
%measuredLob = 334 - 1.8 * zeroCrossingIndex;
```

```
20          refSig = getSignal(numOfSamples,sampling_frequency/
                decimation,refSignalFreq,0);
21          %figure();
22          %plot(filteredData(fullsampRot:100+fullsampRot))
23          %hold on
24          %plot(refSig(fullsampRot:100+fullsampRot))
25          %hold on
26          %plot([-1;1]* envelope);
27          %hold off
28          %legend('Recovered Doppler Signal','Reference Signal
                ')
29          %plot(refSig(1:100))
30          phaseDifference = goertezel_phase(filteredData(2.5*
                fullsampRot:end),refSig(2.5*fullsampRot:end),
                refSignalFreq,sampling_frequency/decimation);
31          avg_energy = 0;
32          max_energy = 0;
33
34          [signalAvailable,varianceInBW] = checkForSignal(
                filteredData,refSignalFreq*0.95,refSignalFreq
                *1.05,1024,sampling_frequency/decimation);
35          %[phaseDifference,avg_energy,max_energy] =
                phdiffmeasure(filteredData,refSig,1024,
                refSignalFreq,sampling_frequency/4,
                delay_correction);
```

qdemod.py

```
1           function qdemod = fm_quad(d_gain,inSignal,
                noutput_items)
2           qdemod = zeros(1,noutput_items);
3           for i = 1:noutput_items
4           if i ~= 1
5           product = inSignal(i) * conj(inSignal(i-1));
6           qdemod(i-1) = d_gain * angle(product);
7           end
8           end
```

checkforsignal.m

```
1
2
3   function [signalAvailable,varianceInBandwidth] =
        checkForSignal(signal, lowerFrequency,higherFrequency,NFFT
        ,fs)
4
5   signalAvailable = 0;
```

```
index_low = ceil(lowerFrequency/fs * NFFT)+1;
index_high = ceil(higherFrequency/fs * NFFT) + 1;

psd = periodogram(signal,[],NFFT);
valuesOfInterest = psd(index_low+1:index_high-1) - 3;
valuesOfInterest(valuesOfInterest < 0) = 0;
%figure
%plot(valuesOfInterest)
%title(var(valuesOfInterest))
varianceInBandwidth = var(valuesOfInterest);
if varianceInBandwidth > 0.1
signalAvailable = 1;
end
content...
```

goertezel_phase.m

```
function getPhase = goertezel_phase(signal1,signal2,
    frequencyOfinterest,sample_rate)

N1 = length(signal1);
N2 = length(signal2);
freq_index1 = round(frequencyOfinterest/sample_rate * N1) +
    1;
freq_index2 = round(frequencyOfinterest/sample_rate * N2) +
    1;
dft_data1 = goertzel(signal1,freq_index1);
dft_data2 = goertzel(signal2,freq_index2);
phase1 = atan2(imag(dft_data1),real(dft_data1));
phase2 = atan2(imag(dft_data2),real(dft_data2));

getPhase = rad2deg(phase1-phase2);
```

E Antenna Control Signal

In order to generate the control signals for RF switch from the SDR modify the FPGA for the LimeSDR. The code is open source and can be found in their github LimeSDR-Mini_GW. The program is written in VHDL and can be edited via Intel's Quartus || software. The following process is added to the 'lms7_trx_to' program which is inside 'LimeSDR-Mini_lms7_trx' folder. The program requires to route GPIO 1 as switch_reset. In case reset switch is not needed, it can be skipped. The program can be directly downloaded to the LimeSDR FPGA via a USB Blaster or through LimeSuite program. It is preferred to use the LimeSuite program as it reduces the effort to install additional pins on the LimeSDR. Use the bit stream settings under 'LimeSDR-Mini_bitstreams' to convert the output result to format required by LimeSuite. The bit stream settings are under 'LimeSDR-Mini_bitstreams' folder.

```
1
2  process(LMK_CLK, switch_reset)
3
4  begin
5  if(rising_edge(LMK_CLK)) then
6  if(temp="1111111111111" or switch_reset = '0') then
7  temp <= "0000000000000000";
8  else
9  temp <=temp+1;
10 end if;
11 end if;
12 end process;
13 FPGA_EGPIO(1) <= temp(12);
14 FPGA_EGPIO(0) <= temp(11);
```

References

[1] D. Roy, *ESPRIT: Estimation of Signal Parameters via Rotational Invariance Techniques*. PhD thesis, Standford University, 1987.

[2] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. On Antennas and Propagation*, vol. AP-34, March 1986.

[3] R. Schmidt, *A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation*. PhD thesis, Standford University, 1981.

[4] J. J. Keaveny, *Analysis and Implementation of Novel Single Channel Direction Finding Algorithm on a Software Radio Platform*. PhD thesis, Virginia Polytechnic Institute, 2005.

[5] R. Rainer and Bumasser, "An approach to hf tactical radio direction finding and signal monitoring," *Journal of Electronic Defense*, October 1983.

[6] D. Peavey and T. Ogunfunmi, "The single channel interferometer using a pseudodoppler direction finding system," *Proc. of 1997 IEEE Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 4129–4132, 1997.

[7] R. Hammerle, "Factors limiting the accuracy of doppler and adclock direction finding systems," *Proc. of 1989 IEEE Colloquium on Passive Direction Finding*, January 1989.

[8] U. G. Survey, "A critique of wildlife radio-tracking and its use in national parks." `http://www.npwrc.usgs.gov/resource/wildlife/radiotrk/convent.html`, 2006.

[9] W. Act, "Gps and vhf tracking collars used for wildlife monitoring." `https://wildlifeact.com/blog/gps-and-vhf-tracking-collars-used-for-wildlife-monitoring`, 2020.

[10] K. VonEhr, S. Hilaski, B. E. Dunne, and J. Ward, "Software defined radio for direction-finding in uav wildlife tracking," *2016 IEEE International Conference on Electro Information Technology (EIT)*, pp. 0464–0469, 2016.

[11] P. J. D. Gething, *Radio Direction Finding and the Resolution of Multicomponent Wave-Fields*. Peter Peregrunus Ltd, 1986.

[12] Y. Aoyan, "Doppler effect." `https://www.pinterest.com/yangaoyan/doppler-effect/`, 2018.

[13] M. Kossor, "A doppler radio-direction finder: Build this version of a time-honored transmitter-hunting tool.," *QST-NEWINGTON-83*, pp. 35–40, 1999.

[14] B. Babjak, S. Szilvasi, and P. Volgyes, "On accurate, low-complexity quasi

doppler based localization," *The Third Inter-national Confo-ence on Digital Information and CommunicationTechnology and its Applications (DICTAP2013)*, pp. 85–92, 2013.

[15] S. Haykin and M. Moher, *Introduction to Analog and Digital Communnications*. Wiley, 2006.

[16] W. Terrence Rogers, "A dopplescant," *QST*, pp. 24–28, May 1978.

[17] P. Denisowki, "A comparison of radio-direction finding technologies," *presentation, Department of Energy Spectrum Management Conference*, 2011.

[18] picoDopp, "About doppler dfs." `http://silcom.com/~pelican2/PicoDopp/ABOUT_DOPP.html`, 2020.

[19] R. B. MacCurdy, R. M. Gabrielson, and K. A. Cortopassi, "Automated wildlife radio tracking," *Handbook of Position Location: Theory, Practice, and Advances, chapter 33*, 2012.

[20] M. S. Sharawi and D. N. Aloi, "Characterizing the performance of single-channel pseudodoppler direction findin systems at 915 mhz for vehicle localization," *International Journal of Communication Systems 24.1*, pp. 27–39, January 2011.

[21] D. N. Aloi and M. S. Sharawi, "Modeling and validation of a 915 mhz single channel pseudo doppler direction finding system for vehicle applications," *2009 IEEE 70th Vehicular Technology Conference Fall*, 2009.

[22] M. C. E. Stieber, "Radio direction finding network receiver design for low-cost public service applications," 2012.

[23] PA8W, "What is a (pseudo-) doppler rdf?." `https://radiodirectionfinding.wordpress.com/wat-is-een-amplitude-rdf/`, 2020.

[24] SPENCH, "Sdrdf." `https://wiki.spench.net/wiki/SDRDF`, 2020.

[25] B. Benchoff, "Shmoocon: Delightful doppler direction finding with software defined radio." `https://hackaday.com/2018/01/23/shmoocon-delightful-doppler-direction-finding-with-software-defined-radio/`, 2020.

[26] H. Won, Y.-K. Hong, K. Isbell, L. Vanderburgh, J. Platt, and M. Choi, "Evaluation on pseudo-doppler antenna array using software-defined-radio," *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, 2019.

[27] M. Kebeli, "Extended symmetrical aperture direction finding using correlative interferometer method," *2011 7th International Conference on Electrical and Electronics Engineering*, 2011.

[28] Rhode and Schwarz, "Introduction into theory of direction finding," *Radiomonitoring and Radiolocation*, 2011.

[29] LimeMicro-systems, "Lime sdr mini." `https://limemicro.com/products/boards/limesdr-mini/`, 2019.

[30] PyQt, "Compass widget." `https://wiki.python.org/moin/PyQt/Compass%20widget`, 2020.

[31] Z. Zhang, "Direction of radio finding via music (multiple signal classification) algorithm for hardware design system," *Journal of Physics: Conference Series 910 012017*, 2017.

[32] KerberosSDR, "Kerberossdr a 4x phase coherent rtl-sdr for passive radar, direction finding and more." `kerbrossdr.com`, 2020.

[33] KerberosSDR, "Kerberossdr." `http://kerberossdr.com/`, 2020.