TRACKING MIGRATORY BIRDS: APPLYING A PASSIVE TRACKING TECHNIQUE USING DIRECTION OF ARRIVAL FROM VHF RADIO TAGS

Jesse Moore
University of Rhode Island, jessetaylormoore@gmail.com

TRACKING MIGRATORY BIRDS: APPLYING A PASSIVE TRACKING TECHNIQUE USING DIRECTION OF ARRIVAL FROM VHF RADIO TAGS

BY

JESSE MOORE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

OCEAN ENGINEERING

UNIVERSITY OF RHODE ISLAND

2020

MASTER OF SCIENCE THESIS

OF

JESSE MOORE

APPROVED:

Thesis Committee:

Major Professor ______________________
Dr. James H. Miller

____
Dr. Gopu Potty

____
Dr. Peter Paton

____
Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2020

ABSTRACT

As the offshore wind power industry grows along the U.S Atlantic Coast, biologists need to assess potential conflicts between birds and wind turbines. There is a pressing need to develop accurate tracking systems that can evaluate movements of volant (flying) organisms near offshore wind facilities during both the pre-construction and post-construction phase in U.S. waters. Based on research in Europe, turbines at offshore wind facilities can present a collision risk to birds, as well as a migration barrier. Available tracking technology has difficulties tracking fine-scale temporal and spatial movements of small ($<$200 g) volant organisms. The functionality of a direction finding (DF) system that uses phase differences to accurately obtain bearings from a 3-element omni-directional antenna array was tested. Each antenna was equally spaced (0.889 m) by a distance of a half-wavelength. The open source Software Defined Radio (SDR) software on a PC integrates the FUNCube dongle (FCD) by setting the tuner frequency and sample rate. The FCD demodulates the received signals and converts them to a digital signal after a series of filtering steps. Once converted, the data is saved to a .wav file for post processing using MATLAB. To test the accuracy of this prototype tracking system, a series of drone flights were initiated, with a digitally-coded VHF transmitter attached, to test the viability of this system. However, one receiver failed due to PC memory issues thus the results for estimating bearings were limited to two antennas. For purposes of algorithm testing, estimates were shifted to the correct quadrants ($0° - 180°$ or $180° - 360°$) based on *apriori* information to compensate for the ambiguous bearing estimates from using only two antennas. At very short ranges, bearing accuracy suffered due to the directivity of the antennas. The system performed best ($\pm 6°$) when the transmitter was farther than 175 m

from the array, relatively well ($\pm 15^\circ$) when the transmitter was over 100 m away from the array, and poorly ($\pm 50^\circ$) when within 100 m of the antenna array. The research did show it was feasible to track azimuth angles through phase measurements. With further developments to the current design, it could be feasible to design an antenna array and continuous receiver to monitor fine-scale movements of VHF-tagged birds using phase measurements.

ACKNOWLEDGMENTS

Being a part of the M.S. program at URI was never a part of my plan but after these two years, I can't imagine my life differently. I am full of gratitude and happiness to have learned all that I have, been challenged by my work and supported by my advisors and peers. This work has challenged my ability to wrap my mind around different topics and I couldn't have done it without my advisors constant excitement, engagement and guidance. Thank you for believing in me, being patient with me, and being great role models for my next steps in the engineering world. I also would like to acknowledge Fred Pease, for not only spending countless hours on designing and building the antenna system but filling my time in Middleton with smiles, knowledge and great stories. This would not have been possible without his help. Finally, my friends for still being my friends when I disappear for days working on my research but still responding when I needed them. Your friendship made this all much easier. A special thank you to Russ Shomberg for answering any and all my RADAR questions, I have learned so much from you.

Contents

List of Figures

Figure **Page**

List of Tables

CHAPTER 1

Introduction

With the expected rapid growth of the offshore wind power industry along the U.S. Atlantic Coast, there is a pressing need to assess potential impacts of these new facilities on volant (flying) wildlife. One approach is developing a system to accurately track the flight paths of birds and bats with digitally-coded transmitters during both the pre-construction and post-construction stages to assess the collision risk and barrier effects.

1.1 Motivation

In Rhode Island, the Governor has set the ambitious goal of having the state powered by 100% renewable energy by 2030 [1], which would make the state the first in the country to be exclusively all renewable energy. One of the primary means to reach that goal is offshore wind energy. Rhode Island currently has the only offshore wind energy facility in the U.S., a 5-turbine 30 MW facility off of Block Island. There are a number of Bureau of Ocean Energy Management (BOEM) lease blocks in southern New England waters that could provide offshore wind energy to Rhode Island [2]. By 2030, wind power has the potential to supply up to 20% of global electricity and reach an annual installed capacity of 8.4 GW [3]. With wind energy companies searching for more efficient systems, such as floating turbines for offshore wind energy set to be commercialized by 2030 [4], they are in the planning stages of constructing offshore wind turbines in federal waters. However, with every energy production comes some potential negative impacts, including with volant wildlife. Offshore wind energy facilities have been operational since 1990 in Europe [5, 6, 7]. European researchers have found that collision mortality, habitat loss and alteration, and behavioral avoidance of wind facilities were the three potential impacts of wind energy facilities on volant wildlife [8, 2]. The research in this thesis can hopefully be used as a tool to monitor the movements of local and migratory birds in offshore waters where wind facilities are proposed or constructed.

1.2 Thesis Content

Chapter 2 is a literature review that summarizes research methods and findings for both radio direction finding (RDF) systems and avian tracking. These studies provide insight into the strengths and weaknesses the current tracking technologies and the science behind those methods. Chapter 3 describes the methods, study design, and limitations of this research project. Chapter 4 explains an experiment to test the accuracy of using phase measurements to estimate bearings to transmitters by flying a drone with a test tag over a predetermined flight path. Chapter 5 describes the post-processing methods of the data obtained from the experiments. Chapter 6 summarizes the results of the experiments. Finally, Chapter 7 discusses the results and limitations of this approach, sources of error and future design. Finally, a bibliography is provided.

List of References

[1] "Raimondo sets goal for 100[Online]. Available: https://www.ri.gov/press/view/37527

[2] P. Loring, P. Paton, J. McLaren, H. Bai, R. Janaswamy, H. Goyert, and P. Sievert, "Tracking offshore occurrence of common terns, endangered roseate terns, and threatened piping plovers with vhf arrays," *Sterling (VA): US Department of the Interior, Bureau of Ocean Energy Management. OCS Study BOEM*, vol. 17, p. 140, 2019.

[3] V. Hamadi, U. Brosnan, I. Loftus, and G. Montgomery, "Offshore substation design: High-level overview of the industry best practices," *IEEE Power and Energy Magazine*, vol. 17, no. 4, pp. 67–74, 2019.

[4] E. R. M. Shouman, “Global prediction of wind energy market strategy for electricity generation,” in *Modeling, Simulation and Optimization of Wind Farms and Hybrid Systems*. IntechOpen, 2020.

[5] M. D. Esteban, J. J. Diez, J. S. López, and V. Negro, “Why offshore wind energy?” *Renewable Energy*, vol. 36, no. 2, pp. 444–450, 2011.

[6] I. Staffell and R. Green, “How does wind farm performance decline with age?” *Renewable energy*, vol. 66, pp. 775–786, 2014.

[7] R. J. Barthelmie, “A brief review of offshore wind energy activity in the 1990’s,” *Wind Engineering*, pp. 265–273, 1998.

[8] A. Fox, M. Desholm, J. Kahlert, T. K. Christensen, and I. Krag Petersen, “Information needs to support environmental impact assessment of the effects of european marine offshore wind farms on birds,” *Ibis*, vol. 148, pp. 129–144, 2006.

CHAPTER 2

Literature Review

This thesis research requires knowledge and background in Radio Direction Finding (RDF), antennas, signal characteristics including phase, and the current tracking systems used by ornithologists. This chapter is divided into sections that describe approaches to RDF, as well as current approaches to tracking wildlife. Much of the RDF section is based on research compiled by Jenkins [1991] for general RDF and by Salido-Monzú et al. [2016] which investigates phase data errors. The wildlife tracking sections summarizes standard methods of bird tracking, some recent emerging methods tested and some methods applied specifically tracking volant wildlife near offshore wind turbines. Based on this thorough literature review, the technology and analytical approaches implemented in this thesis are novel. However, there are still accuracy and instrumentation issues with this novel approach that need to be addressed, as with other tracking technologies, before it can be implemented at a broader scale.

2.1 Radio Direction Finding: Phase Delay

RDF requires a passive receiving system that extracts information from passing electromagnetic radio waves from a very high frequency (VHF) transmitter to estimate its direction of arrival (DOA). There are three main measurement methods using RDF to obtain DOA: amplitude response, phase delay and time delay [1].

$$\tau = \frac{2\pi d}{\lambda} sin(\phi) \qquad (1)$$

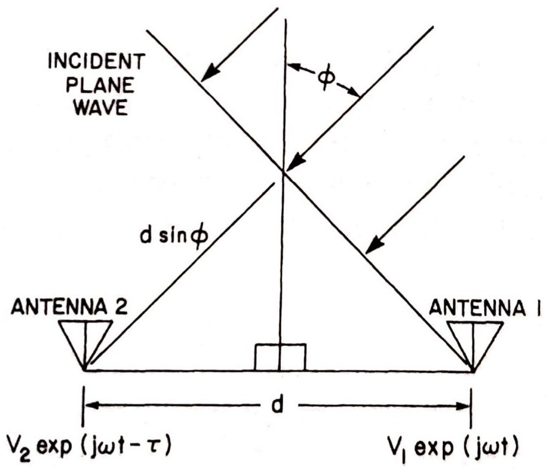


Figure 1: Schematic showing Radio Detection Finding based on phase delay measurements [1].

Phase delay measurements require at least two antennas (Figure 1), of any kind but work best with omni-direcitonal, separated by distance, d, and are calculated using Equation 1. As an incident plane wave arrives at angle, ϕ, it first reaches antenna 1. The wave then travels the distance of $dsin(\phi)$ before reaching antenna 2. The voltages induced at antenna 1 and 2 are $V_1 \exp j\omega t$ and $V_2 \exp j\omega t$ respectively. Therefore, the bearing angle ϕ is a function of phase delay.

2.1.1 Two Antennas

When designing a phase comparison system, the spacing between antennas spacing must have a maximum of $\lambda/2$, where λ is the wavelength of the transmitter signal, limiting the maximum phase shift δ to be within π radians [1]. DOA precision can be shown as a function of $\Delta\delta/\Delta\phi$, phase delay over bearing, with a unit of degrees per degrees.

$$\theta = \arcsin \sqrt{\delta_1^2 + \delta_2^2} \frac{\lambda}{2\pi d} \tag{2}$$

Equation 2 depicts calculations needed to obtain azimuth angle, where δ_1 is phase

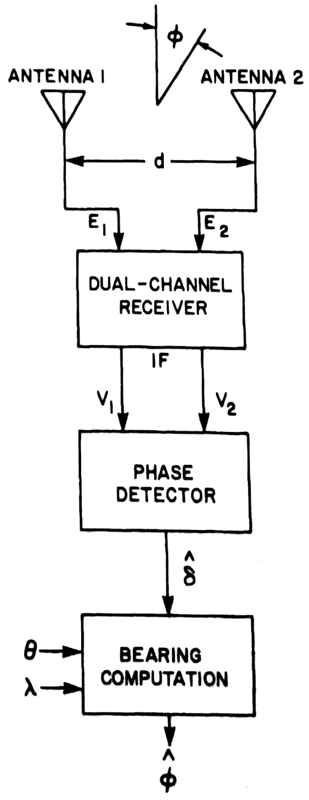


Figure 2: Process to obtain a bearing from a remote VHF transmitter using a dual-channel two antenna system using phase measurements [1].

delay for the respective antenna, d is distance between antennas and λ is wavelength.

The phase detectors can output either: (1) analog measurements by summing the analog vectors, (2) digital versions of the sine and cosine waves or (3) passed to a phase comparator (CRT) for visual outputs. Typically, the analog phase measurement accuracy decreases when there are frequency variations and low signal-to-noise ratio (SNR), while digital phase measurements tend to be more stable.

To minimize phase errors, the unmodulated carrier frequency should be set to the center frequency. Phase errors are introduced to the system when there are

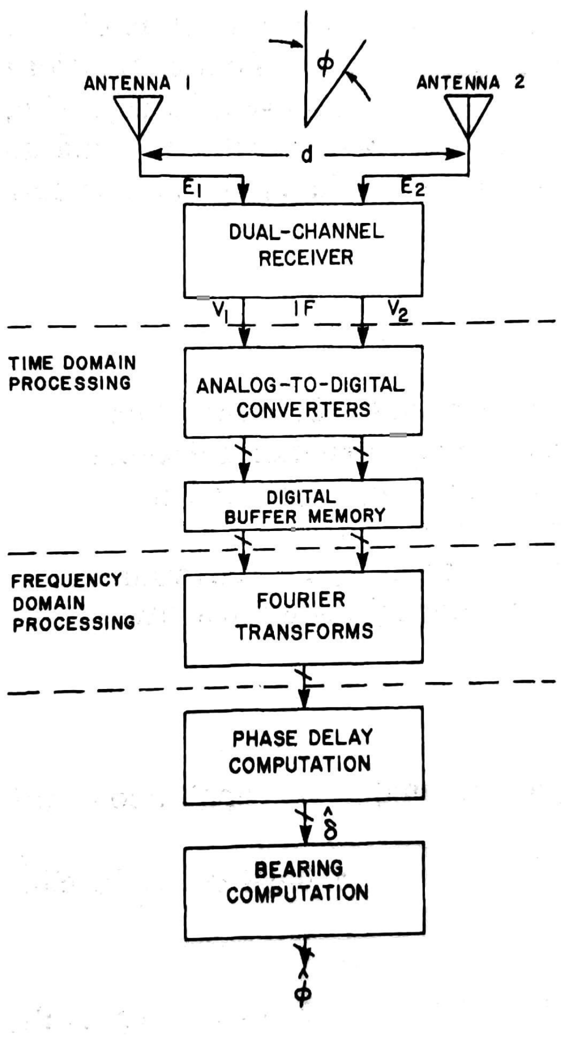


Figure 3: Steps needed to estimate signal bearing using the Fourier transform method [1].

frequency-shift-keyed (FSK) signals, where a fast Fourier transform (FFT) would show asymmetrical spectral features for both antennas but with differently shifted signals [1]. Phase averaging can be implemented to reduce these phase errors.

To increase the bearing accuracy of two-antenna systems that include neither information about elevation nor the use of a phase detector, the FFT method is a good alternative. This method is implemented in this thesis and the general steps (Figure 3) are:

1. Perform the time-domain processing
2. Process the frequency data
3. Compute phase-delay
4. Compute bearing

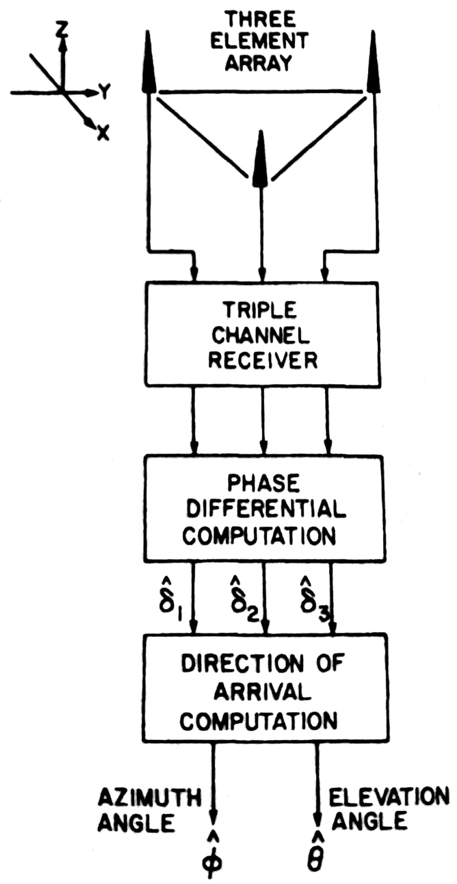
Step 1 consists of converting the analog signal to digital signal conversion and then storing digital samples, which then are used for performing a Fourier transform. Then the real and imaginary components can be used to calculate phase using the Equation 3.

$$\gamma = \arctan(i/r) \tag{3}$$

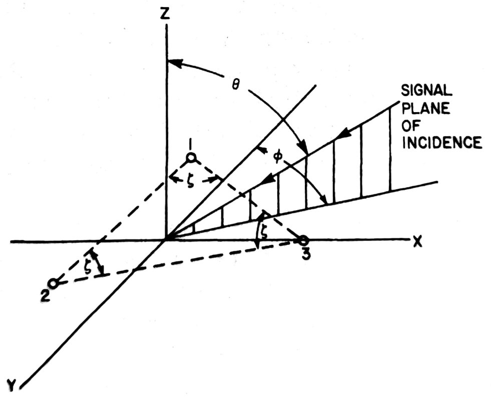
The Fourier transform method has many advantages including (1) reduced signal amplitude variations, (2) improved sensitivity (i.e., up to 30 dB), (3) options for frequency and phase corrections, (4) ideally suited for short duration bursts, (5) interference can be limited by eliminating the undesired spectral components, and (6) digital can be stored and adapted through extra post-processing. This technique is adept for short duration uncooperative transmissions in dense electromagnetic environments [1].

2.1.2 Three Antennas

Phase interferometry typically includes three antenna elements, set up in an equilateral triangle, and a triple-channel receiver [1]. For this approach, each antenna is connected to the same receiver to maintain system coherence, which eliminates antenna switching and produces phase measurements and DOA calculations in real time (Figure 4a). This system can estimate both azimuth and elevation of the target with similar strengths as a two-antenna system that has with the Fourier transform method implemented. The limitations of a three-antenna system in-



(a) Process to estimate a bearing using the Fourier transform method.



(b) Geometry of triple channel receiver, showing azimuth, elevation and relative angles of antennas $\zeta = 60°$.

Figure 4: Schematics for understanding the processing and geometry of a triple-channel receiver [1].

clude the necessity for a highly intensive computational capability for calculating the phase and a signal calibration injected into the system [1].

$$\delta_{1-2} = (2\pi d/\lambda)cos(\phi)sin(\theta) \tag{4a}$$

$$\delta_{3-1} = (2\pi d/\lambda)cos(\phi - 120°)sin(\theta) \tag{4b}$$

$$\delta_{2-3} = (2\pi d/\lambda)cos(\phi - 240°)sin(\theta) \tag{4c}$$

Equations 4a - 4c calculate phase differentials for a three-antenna system with a triple-channel receiver for phase interferometry.

$$\theta = \arcsin\left(\sqrt{\delta_{1-2}^2 + \delta_{3-1}^2 + \delta_{2-3}^2}/[(2\pi d/\lambda)(\sqrt(3/2))]\right. \tag{5a}$$

$$\phi_{1-2} = \arctan\left(2\delta_{2-3} + \delta_{1-2}\right)/[(\sqrt(3))(\delta_{1-2})] \tag{5b}$$

$$\phi_{1-3} = \arctan\left(2\delta_{3-1} + \delta_{1-2}\right)/[(\sqrt(3))(\delta_{1-2})] \tag{5c}$$

$$\phi_{2-3} = \arctan\left(2\delta_{2-3} + \delta_{3-1}/[(\sqrt(3))(\delta_{2-3} + \delta_{3-1})]\right. \tag{5d}$$

Both azimuth and elevation can be estimated using three calculated azimuths angles references to the three baselines seen in Equations 5a - 5d, where θ is elevation and ϕ is azimuth as shown in Figure 4b.

With the ϕ_{i-j} values, summing them estimates the azimuth DOA using Equation 6.

$$\phi = \arctan \frac{\cos(\phi_{1-2}) + \cos(\phi_{1-3}) + \cos(\phi_{2-3})}{\sin(\phi_{1-2}) + \sin(\phi_{1-3}) + \sin(\phi_{2-3})} \tag{6}$$

Three methods of reducing error are using (1) three antenna systems to remove random measurement errors, (2) a calibration source as a common local oscillator between the antennas for phase mismatch errors, and (3) create correction tables based on errors from scattering and coupling.

2.2 SNR Phase Degradation

In order to obtain near-field direction of arrival estimation, one could apply phase differences between multiple signals [2]. One approach is using a quadrature phase detector which works in the digital domain after being converted using an ADC converter [2]. Salido-Monzú et al. [2016] focuses on the degradation of the phase value and how it occurs.

Sampling radio signals can be intensive on an acquisition system. Salido-Monzú et al. [2016] chose to purposefully undersample a signal to reduce those requirements on the acquisition and concluded that it can be beneficial but causes some phase estimation uncertainty. The phase degradation results from noise aliasing that causes low SNR and instabilities in the frequency response.

To reduce the noise aliasing effects, Salido-Monzú David et al. [2016] proposes a more restrictive anti-aliasing filtering of the analog signal. However, this would cause more phase uncertainty due to drifts.

$$\Delta\phi_{H(jf)}\Big|_{\Delta f_0, f\approx f_0} \approx 2Q\frac{\Delta f_0}{f_0} \tag{7}$$

$$\Delta\phi_{H(jf)}\Big|_{\Delta Q, f\approx f_0} \approx 2Q\frac{f_0 - f}{f_0}\frac{\Delta Q}{Q} \tag{8}$$

Equation 7 shows that variations in f_0 are greatly affected by the phase sensitivity based on Q, the quality factor of the filter. Equation 8 shows that both the quality factor is affected by the relationship between the quality factor and relative frequency deviation. By assuming that $f \approx f_0$, the sensitivity of phase can be related to the quality factor directly. With these equations the minimum stability of components and maximum quality value can be used to design the band-pass filter.

The designed filter resulted in a constant phase shift for every received signal where the difference between the two signals is the relative phase difference between them and is a constant (Figure 5). Overall Salido-Monzú David et al. [2016] found that a bandpass filter, rather than a low pass filter, reduces the sampling speed by 60% with <29% loss of precision. This is important for reducing digitization system

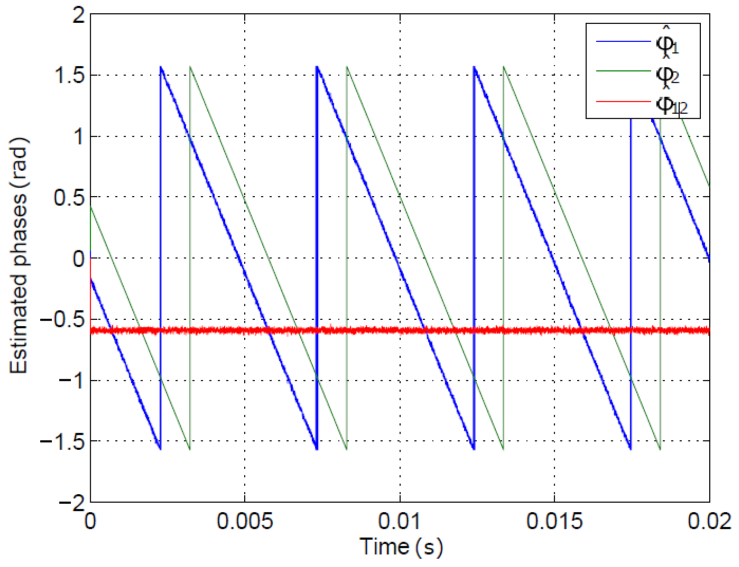


Figure 5: Output of phase meters [2].

cost and computing load, which relatively reduces loss of precision.

2.3 Avian Tracking

Ornithologists have been tracking birds with VHF transmitters for decades [3, 4, 5, 6]. Radio telemetry has been implemented to include small animals with high temporal and spatial precision, as there are limitations due to tag size (e.g 3.5 grams being too heavy for animals <100 g). However, with the advancements over the years, there are now tags that weigh as little as 0.3 g and as a result VHF technology remains one of the only options for tracking of smaller birds [7]. Transmitters weigh <0.3 - 2 g, with burst intervals every 3 - 29 seconds, and last for long time durations (19 – 359 days) depending on battery size and signal intervals [8]. In addition, automated receivers were developed that could collect and store these data from thousands of individual organisms [9]. Using this technological advance, the Motus network was developed to track the movements to hundreds of birds with stations located across North America [9].

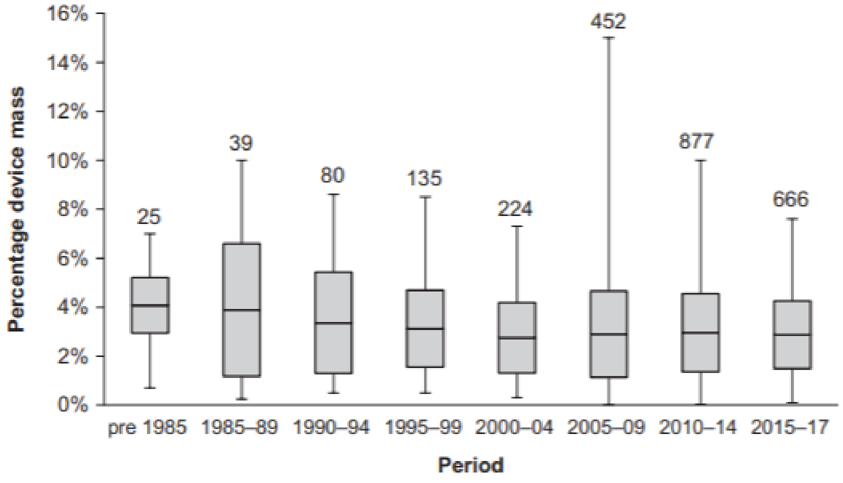
Summaries of tracking techniques that are commonly implemented including a number of new studies and wind turbine specific studies are provided.

2.3.1 Wildlife Tracking Systems

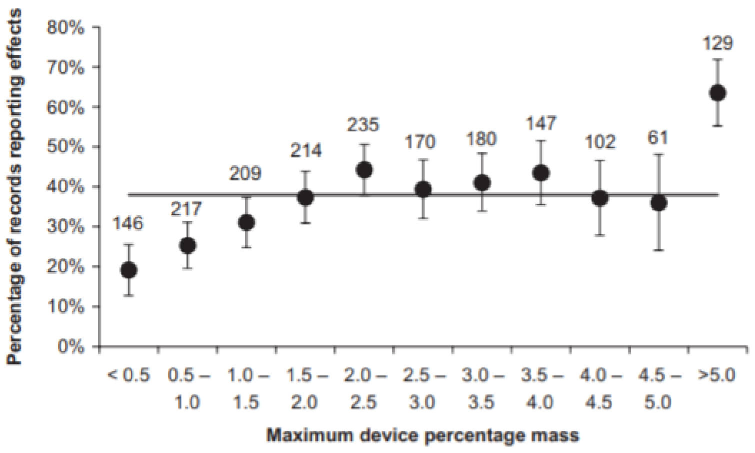
In the past, VHF wildlife tracking systems consisted of a handheld receiver that the operator would tune to the expected carrier frequency of the tag, and manually rotate a 3-5 element yagi antenna until the maximum audible tone was detected [6]. However this requires a relatively loud, long tone for a human to detect, which typically signals lasting 20 ms. The operator would then move to a new position, manually determine signal bearing, and then use triangulation to estimate the position. However, this is time intensive, requires detections from at least two locations to triangulate the tag position and is often inaccurate if the bird with the transmitter is in flight [6]. Recently engineers and biologists have developed automated telemetry systems that can track a large the number of birds simultaneously [5, 6].

Alternative tracking methods include satellite relay systems, GPS tags, geolocator dataloggers and radar [5]. Below include some advantages and disadvantages to these technologies.

Satellite tracking systems typically use Doppler effect (Argos transmitters) or GPS technology. A couple disadvantages of this method are (1) there are weight limits to the technology to mitigate the negative effects on wildlife and (2) there are limited numbers of fixes per day causing detecting a bird flying by a turbine at the ping to the satellite are low. More specifically, to avoid negative impacts on the tagged animals, the systems currently weight averages to 3%. The relation between body mass percentage and effects with measurements of body mass percentages used in studies can be seen in Figure 6. The 64% of device mass in terms of percentage of



(a) Mean percentage device mass for 5-yr periods with ± 1 SD shown by the boxes and the maximum and minimum values shown through whiskers. [10].



(b) Percentage of reported effects based on device percentage mass with errors bars showing 95% confidence limits [10].

Figure 6: Device mass percentage and reported effects based on device [10].

bird mass was 3% or less and only 6% of devices were >5%.

If a GPS logger is used, rather than a conventional GPS tag, the energy demands on the battery are high, limiting the total number of data transmissions (e.g., <100 stored in the unit for a 1 g tag [11]), thus have limited utility to accurate estimate migratory pathways or fine-scale movements near an offshore wind facility due to the limited number of fixes.

One of the less traditional but recent tracking methods is implementing short-range VHF transmitters and usually an array of antennas. These tags are usually coded with different frequencies and pulse patterns and is a common method used for the smaller birds as small as a hummingbird. The VHF tag costs are a relatively low-cost of $140-$300 per transmitter, but the system itself can cost up to a few thousand dollars. This method has been employed by the Motus Wildlife Tracking Network (Motus; https://motus.org).

Motus is a collaborative network that has deployed automated radio-telemetry arrays to track multiple individual species, like birds, in local to regional and hemispherical scales. They employ receivers that listen for a 166.380 MHz frequency. By limiting the frequency scan to one value, the probability of getting a detection improves greatly. Each receiver station includes a high-accuracy GPS sensor for precise time synchronization and geolocation of the receiver. The distance that a tag can be detected is based on the antenna at a given site. Close range tracking, up to 500 m, a small omni-directional antenna is stationed. For distances up to 2-3 km, 3-5 element Yagi antennas are set up. For longer range detections, 9-element Yagi antennas are implemented and can reach ranges of 15 km. Although the location data compared to GPS tags is less accurate, with higher temporal resolution the data is sufficient enough to estimate fine-scale behavior such as diel activity levels, departure time and timing of movements, orientation and flight speed [12].

There are also datalogging systems such as the solar geolocation tracking method which uses a photodiode and micro-controller that continuously tracks the light level as a function of time [6]. Latitude and longitude can be calculated based on the sunrise/sunset time and the length of day respectively with accuracy estimates of within 100 km [6]. This is sufficient for long-distance migratory birds for determining general flight paths and stopover sites [6]. However, data collected from geolocators cannot be used to accurately determine migratory routes and could never be used to estimate fine-scale movement patterns near proposed or active offshore wind energy facilities. This method is challenging as it can require recovery of the tag.

Finally, various radar systems have been used to track movements of birds include surveillance, doppler and tracking radar [13]. Surveillance radar has capabilities of detecting individual large birds within a few kilometers and a flock up to 10 km, but with a higher power system they can have a range of 100-240 km [13]. Doppler can be useful for tracking small differences in movements and velocity and tracking radar can even analyze wing beat signatures as long as there is only one bird present [13]. With the wing beat and/or amplitude signatures, recordings from radar can be used to attempt to characterize a bird's species but overall radar systems can't tell species apart [14]. Additionally they can be expensive and challenging to set up for offshore environments. These systems have been effective for investigating local movements [13].

Each of these methods are always being improved upon and additional new methods/technology are being developed as well.

2.3.2 Emerging Tracking System Studies

There have been many studies over the years to improve these bird tracking methods. Three tracking systems include tests that include a receiver set up on drones and one that includes new processing techniques.

The Cornell Lab of Ornithology group had designed a new time of arrival approach (TOA) that uses pseudorandom noise (PRN) code for processing. This method entails a continuous network of receivers and records the arrival times of detected transmissions. The receivers then send the information to a centralized server to calculate the transmitter position. The transmitter is based on a microcontroller with a precision reference clock, frequency synthesizer, modulator and amplifier. This tag uses a binary phase-shift keying (BPSK) modulation scheme that sends a 1- to 2-ms-long signal once every minute during the most active times of day for birds.

This method was developed to reduce the issues that come with transmitters with shared carrier frequencies close by. They can be hard to differentiate but the autocorrelation property of the PRN uses cross-correlation to allow these signals to have minimal interference with precise synchronization [6].

However, if the carrier frequency is uncertain with clock errors and doppler shift present, the PRN code shows correlation disagreements. This can be seen in Figure 7.

It was noted that the incoming signal becomes inverted halfway through transmission, which is why there are disagreements in the correlation. However, this error was accounted for using the Equation 9 showing the expected values of the correlator output signal S.

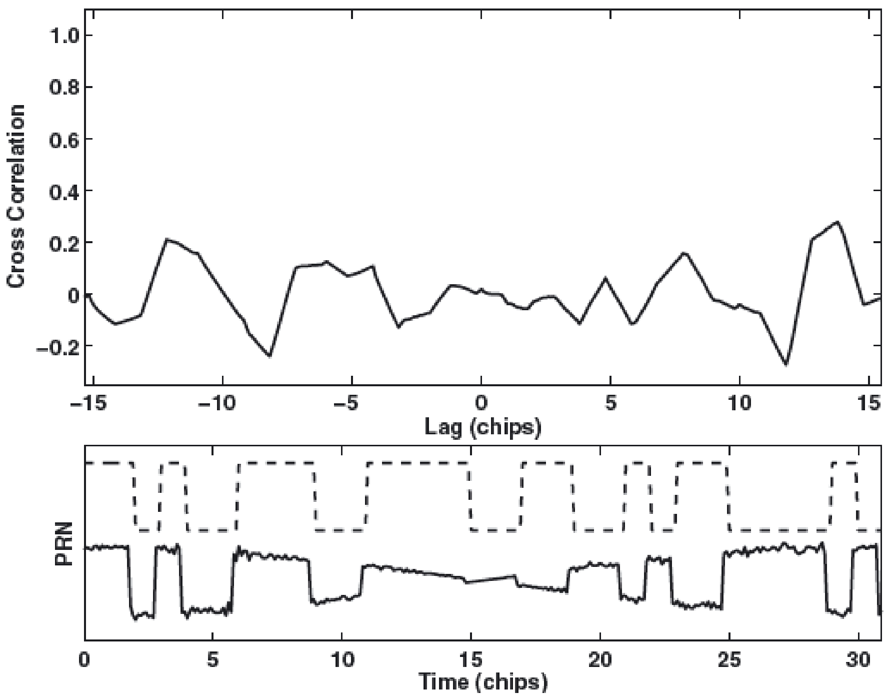


Figure 7: Cross correlation between coded transmission (dotted) and received signal (solid) with a small carrier and local oscillator (LO) mismatch [6].

$$E\{S\} = A \exp^{j(\Delta\Theta + \pi\Delta f T_D)} \bar{R}(\Delta\tau) sinc(\pi\Delta f T_D) \tag{9}$$

Using a value of $\Delta T_D < 2/5$ ensures that at least part of the signal is available to detect, they also decrease the oscillator error by implementing a high-precision one. They also needed to use a sequence that emits a 1-to-2ms-long signal once every minute to be able to ignore the sinc term in Equation 9. The variables in this equations are $\Delta\Theta$ and Δf which are the phase and frequency differences between the tag carrier and LO respectively, T_D is the duration of the PRN sequence, $\Delta\tau$ is the phase difference between received PRN sequence and template and A is the carrier amplitude.

It was found that the better features of the TOA radio tracking system include the low-power tags, automatic detection and good location accuracy but they depend on a continuously listening network of antennas. They learned that a TOA receiver requires very precise frequency or time information such as ADC sampling, the LO generator, and the buffer time stamp. It was noted that a tolerance as low as 0.1 ppm does not offer the stability needed for precise synchronization over time [6]. Overall, this method is similar to Motus by having antennas that are continuously listening and send data to a centralized server however the tag emissions are not bursts but are pseudorandom noise.

VonEhr et al. [2016] employed an unmanned automated vehicle (UAV) mounted Yagi Rotation (YR) Radio Direction Finder (RDF) (Figure 8) to estimate DOA using phased calculation techniques [15]. They also attempted to test a pseudo-Doppler method that included omni-directional monopoles, however they found that the SNR was too low to utilize that approach. Their YR system used the FUNcube Dongle Pro+, Software Defined Radio (SDR) for RF passband to baseband conversion, a Raspberry Pi 2, 3DR Pixhawk which is a flight controller with GPS, and station laptop running 2DR Mission Planner c1.3.28. This method was not completely automated and needed a controller that either piloted the drone or the radio-controlled transmitter.

The goal was develop a more efficient protocol for trackers search for tagged animal. Tracking VHF radio tagged animals normally employs biologists to perform ground searches using line-of-sight (LOS) with an aircraft to increase signal reception. However, fly time is expensive and potentially dangerous when searching at low altitudes. They detected errors from multi-path interference from surrounding buildings, however once those were mitigated, the accuracy of bearing estimates increased to $\pm 20°$. Additionally, they found that the FDP+ had some DC inter-

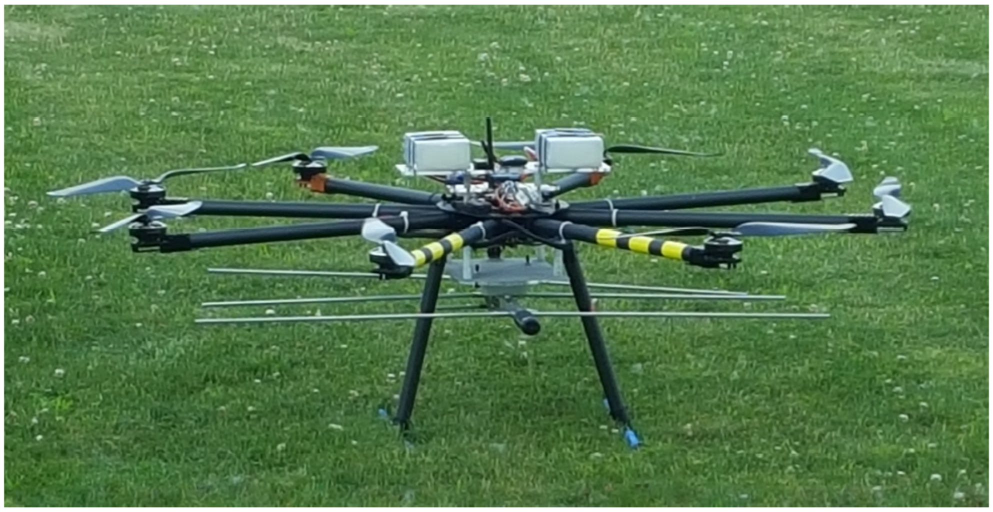


Figure 8: Mounted Yagi Rotatation system on Octocopter [15].

ference that would cause a DC offset of 25 dB with 1 kHz bandwidth, but they reduced this error with filtering.

Another receiving system was installed quadrotor drone with omni-directional dipole antennas [16]. They used the FDP+ but with a BeagleBone computer to record the signals of multiple transmitters simultaneously. Using Lotek transmitter, a coded VHF NanoTag, they performed calibration tests in difficult to access terrain.

One of the two methods used was selecting the drone location where the signal strength was strongest and calculating the Euclidean distance between those XY coordinates. The other method modeled the signal strength as a function of XY coordinates using a quadratic equation. They found that the transmitter signals were dampened by the terrain. and multi-path effects made estimating bearings practically impossible.

Overall the method of using the strongest signal strength positions and finding the Euclidean distance sustained a mean error of 134 m with a range of 44–278 m.

The second method, using the quadratic and a least-squared regression showed a mean error of 94 m for range values of 15-285 m. This is not precise enough to find nests, it still is a more cost effective way to enhance the quality and quantity of data. It also shows that using signal strength is not enough to track wildlife accurately and doesn't work well for tracking moving birds offshore as drone can't be flown all the time.

2.3.3 Bird-Wind Turbine Interactions

There are 13,000 offshore wind turbines operating in European offshore waters [13], with offshore wind energy facilities continuing to expand globally and to mitigate climate change [17, 18, 19]. However, wind energy facilities are not without environment conflicts, which includes potential negative consequences to birds [20, 21, 22].

Offshore Tracking with VHF Arrays

The Bureau of Ocean Energy Management (BOEM) oversees the renewable energy developments on the Outer Continental Shelf (OCS) of the United States [7]. These renewable energy systems expose birds to collisions risks and assessments require species-specific information about movement patterns, flight altitude and behavioral avoidance of turbines [22]. Assessments for exposure levels to birds include macro-scale (geographical area), meso-scale (within rotor-swept altitudes) and micro-scale (rotos-swept area) [20].

According to Marques et al. [2014], after an extensive literature review, they found 217 papers with experimental designs that investigated bird communities and geographical areas being affected by the wind turbines, 60% of which were from 2008 of later. Of these papers 60% of them were from Europe (mainly Spain and the UK) and 33% from the USA that identify contributing factors to collisions such as species characteristics, site details and wind farm features [21]. The common broad categories of factors affecting birds are 1) the avoidance response, 2) physical habitat loss/modification and 3) collision mortality either from either the infrastructure or pressure vortices [22, 7].

Loring et al. [2019] performed a study to track offshore occurrences of Common Terns, endangered Roseate Terns and threatened Piping Plovers with VHF Arrays. VHF transmitters are currently the sole option available to track small birds. GPS loggers weight has decreased (1 to 3 g) require hundreds of high accuracy (<10 m) 3-dimentional tracking data but these units are archival and must be recovered [7]. With recent developments of the VHF technology these coded transmitters can be simultaneously tracked which allowed for multiple species tracking [7]. Triangulation was used to localize the species of interest. This method uses both signal strength and bearings from signals received on the directional antennas from multiple towers. There were 22 automated radio telemetry stations in addition to the Motus network data was added for spatially-explicit, empirical assessment of exposure risk. The Motus data provided the following information from the coded transmitter: bird ID, location, time, date, receiving antenna, signal strength value.

Pamela et al. 2019 has also developed bird migration movement models that apply radio propagation theory based on the measured signal strengths to improve the Motus Wildlife Tracking Network analysis.

Understanding Avian-Wind Turbine Interactions in the Offshore Environment

Population-level impacts for marine birds are a concern when constructing offshore wind turbines. Some physical factors to take into consideration when determining which species are more likely to influence collision mortality risk are flight maneuverability, percentage of time flying (peaks of flight activity in migrant species), habitat specialization, nocturnal light activity, disturbance by the wind farm structures and flight altitude being the most important [23, 24].

Hüppop et al. [2006] combined multiple techniques to study bird migration near offshore wind turbine sites. These techniques included radar, thermal imaging, with visual and acoustic observations. They found that half of the birds fly at rotor swept altitudes, the reverse migration numbers can cause increased risk of collision and some terrestrial birds species get struck in large numbers under poor visibility [24].

An example of thermal imaging is the Thermal Animal Detection System that has a 360° of azimuth coverage from using surveillance radars and obtains altitude from the vertical scanning with a 'T-bar'. Infrared camera systems apply fast viewing or recordings, exclude the non-bird observations through trigger software, and can superimpose several hours of recordings onto one image for easier processing. Biologists are currently using TADS even though it can be challenging to operate offshore [13].

Different countries have already deployed systems to check for flight changes, habitat deconstruction, and collisions using different forms of radar. TADS doesn't measure the collisions directly it offers input data to collision models by obtaining the avoidance behavior close to the turbine rotor-blades, flock size and flight altitude. It also is required to validate these models/numbers generated of birds being killed by either the turbine structure or the vortices encountered near the turbine wakes.

Wildlife impact mitigation techniques suggested are avoiding constructing them in dense migration zones, align the turbines in-line with migratory patterns, space out wind farms several kilometers, avoid construction between resting and foraging grounds, shutting down turbines for nights with poor visibility weather during the migration periods, illuminations should be small and intermittent, and general turbine design should be more recognizable to birds [24].

List of References

[1] H. H. Jenkins, *Small-Aperture Radio Direction-Finding.* Massachusetts, United States of America: Artech House Inc., 1991.

[2] S.-M. David, F. J. Meca-Meca, E. Martín-Gorostiza, and J. L. Lázaro-Galilea, “Snr degradation in undersampled phase measurement systems,” *Sensor*, vol. 1, no. 10, p. 1772, 2016.

[3] P. López-López, “Individual-based tracking systems in ornithology: welcome to the era of big data,” *Ardeola*, vol. 63, no. 1, pp. 103–136, 2016.

[4] J. P. Severson, P. S. Coates, B. G. Prochazka, M. A. Ricca, M. L. Casazza, and D. J. Delehanty, “Global positioning system tracking devices can decrease greater sage-grouse survival,” *The Condor*, vol. 121, no. 3, p. duz032, 2019.

[5] E. S. Bridge, K. Thorup, M. S. Bowlin, P. B. Chilson, R. H. Diehl, R. W. Fléron, P. Hartl, R. Kays, J. F. Kelly, W. D. Robinson, *et al.*, “Technology on the move: recent and forthcoming innovations for tracking migratory birds,” *BioScience*, vol. 61, no. 9, pp. 689–698, 2011.

[6] R. B. MacCurdy, R. M. Gabrielson, and K. A. Cortopassi, “Automated wildlife radio tracking,” *SA Zekavat & RM Buehler (red). Handbook of position location: theory, practice, and advances. Wiley, Hoboken, NJ*, 2011.

[7] P. Loring, P. Paton, J. McLaren, H. Bai, R. Janaswamy, H. Goyert, and P. Sievert, “Tracking offshore occurrence of common terns, endangered roseate terns, and threatened piping plovers with vhf arrays,” *Sterling (VA): US Department of the Interior, Bureau of Ocean Energy Management. OCS Study BOEM*, vol. 17, p. 140, 2019.

[8] *NanoTags (Coded VHF)*, Lotek, 2020, rev. 1.2. [Online]. Available: https://www.lotek.com/wp-content/uploads/2017/10/NanoTags-Spec-Sheet.pdf

[9] P. Taylor, T. Crewe, S. Mackenzie, D. Lepage, Y. Aubry, Z. Crysler, G. Finney, C. Francis, C. Guglielmo, D. Hamilton, *et al.*, "The motus wildlife tracking system: a collaborative research network to enhance the understanding of wildlife movement," *Avian Conservation and Ecology*, vol. 12, no. 1, 2017.

[10] M. T. Hallworth and P. P. Marra, "Miniaturized gps tags identify non-breeding territories of a small breeding migratory songbird," *Scientific reports*, vol. 5, p. 11069, 2015.

[11] P. Loring, "Evaluating digital vhf technology to monitor shorebird and seabird use of offshore wind energy areas in the western north atlantic," Ph.D. dissertation, UMass Amherst, 2020.

[12] P. Taylor, T. Crewe, S. Mackenzie, D. Lepage, Y. Aubry, Z. Crysler, G. Finney, C. Francis, C. Guglielmo, D. Hamilton, *et al.*, "The motus wildlife tracking system: a collaborative research network to enhance the understanding of wildlife movement," *Avian Conservation and Ecology*, vol. 12, no. 1, 2017.

[13] M. Desholm, A. Fox, P. Beasley, and J. Kahlert, "Remote techniques for counting and estimating the number of bird–wind turbine collisions at sea: a review," *Ibis*, vol. 148, pp. 76–89, 2006.

[14] R. C. Beason, T. J. Nohara, and P. Weber, "Beware the boojum: caveats and strengths of avian radar," *Human–Wildlife Interactions*, vol. 7, no. 1, p. 4, 2013.

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

[16] A. Desrochers, J. A. Tremblay, Y. Aubry, D. Chabot, P. Pace, and D. M. Bird, "Estimating wildlife tag location errors from a vhf receiver mounted on a drone," *Drones*, vol. 2, no. 4, p. 44, 2018.

[17] R. J. Barthelmie, "A brief review of offshore wind energy activity in the 1990's," *Wind Engineering*, pp. 265–273, 1998.

[18] V. Hamadi, U. Brosnan, I. Loftus, and G. Montgomery, "Offshore substation design: High-level overview of the industry best practices," *IEEE Power and Energy Magazine*, vol. 17, no. 4, pp. 67–74, 2019.

[19] E. R. M. Shouman, "Global prediction of wind energy market strategy for electricity generation," in *Modeling, Simulation and Optimization of Wind Farms and Hybrid Systems*. IntechOpen, 2020.

[20] J. Burger, C. Gordon, J. Lawrence, J. Newman, G. Forcey, and L. Vlietstra, "Risk evaluation for federally listed (roseate tern, piping plover) or candidate (red knot) bird species in offshore waters: A first step for managing the potential impacts of wind facility development on the atlantic outer continental shelf," *Renewable Energy*, vol. 36, no. 1, pp. 338–351, 2011.

[21] A. T. Marques, H. Batalha, S. Rodrigues, H. Costa, M. J. R. Pereira, C. Fonseca, M. Mascarenhas, and J. Bernardino, "Understanding bird collisions at wind farms: An updated review on the causes and possible mitigation strategies," *Biological Conservation*, vol. 179, pp. 40–52, 2014.

[22] A. Fox, M. Desholm, J. Kahlert, T. K. Christensen, and I. Krag Petersen, "Information needs to support environmental impact assessment of the effects of european marine offshore wind farms on birds," *Ibis*, vol. 148, pp. 129–144, 2006.

[23] R. W. Furness, H. M. Wade, and E. A. Masden, "Assessing vulnerability of marine bird populations to offshore wind farms," *Journal of environmental management*, vol. 119, pp. 56–66, 2013.

[24] O. Hüppop, J. Dierschke, K.-M. EXO, E. Fredrich, and R. Hill, "Bird migration studies and potential collision risk with offshore wind turbines," *Ibis*, vol. 148, pp. 90–109, 2006.

CHAPTER 3

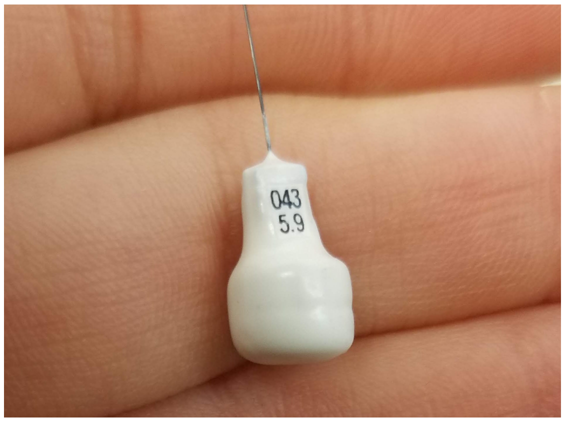
Methods, Set-up and Limitations

The overall design can be separated into three parts: the transmitter, the receiver design and the data collection which are outlined in the sections below.

3.1 Transmitter

Wildlife tracking using telemetry has a bandwidth of approximately 1 kHz in the VHF range and has a power less than 10 mW [1]. A typical bird tracking transmitter that uses telemetry is the VHF transmitter (nanotag, Lotek.com). The tag includes a local oscillator set at a specific carrier frequency that can switch on and off by a timing circuit. A 20 ms pulse is transmitted when the circuit is turned on based on the programmed period. When deciding the duration and period of the timing circuit there are a few things to consider. The longer pulses mean more energy consumption which cause shorter battery life. If there were longer delays between bursts instead the tag would become more difficult to track. The optimal duty cycle parameters, according to Maccurdy et al. [2011], are 20 ms pulses with periods of less than 60 seconds. The tag usually costs less than $250, which is generally low compared to other tracking tags [2].

The transmitter used in this work was the Lotek digitally-coded VHF NanoTag with a carrier frequency of 166.380 MHz and 7 s intervals (hereafter test tag). This test tag had a lifespan of approximately 280 days, weighed 0.9 g, and a transmission power of -30 dBm (Figure 9). Note that other tags, such as the plover and tern tags, typically emit a signal every 5-6 s [3].



(a) NanoTag used for testing.

(b) Tagged Piping Clover. Photographed by Peter Paton.

Figure 9: Digitally-coded VHF transmitter (Lotek NanoTag) used for this study.

3.2 Receiver Design

The list of materials needed to construct the 3-element antenna array, including the receiver, and associated equipment are shown in Appendix A (for design and pricing see Antenna Structure Build and Antenna Components and Instruments).

3.2.1 Antennas

In order to design and build a system that can accurately estimate DOA within 1 km, selecting the correct type of antenna is critical. For this research, three Shakespeare 476 Classic VHF (Figure 10) antennas were utilized. These antennas were 6.4 m tall, omnidirectional in azimuth, and have a gain of 10 dB. These antennas are collinear arrays, specifically Franklin arrays, that consist of a meander-line phase reversal [5].

In order to calculate DOA using phase, the distance between the antennas must be less than or equal to half the wavelength of the incoming electronic pulse (for a 166.38 MHz – 0.9015 m) to limit the ambiguity of the signal bearing. Thus, the three antennas are equally spaced at 0.5 wavelength, or 0.889 m, for this

Figure 10: Shakespeare 476 Classic VHF Antenna [4].

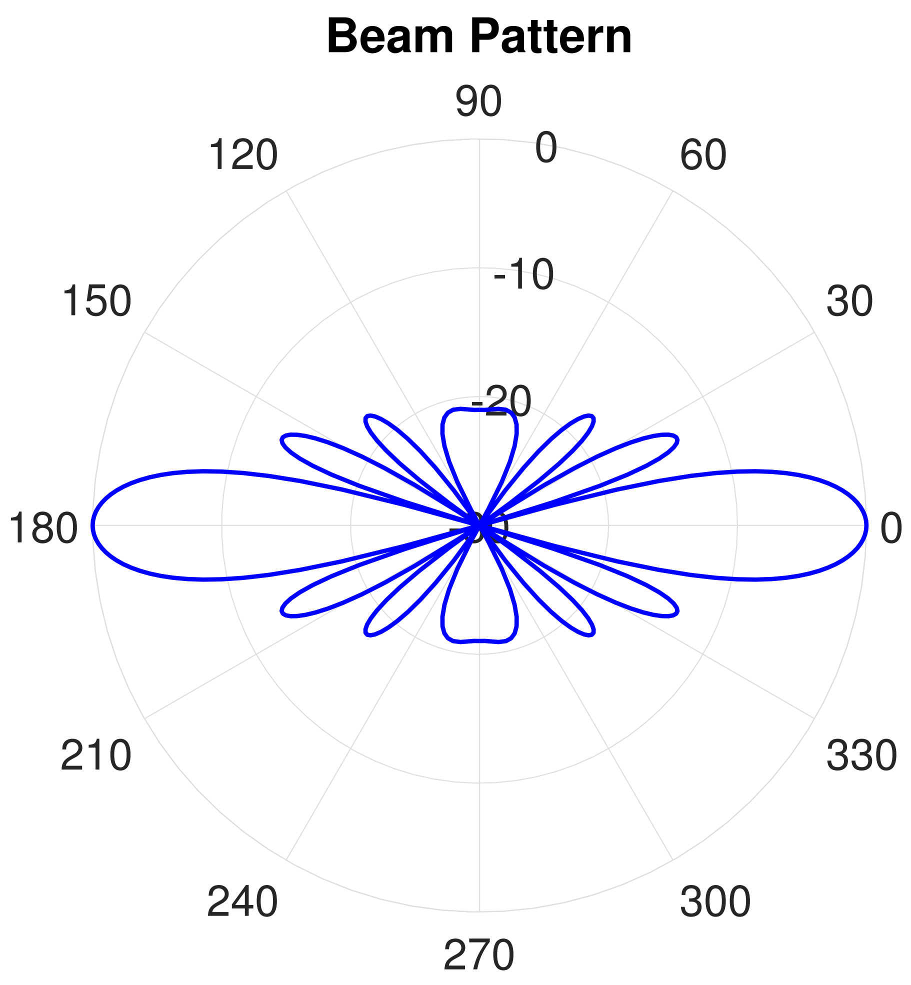


Figure 11: Ideal beam pattern for a collinear array of phased 1/2-wave elements.

study.

The ideal beam pattern for the antenna is shown in Figure 11. The main beam of the antenna is directed horizontally and is omnidirectional in azimuth. Also present are sidelobes -13.3 dB at 24.2°, -17.9 dB at 44.3°, and -21.0 dB at 90.0° above and below the horizontal. These sidelobes are also omnidirectional in azimuth.

3.2.2 Antenna Structure

The antenna structure had the following RDF and environmental constraints to consider. For direction finding, the main objectives were spacing and keeping the antennas rigid and vertically straight. The environmental constraints were based on future testing and installment of this system which included being stable and durable enough to handle harsh weather, uneven land, wind and being elevated enough for optimizing the beam pattern and/or the possibility of placing the system in uneven terrain.

The design and machining of this base was done by Fred Pease at URI (Figures 12 and 13) highlighting the main components that make it both dependable and adjustable.

This included the welded base center set into the group, 2x6 pieces of wood for extra surface tension (Figure 13b), galvanized metal clamps securing the antennas 0.889 m equidistantly from the metal poles (Figure 13a), and the PVC spacers (Figure 13c). The spacers were set 3.4 and 4.8 meters up the antenna using custom milled PVC 4" diameter discs attached with 1" diameter PVC rods.

The antenna height aids in being able to place the antenna systems in uneven terrain environments while still maintaining a relatively useful beam pattern. The final step for the installing of this structure includes three guy wires to ensure stability of the system as well.

3.3 Data Collection

The collection of received signals includes both hardware and software. By incorporating radio software, modifications can be made to the de/modulation scheme and bandwidth spread, which can be useful for tuning the RF mixer to capture different digitally-coded VHF transmitter frequencies. For this thesis, the open source software Software Defined Radio (SDR) was utilized.

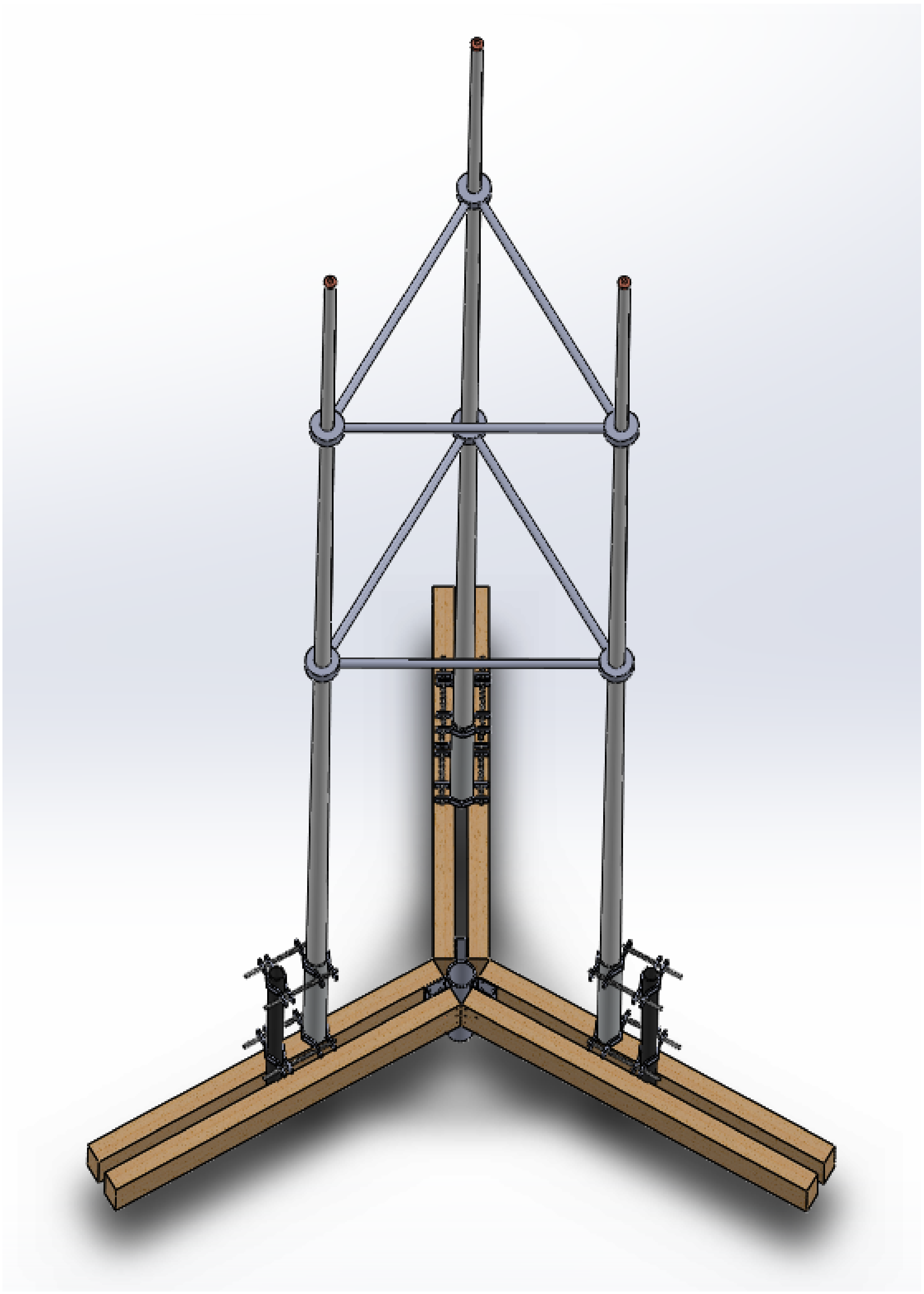
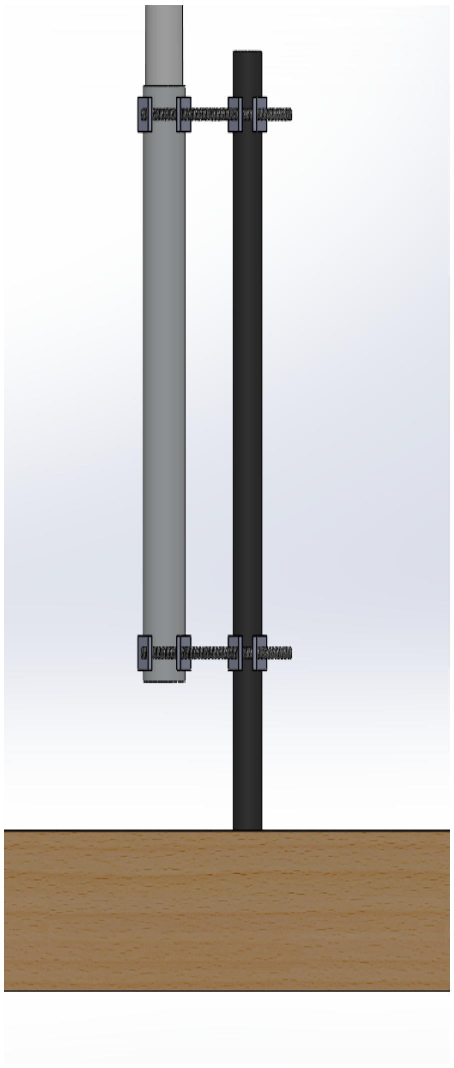
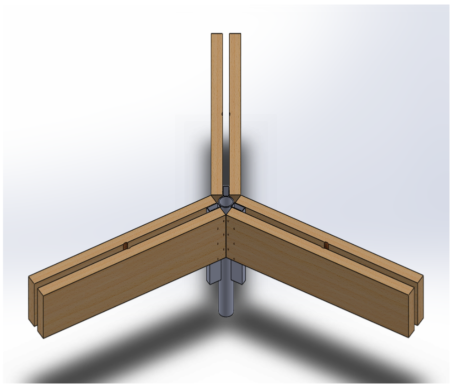


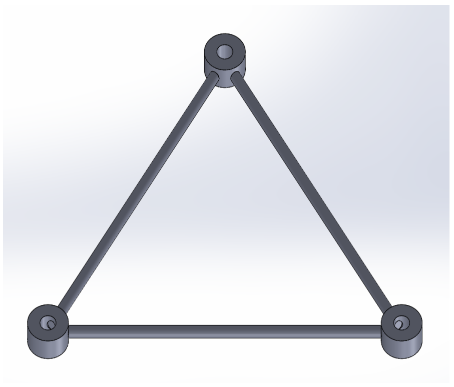
Figure 12: Isometric view of the Solidworks model of the 3-antenna system.



(a) Galvanized antenna clamps with adjustable threaded rod.



(b) Base of the antenna structure, including a welded metal piece with wooden 2x6 screwed in.



(c) PVC rods and milled PVC glued together.

Figure 13: Components of the antenna system.

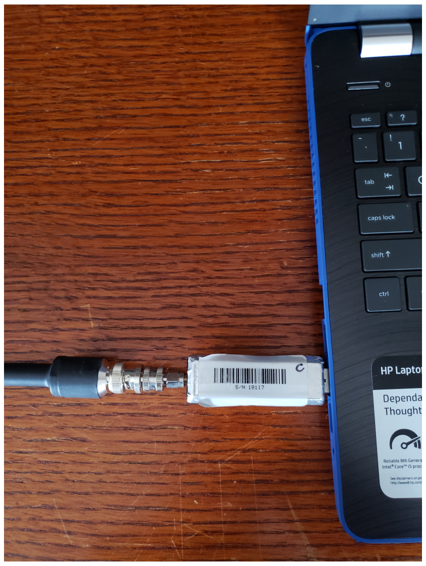


Figure 14: Coax cable connected to FUNCube Dongle to convert the analog signals to digital for the SDR software.

3.4 Hardware

Appendix A shows the list of materials used under Antenna Components and Instruments. The electronic arrangement consisted of a coax cable (TWS400-15 - 15' Jumper with TWS400 3/8", 50 ohm braided cable with BNC Male and UHF Male Installed) connecting the antenna to a USB FUNCube Dongle (FCD) that is plugged into a laptop running SDR seen in Figure 14. The software set up is outlined in Appendix C.

The antennas receive the electromagnetic plane waves and send the voltage to the FCD (process outlined in Figure 15). The received signal, $r(t)$, is passed through a low-noise amplifier (LNA) and a radio frequency (RF) tuner connected to a local oscillator to grant the user tuning control for the center frequency. A RF mixer has three ports, two of which are inputs and one output. These ports are referred to as the RF input port, local oscillator (LO) input port and the RF output port. The output is the two signals $r(t) \pm LO$, where the signal from LO is usually a sinusoidal continuous wave or square wave. The purpose of the RF mixer is to change the

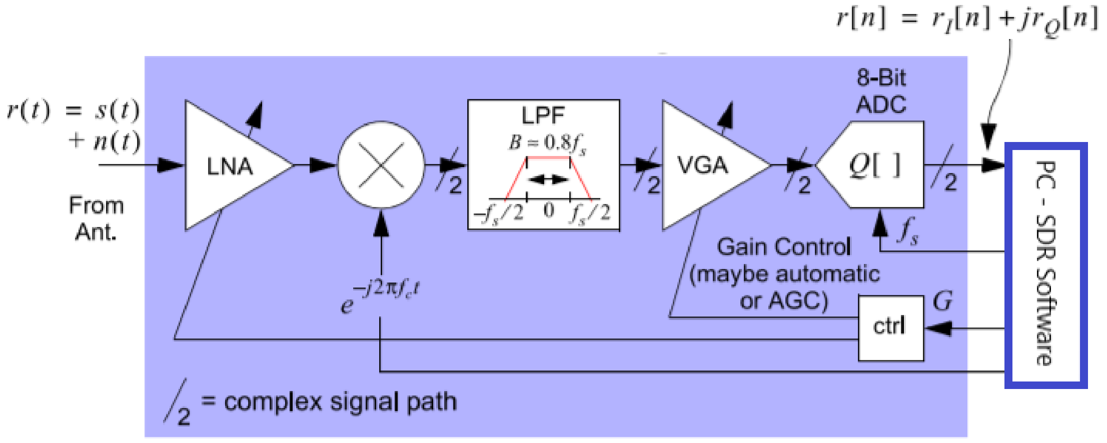


Figure 15: Behavior level model of RTL-SDR, comparable to the FUNCube Dongle. Adapted from [6].

frequency of the electromagnetic signal while maintaining original characteristics such as phase and amplitudes. By doing this the signal can be amplified at a baseband frequency [7].

$$A(t)\cdot\sin[2\pi ft+\phi(t)] = A(t)\cdot\sin(2\pi ft)\cdot\cos[\phi(t)] + A(t)\cdot\sin(2\pi ft+\frac{\pi}{2})\cdot\sin[(\phi(t))] \tag{10}$$

This process is known as demodulation (Equation 10), where $A(t)$ is amplitude, $\phi(t)$ is phase, partly due to propagation. The demodulated signal is then filtered by the low-pass filter (LPF) and then to the variable gain amplifier (VGA) to help increase SNR before being output through the analog-to-digital converter (ADC). The ctrl box on Figure 15 shows that the through SDR software, the LNA and VGA can be adjusted.

The accuracy of the output signal, $r[n]$ in Figure 15, is dependent on the ADC bit depth, ADC sample rate, local oscillator quality and RF mixer stability. In this case, the documentation showed that the FCD has a stability of $\pm 1.5ppm$ [8] . When applied to the test tag frequency, the frequency has a variation of $\pm$ 249.57 Hz [8]. Also, the ADC bit depth affects the noise floor. To reduce noise

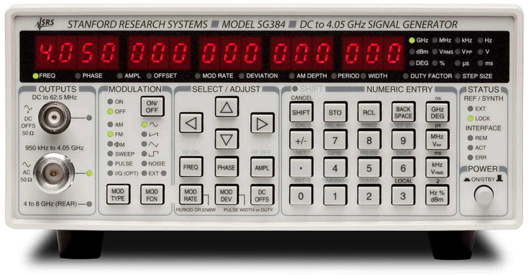


Figure 16: Source generator with rubidium timebase [10].

and increase dynamic range a larger bit depth is necessary [9].

To minimize some of these errors and synchronize the three antennas, a signal generator (Stanford Research Systems SG380, Sunnyvale, California) with a rubidium timebase provided a stability of $<\pm 0.0001 ppm$ (166.38 MHz ± 0.016 Hz) [10]. The signal generator was set to propagate a continuous 166.385 MHz sine wave (5,000 Hz above the NanoTag carrier signal) through a 5.8 GHz rubber duck omnidirectional antenna. The calibration source antenna was set equidistant from the three antennas.

Once the array was constructed, the software was set to begin recording data at specific time intervals for a set duration on the three computers. The downconverted signal was then saved into a .wav file for post-processing. For this thesis, a sample rate of 44.1 kHz was chosen with a center frequency of 166.385 MHz. The two channels of the .wav file contained the inphase and quadrature components of the tag signal complex envelope [11].

List of References

[1] A. L. Kolz, “Radio frequency assignments for wildlife telemetry: A review of the regulations,” *Wildlife Society Bulletin (1973-2006)*, vol. 11, no. 1, pp. 56–59, 1983.

[2] R. B. MacCurdy, R. M. Gabrielson, and K. A. Cortopassi, “Automated wildlife radio tracking,” *SA Zekavat & RM Buehler (red). Handbook of po-*

sition location: theory, practice, and advances. Wiley, Hoboken, NJ, 2011.

[3] P. Loring, “Evaluating digital vhf technology to monitor shorebird and seabird use of offshore wind energy areas in the western north atlantic,” Ph.D. dissertation, UMass Amherst, 2020.

[4] 2020. [Online]. Available: https://www.marine.com/products/11-10612/shakespeare-476-21-vhf-antenna

[5] Q. Xue and S. Liao, “High frequency and high gain two-element collinear antenna array,” in *2015 International Symposium on Antennas and Propagation (ISAP)*. IEEE, 2015, pp. 1–3.

[6] M. A. Wickert, “Lab 6: Software defined radio and the rtl-sdr usb dongle,” Lab Practical ECE4670, 2000. [Online]. Available: http://www.eas.uccs.edu/~mwickert/ece4670/lecture_notes/Lab6.pdf

[7] C. Nickolas. Digi-Key Electronics. “The basics of mixers.” [Online; accessed Apr 4, 2020]. 2011. [Online]. Available: https://www.digikey.com/en/articles/the-basics-of-mixers

[8] “Specifications.” [Online]. Available: http://www.funcubedongle.com/?page_id=1201

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

[10] S. R. Systems, “Rf signal generators: Sg380 series,” 2018, [Online; accessed Nov 8, 2018]. [Online]. Available: https://www.thinksrs.com/products/sg380.html

[11] M. D. K. Proakis, John G, *Digital Signal Processsing (4th Edition)*. Prentice-Hall, Inc., USA, 2006.

CHAPTER 4

Experiments

During the summers of 2018 and 2019 both stationary testing and drone testing occurred. During the summer of 2018, the system originally included a RF Explorer (Digi-Key Electronics 1597-1173-ND, Thief River Falls, MN) as the calibration source. The generator has a frequency stability of ± 0.5 ppm and a frequency accuracy of ± 1 ppm [1]. After preliminary analysis of the data, it was determined the accuracy was not suitable for phase processing for this research and Stanford Research Systems SG380 signal generator (Stanford Research Systems SG380, Sunnyvale, CA) with greater precision was acquired for the 2019 field season to increase the accuracy of bearing estimates.

4.1 Fixed Tests

In the summer of 2018, a series of experiments were performed to determine the differences between each of the recording systems. FUNcube dongles (FCD) were labeled as A, B and C to estimate the variation in their clocks. In this test, the original calibration source (RF Explorer) was turned on after a few seconds. A snippet of the recorded data (Figure 17) shows the magnitude after filtering and the corresponding unwrapped phase values. This visually shows the time differences of the computer clocks and respective amplitude differences. The time of arrival differences from the antenna 1 to antenna 2 and 3 recording systems are 36 and 6 ms respectively, and have amplitude differences of 0.011, and 0.023. The magnitude inconsistencies could be attributed to the filtering process within the FCD or antenna cable connections having different percentages of physical contact. In other words, the SO-239 type connector on the bottom of the antenna and the

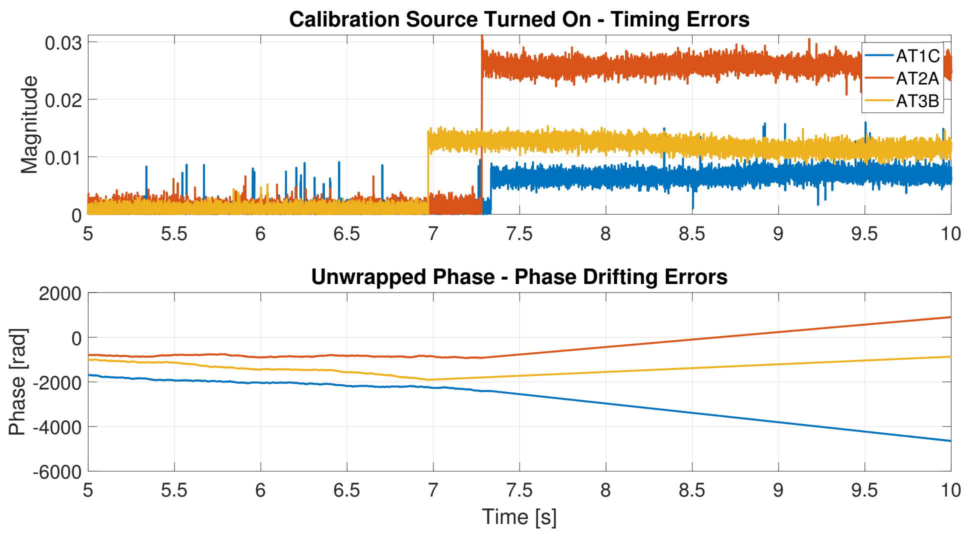
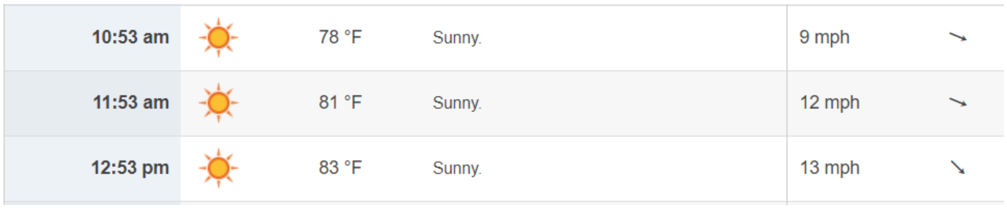


Figure 17: Preliminary tests for determining timing errors (upper panel) and phase drifting errors (lower panel) at three omnidirectional antennas (blue = Antenna 1, red = Antenna 2, yellow = Antenna 3).

UHF connector from the coax cable are not perfectly snug. On the bottom subplot the unwrapped phase is shown with and without the calibration source on. At first there were small variations in the phase, but when source was turned on the phase began to drift. By having the calibration source equidistant from each antenna, phase values should be equal, or close to. In the lower panel of Figure 17, the phase values differentials are varying. In other words, the slopes for each antenna should be close in value, but are not. The unwrapped phase values from antenna 1 show a negative differential fringe, this is from the wrapped phase values jumping from $-\pi$ to π unlike the antennas 2 and 3 that have phase jumps from π to $-\pi$. The differentials for antennas 1 through 3 are -838, 670, 343 rad/s respectively.

4.2 Drone Tests

Drone flights with the test tag attached were scheduled with pilot Greg Bonynge from the Department of Natural Resources Science at the University of Rhode Island to estimate the accuracy bearing calculations from phase measurements by



10:53 am	☀	78 °F	Sunny.	9 mph	↘
11:53 am	☀	81 °F	Sunny.	12 mph	↘
12:53 pm	☀	83 °F	Sunny.	13 mph	↘

Figure 18: Weather on the day of testing for Narragansett, RI [2].

knowing the drone's location.

The tests occurred on August 9th, 2019 and the weather at the time of testing can be seen in Figure 18. The weather is important because technology has temperature ratings for both operation and measurement stability. For example, the NanoTag from Lotek operates within 32 - 95$°F$ [3] and although temperature stability was not rated for the calibration source, the specifications indicate the rubidium timebase improves temperature stability [4]. Additionally, the wind speed and direction affect the antennas spacing and the noise floor by shaking the system.

The drone was programmed to record the GPS coordinates (accuracy $\pm$ 1m) after hovering for 15 seconds to estimate the test tag location. The hovering points were set before the drone changed direction to aid in both GPS stability and to record two fixes at the same bearing.

There were six flights total. For Flight 1 the drone flew a diamond pattern around the antenna system to test a full 360° bearing estimations to check for consistency twice. The first time around the the drone was set to fly at 60 m altitude and 2 m/s and the second time for 120 m altitude and 13 m/s. Flight 2 is described below, Flight 3 and 4 include a west to east transect at 2 m/s at altitudes 60 and 120 m respectively. Finally Flights 5 and 6 were a triangular shape pattern over the bay at 3 m/s to check the relation between distance and bearing estimation

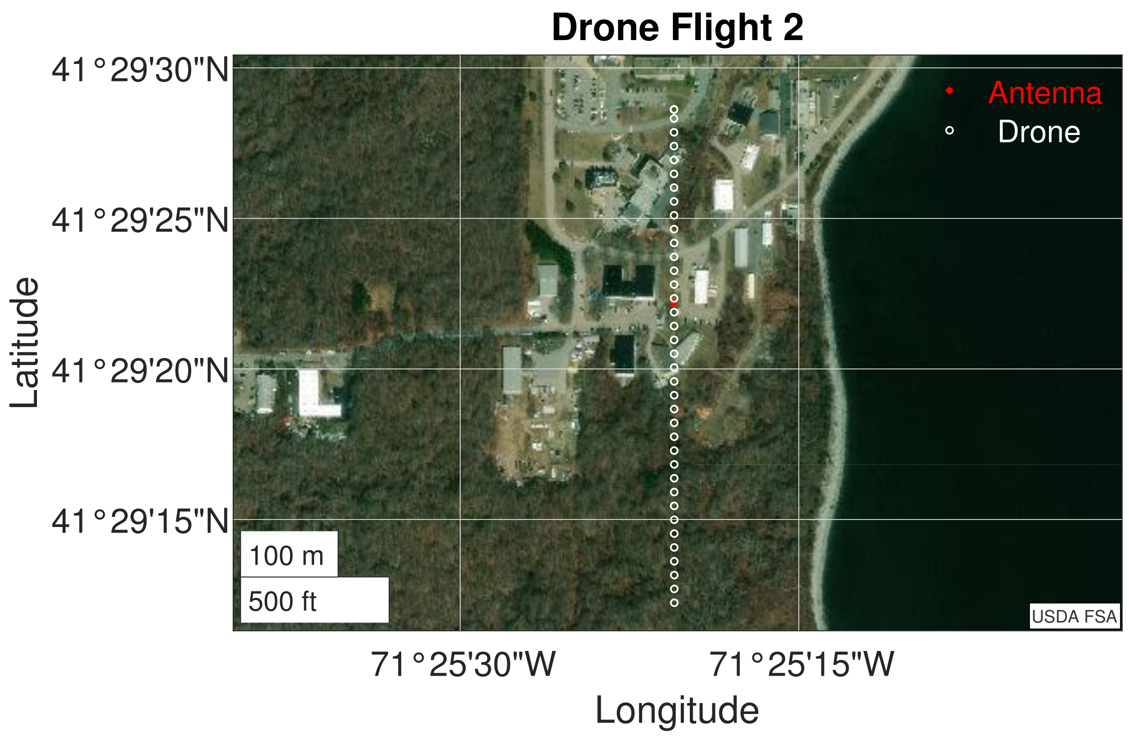


Figure 19: Drone flight 2, a north to south transect over the antenna system.

accuracy, also at 60 and 120 m altitude.

Originally there was an altitude test where the drone stayed at a fixed location and only increased in the z-direction. However, there were cable connection during that test making the data unusable. However, the two altitudes, 60 and 120 m, and the transects directly over the antenna system were enough to see some beam pattern effects. The speed variations were based on file recording length and were slow enough to obtain a large amount of fixes on the test tag. At a distance of 650 m the SNR was large enough to see detections. Flight plans and notes taken can be found in Appendix C. However, due to the limitations and lack of coherency of the equipment, the data processing had shifted the focus to specifically Flight 2, the transect from North to South over the antennas. Flight 2 began at 11:32 AM, the wind was north westerly and somewhere between 4 and 5.3 m/s (Figure 18).

Figure 19 shows the transect with perspective to the antennas. The white circles are the locations of the drone when the burst transmission occurred. The red dot represents the antenna system receiver.

List of References

[1] *RF Explorer Signal Generator*, Digi-Key Electronics, 2019, updated to Firmware Version 1.31. [Online]. Available: http://j3.rf-explorer.com/40-rfe/article/132-rf-explorer-signal-generator-rfe6gen-specification

[2] Time and Date, “Past weather in narragansett, rhode island, usa — august 2019,” cited 2020, [Available online at ”https://www.timeanddate.com/weather/usa/narragansett/historic?month=8&year=2019”].

[3] *NanoTags (Coded VHF)*, Lotek, 2020, rev. 1.2. [Online]. Available: https://www.lotek.com/wp-content/uploads/2017/10/NanoTags-Spec-Sheet.pdf

[4] S. R. Systems, “Rf signal generators: Sg380 series,” 2018, [Online; accessed Nov 8, 2018]. [Online]. Available: https://www.thinksrs.com/products/sg380.html

CHAPTER 5

Processing Methods

5.1 Theory

This section includes the derivations used in the research to obtain the direction of arrival estimations.

An CW signal from a source, such as a digitally coded VHF bird tag, is $y(t) = A\sin(2\pi f_c t + \theta)$ transmitted from a location $\mathbf{r_b}$ assuming plane wave propagation in the x-y plane.

$$y(t, \mathbf{r}) = A \sin\left[2\pi f_c \left(t - \frac{\hat{n}\cdot(\mathbf{r} - \mathbf{r}_b)}{c}\right) + \theta\right] \tag{11}$$

Equation 11 shows the RF signal radiating from the source located at $\mathbf{r}_b$ with $\mathbf{k} = k\hat{n} = k(u\hat{x} + v\hat{y})$, where k is the wavenumber and is $\frac{2\pi f_c}{c}$, f_c is the carrier frequency, and c is the speed of light.

$$u = \cos\psi \tag{12a}$$

$$v = \sin\psi \tag{12b}$$

The signal is propagating towards the antennas located near the origin on the x-axis and where u and v are the direction cosines given by Equations 12a and 12b.

$$y_1(t) = A \sin\left[2\pi f_c \left(t - \frac{\hat{n}\cdot(0\hat{x} - \mathbf{r}_b)}{c}\right) + \theta\right] \tag{13a}$$

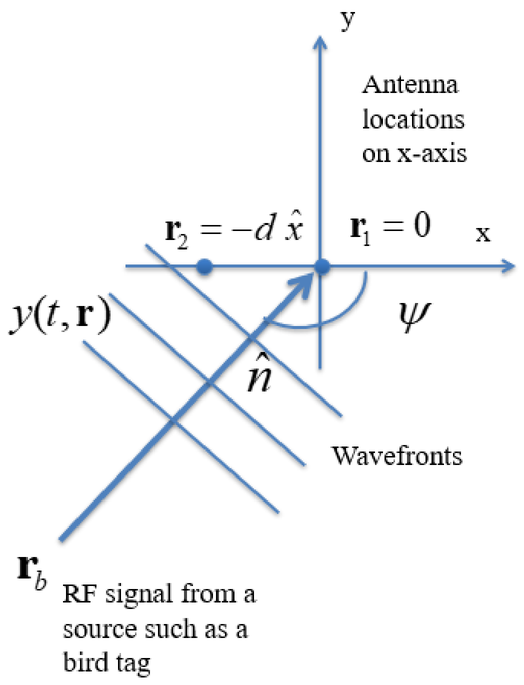

y
Antenna locations on x-axis
$\mathbf{r}_2 = -d\,\hat{x}$
$\mathbf{r}_1 = 0$
x
$y(t,\mathbf{r})$
ψ
$\hat{n}$
Wavefronts
$\mathbf{r}_b$
RF signal from a source such as a bird tag

Figure 20: Example of frequency differences between the shifted baseband signal and calibration source and their respective bandwidth.

$$y_2(t) = A\sin\left[2\pi f_c\left(t - \frac{\hat{n}\cdot(-d\hat{x} - \mathbf{r}_b)}{c}\right) + \theta\right] \tag{13b}$$

The RF signal is received by two antennas located at $\mathbf{r}_1 = 0\hat{x}$ and $\mathbf{r}_2 = -d\hat{x}$, which produces the two signals seen in Equations 13a and 13b.

$$y_1(t) = A\sin\left[2\pi f_c\left(t - \frac{(0 - R)}{c}\right) + \theta\right] \tag{14a}$$

$$y_2(t) = A\sin\left[2\pi f_c\left(t - \frac{(-d\cos\psi - R)}{c}\right) + \theta\right] \tag{14b}$$

These two signals can be simplified using the fact that $\hat{n}\cdot\mathbf{r}_b = R$, this distance from the source such as a bird tag to Antenna 1 and $\hat{n}\cdot d\hat{x} = d\cos\psi$ and are shown in Equations 14a and 14b.

$$y_1(t) = A\sin\left[2\pi f_c\left(t - \tau_0\right) + \theta\right] \tag{15a}$$

$$y_2(t) = A\sin\left[2\pi f_c\left(t + \frac{d}{c}\cos\psi - \tau_0\right) + \theta\right] \tag{15b}$$

Further simplifying the expressions for the two signals are given in Equations 15a and 15b where $\tau_0 = \frac{R}{c}$, the travel time from the source location to the Antenna 1.

$$y_1(t) = A\sin\left[2\pi f_c t + \theta\right] \tag{16a}$$

$$y_2(t) = A\sin\left[2\pi f_c\left(t + \frac{d}{c}\cos\psi\right) + \theta\right] \tag{16b}$$

Note that $2\pi f_c \tau_0$ is a constant phase and can be incorporated into θ, seen in Equations 16a and 16b.

$$y_1(t) = A\sin\left[2\pi f_c t + \theta\right] \tag{17a}$$

$$y_2(t) = A\sin\left[2\pi f_c t + kd\cos\psi + \theta\right] \tag{17b}$$

These expressions can then be simplified into the final forms seen in Equations 17a and 17b, where $k = \frac{2\pi f_c}{c}$, with $f_c = 166.380$ MHz. This produces a wavenumber of $k = 3.48 rad/m$ and a wavelength of $\lambda = \frac{c}{f} = 1.8m$.

Each signal is then added a small calibration signal, $g(t) = G\sin 2\pi f_g t$, with a frequency near the carrier frequency ($f_g = f_c + \Delta f$) before demodulation so that $z_1(t) = y_1(t) = g(t)$ and $z_2(t) = y_2(t) = g(t)$ (Figure 21).

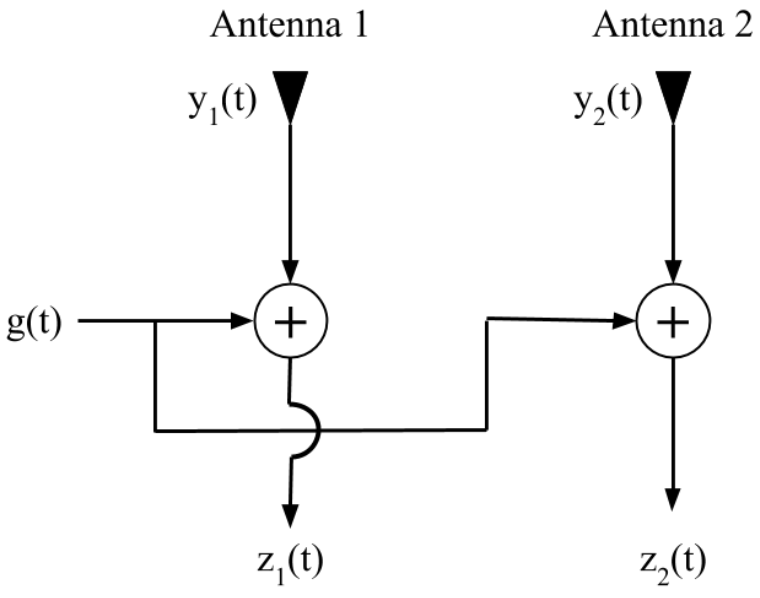


Figure 21: Addition of the incoming signals and the calibration source injected through or after the antenna.

Once the calibration source is added, $z_1(t)$ is demodulated at f_1 and $z_2(t)$ at f_2 to compute the respective complex envelops, where f_1 and f_2 are near the carrier frequency f_c.

$$\tilde{z}_1(t) = \tilde{y}_1(t) + \tilde{g}_1(t) = y_{c1}(t) + jy_{s1}(t) + g_{c1}(t) + jg_{s1}(t) \tag{18a}$$

$$\tilde{z}_2(t) = \tilde{y}_2(t) + \tilde{g}_2(t) = y_{c2}(t) + jy_{s2}(t) + g_{c2}(t) + jg_{s2}(t) \tag{18b}$$

Figure 22 shows the demodulation process of channel 1 signal, which would be similar to channel 2. The combined demodulated equations are shown in Equations 18a and 18b.

$$\tilde{y}_1(t) = A \exp\left[j2\pi \left(f_c - f_1\right) t + j\theta\right] \tag{19a}$$

$$\tilde{y}_2(t) = A \exp\left[j2\pi \left(f_c - f_2\right) t + jkd\cos\psi + j\theta\right] \tag{19b}$$

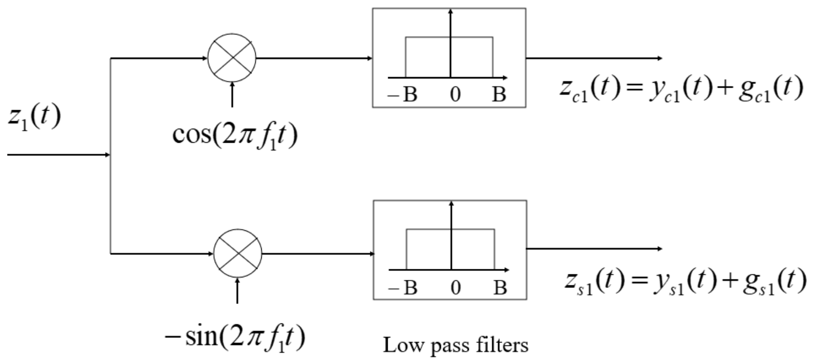

$z_1(t)$
$\cos(2\pi f_1 t)$
$-\sin(2\pi f_1 t)$
$-B$
0
B
$z_{c1}(t) = y_{c1}(t) + g_{c1}(t)$
$z_{s1}(t) = y_{s1}(t) + g_{s1}(t)$
Low pass filters

Figure 22: Example demodulation of Antenna 1 signal $z_1(t)$ performed by the FUNCube dongle and the resultant output real and imaginary parts.

Further, the complex envelopes of the two RF signals are represented in Equations 19a and 19b which can then be simplified further known that $\exp j\theta$ is a constant and incorporating it into A.

$$\tilde{y}_1(t) = A \exp\left[j2\pi\left(f_c - f_1\right)t\right] \tag{20a}$$

$$\tilde{y}_2(t) = A \exp\left[j2\pi\left(f_c - f_2\right)t + jkd\cos\psi\right] \tag{20b}$$

$$\tilde{g}_1(t) = G \exp\left(j2\pi\left(f_g - f_1\right)t\right) \tag{20c}$$

$$\tilde{g}_2(t) = G \exp\left(j2\pi\left(f_g - f_1\right)t\right) \tag{20d}$$

$$\tilde{z}_1(t) = A \exp\left[j2\pi\left(f_c - f_1\right)t\right] + G \exp\left(j2\pi\left(f_g - f_1\right)t\right) \tag{20e}$$

$$\tilde{z}_2(t) = A \exp\left[j2\pi\left(f_c - f_2\right)t + jkd\cos\psi\right] + G \exp\left(j2\pi\left(f_g - f_2\right)t\right) \tag{20f}$$

The simplified complex envelope signals (Equations 20a and 20b), the complex envelope of the small calibration signal (Equations 20c and 20d), and then complex envelopes of the signal plus the calibration signal (Equations 20e and 20f).

$$\tilde{z}_1(t) = A \exp\left[j2\pi\Delta f_{c1}t\right] + G \exp\left(j2\pi\Delta f_{g1}t\right) \tag{21a}$$

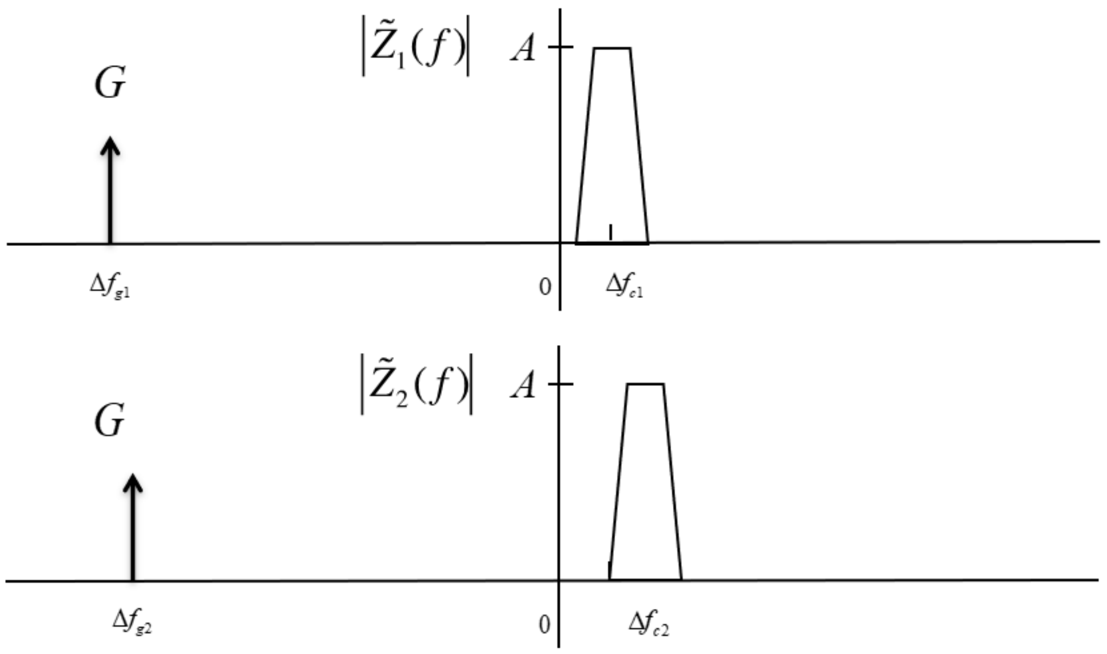


Figure 23: Example of frequency differences between the shifted baseband signal and calibration source and their respective bandwidth.

$$\tilde{z}_2(t) = A\exp\left[j2\pi\Delta f_{c2}t + jkd\cos\psi\right] + G\exp\left(j2\pi\Delta f_{g2}t\right) \tag{21b}$$

By renaming the $(f_c - f_1) = \Delta f_{c1}$, $(f_c - f_2) = \Delta f_{c2}$, $(f_g - f_1) = \Delta f_{g1}$, $(f_c - f_1) = \Delta f_{c1}$, and $(f_g - f_2) = \Delta f_{g2}$, the complex envelops then are can be written as Equations 21a and 21b.

As an example, assume that $|\Delta f_{c1}|, |\Delta f_{c2}| \ll |\Delta f_{g1}|, |\Delta f_{g2}|$ and that $|\Delta f_{c1}|, |\Delta f_{c2}| \sim 1kHz$ and $|\Delta f_{g1}|, |\Delta f_{g2}| \sim 10kHz$. A general schematic can be seen in Figure 23 showing the incoming digitally coded VHF tag signal with a larger bandwidth than the calibration signal. An FFT can be used to separate these signals.

Overall, the algorithm measures the phase associated with the spatial offset between the two antennas $kd\cos\psi$ and then computes ψ, the direction of arrival. Using an FFT, Δf_{g1} and Δf_{g2} can be estimated from their respective signals. Once found the following parameters are calculated: $f_1 = f_g - \Delta f_{g1}$, $f_2 = f_g - \Delta f_{g2}$, $\Delta f_{c1} = (f_c - f_1)$, $\Delta f_{c2} = (f_c - f_2)$. Finally, the phase difference can be computed between the two signals after shifting the complex envelops to baseband to solve for ψ.

5.2 Algorithm

In the radio tracking system, the RF signal is first shifted to the baseband in-phase (I) and quadrature (Q) components. These data are then saved digitally into a .wav file which is then loaded into MATLAB for processing (see Appendix D for all MATLAB code). The *chooseFlight.m* function allows the user to choose which flight they would like to process. The I/Q samples from the .wav file are combined to recreate the received complex signal and window the time block of interest. The complex data goes through a series of functions to eventually estimate the phase values of the received emissions from the test tag.

5.3 Equations and Code

The processing is subdivided below into the main functions used to estimate the tag bearings (see Figure 24).

1. The command *findpeaks* locates the first transmission after filtering the complex data to increase the SNR. For the purpose of solely increasing SNR and not focusing on peak definition yet, the filter for this step has a cut-off frequency that over filters the complex data (i.e., 1,000 Hz).

2. Once the first emission is located, the function *getInitialBounds.m* determines the start and end index/time the emission exists in and adds a buffer to each side. This windows the data, ideally as small as possible, to reduce the computational load. When the first ping group was a missed detection, the

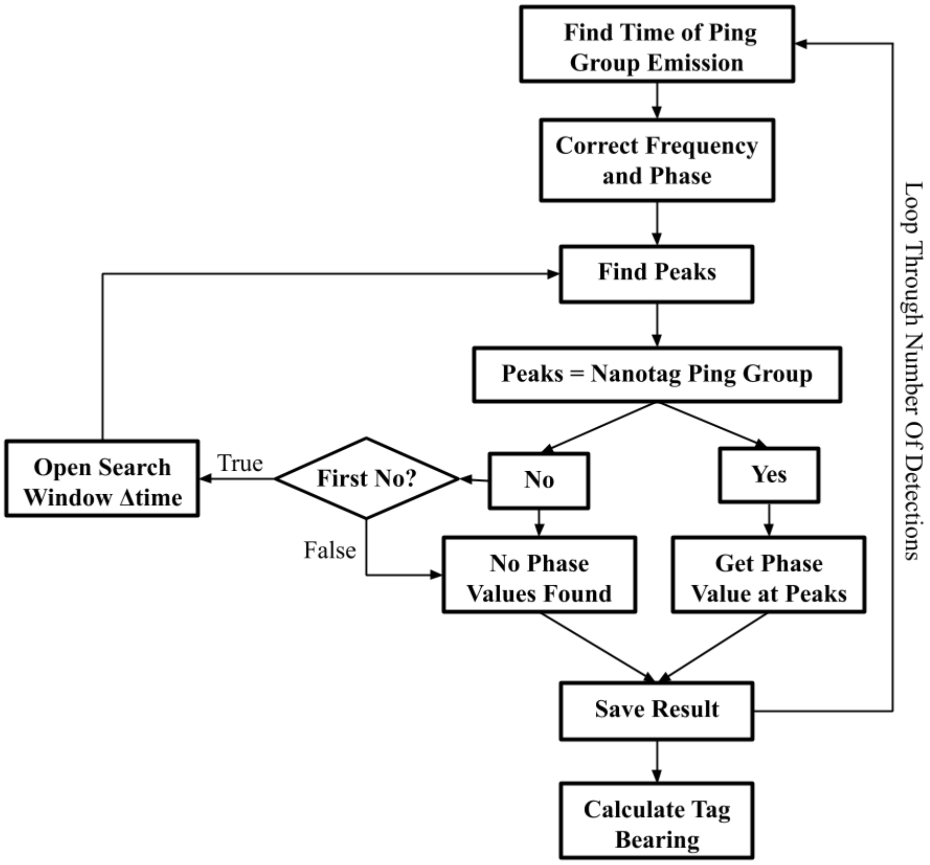


Figure 24: Block diagram outlining code process to estimate tag bearing.

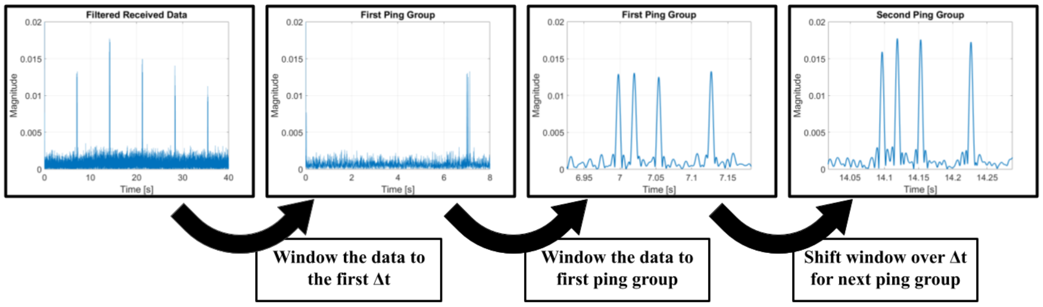


Figure 25: Automated ping group detection method.

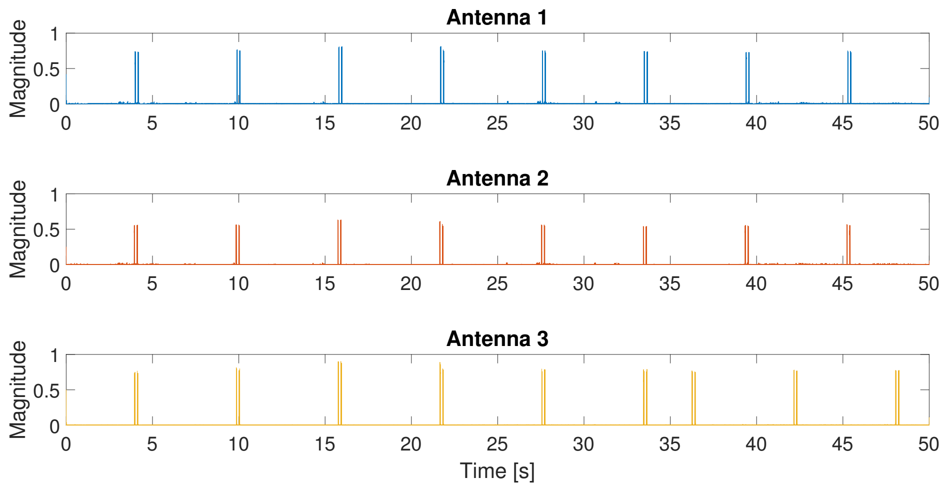


Figure 26: An example of filtered signals of fixed location data where there were missing data; see Antenna 3 at time = 36-43 seconds.

search window shifts over another δt iuntil detections are found. Because the test Lotek Nanotag emits approximately every 7 seconds, δt is set to this value. . The first detection is critical to code automation, as the first ping group determines where the window shifts (Figure 25).

If the window shift by δt does not find the 4 pings, the window is then opened up to size δt. This allows the code to find pings that were shifted in time if the data has some missing values (Figure 26). In this example. after 35 seconds on Antenna 3 two ping groups occur before a full δt interval had passed, which results in the ping groups not lining up with the Antenna 1 or 2 data. However, after searching the full 7 second window the shifted ping group is detected, this becomes the new starting point for the shifting window.

3. Once the ping group is detected from the window, the function *processData.m* processes the data to account for the accumulated errors from using equip-

ment with low precision and accuracy (see Chapter 7). This step is labeled as "Correct Frequency and Phase" (Figure 24) The script starts by calculating the frequency offset between the calibration source peak and zero. As stated in Chapter 3, the data were sampled with a center frequency of 166.385 MHz, which was the same frequency of the calibration source. This means that a FFT is expected to show the largest peak centered at 0 Hz. This was not the case and varied differently among FCDs.

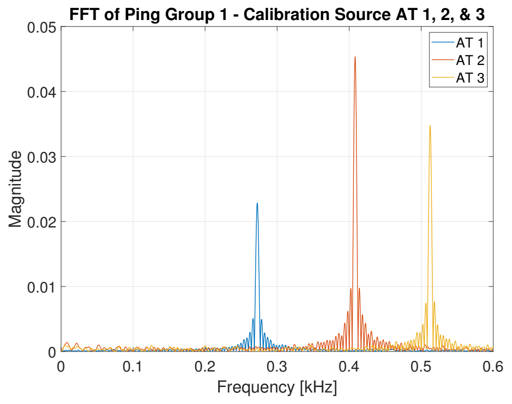
$$v = V_c . \exp(-i2\pi f_c t + \theta); \tag{22}$$

The frequency at the peak was then plugged into Equation 22 found in the function *demodulateSignal.m*. The variables are the demodulated signal as v, V_c is the complex signal, f_c is the carrier frequency, t is time and θ is phase. After the frequency offset is corrected by shifting the calibration source frequency to zero, the phase must also be corrected.

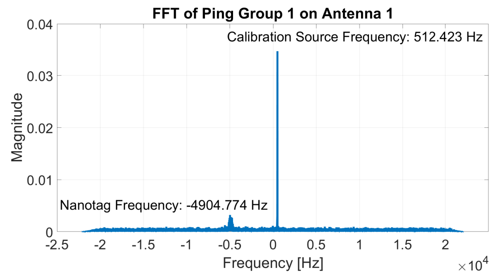
Examples of this frequency offset are shown in the FFTs in Figure 27. In Figures 27a, the FFTs are plotted to show variation among antennae. The calibration source frequency should align with 0 kHz for this shifted FFT.

When looking at the frequency offsets over 300 seconds through multiple FFTs, it was found that there is a local oscillator drift due to poor FCD stability. Figure 28 shows these offsets changing over time. The left y axis showing Antenna 2 offset ranging from 407.5 - 408.3 Hz (variation 0.8 Hz) and right y axis shows Antenna 3 ranging from 268.5 - 272.3 Hz (variation of 3.8 Hz). These average frequencies are also seen in Figure 27.

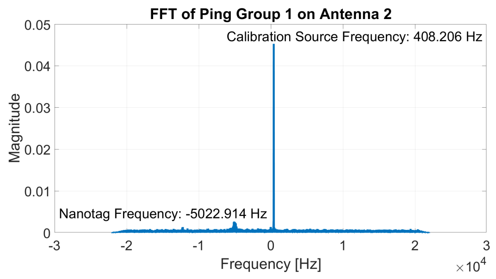
In Figures 27b, 27c and 27d show that the nanotag frequency shifts as well.



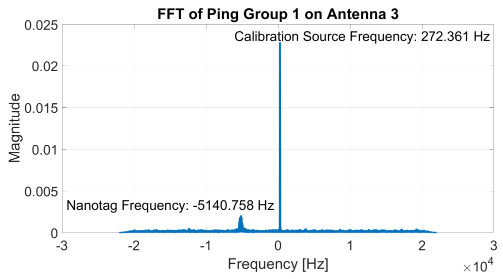
(a) FFT zoomed into the calibration source.



(b) FFT of window for emission 1 from Antenna 1 data.



(c) FFT of window for emission 1 from Antenna 2 data.



(d) FFT of window for emission 1 from Antenna 3 data.

Figure 27: FFT of each Antenna for the first emission in Flight 2.

0.5

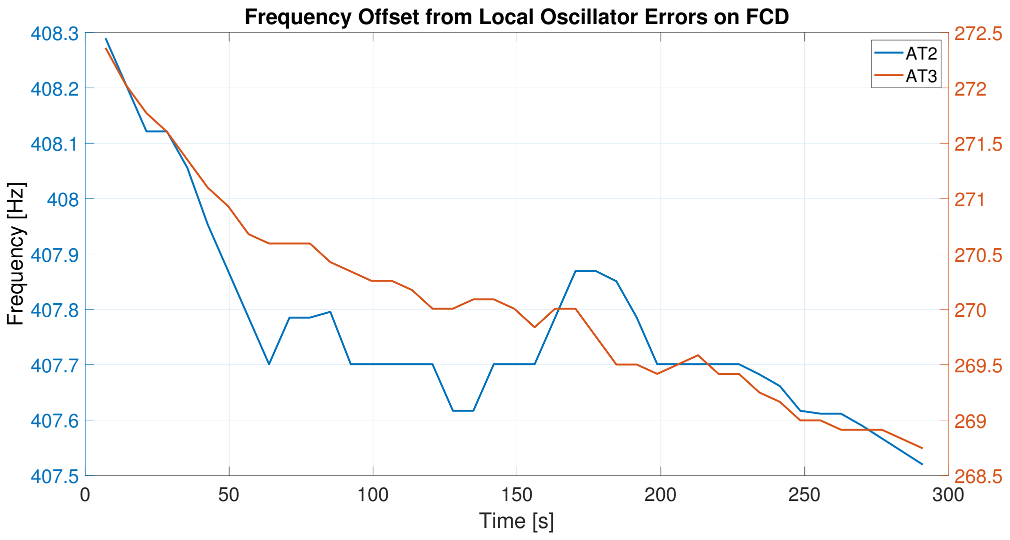
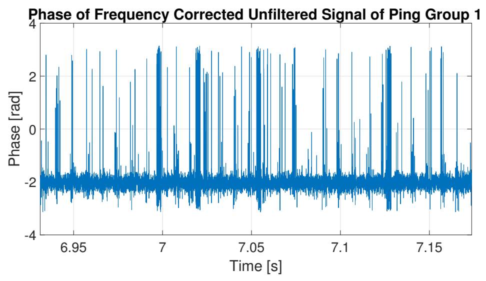
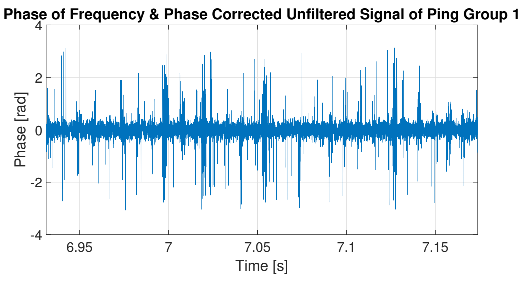


Figure 28: Frequency offsets over time as a result of the FCD local oscillators.



(a) Before phase correction



(b) After phase correction.

Figure 29: Phase of unfiltered demodulated signal centered at the calibration source frequency

This phase offset is found and corrected by $phaseCorrectedSignal = demodulatedSignal \exp -i(median(demodulatedSignalPhase)))$. In Figure 29a the offset shows approximately 2 radians. Once the phase has been corrected (Figure 29b), the demodulated signal is demodulated once more but with the carrier shifted to center the Nanotag's carrier frequency. Although the advertised center frequency is 166.380 MHz, the FFT (Figure 27) showed that the test tag had a frequency of approximately 166,379,606 Hz. These few hundred Hz differences are needed to be accounted for to filter correctly and obtain a clear and distinguished peak from the pings.

The filtering method implemented was a 4th-order low pass Butterworth filter with a cutoff frequency of 200 Hz using the *butter* command in MATLAB. This command produces the transfer function coefficients to be passed into the *filtfilt* command that filters the data with zero phase distortion by filtering the signal in both the forward and reverse directions. An example of the filtered signal can be seen in Figure 30 with the phase values as well. The circles capture the peak time of arrival and magnitude which is used to grab the corresponding phase values. This process is repeated until all the ping groups have been found.

Finally, with all phase values saved, the function *getDOAFrom2Antennas.m* or

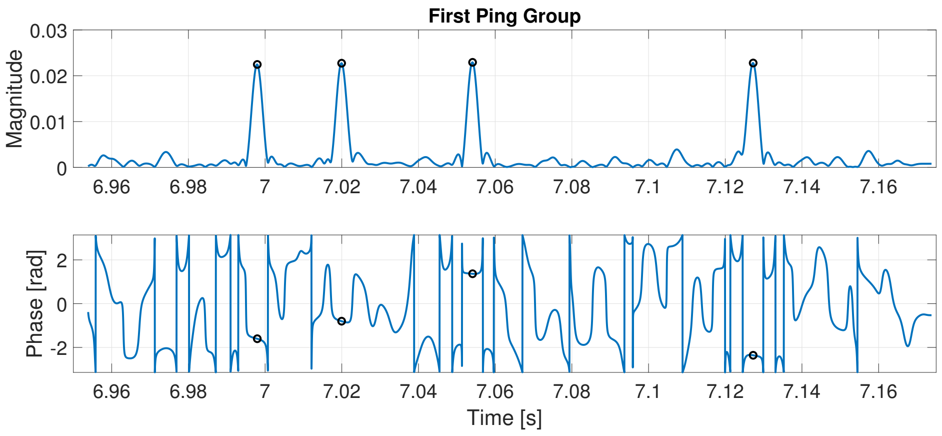


Figure 30: Filtered signal showing precise peak detection magnitude (upper panel) and the corresponding undistorted phase values (lower panel).

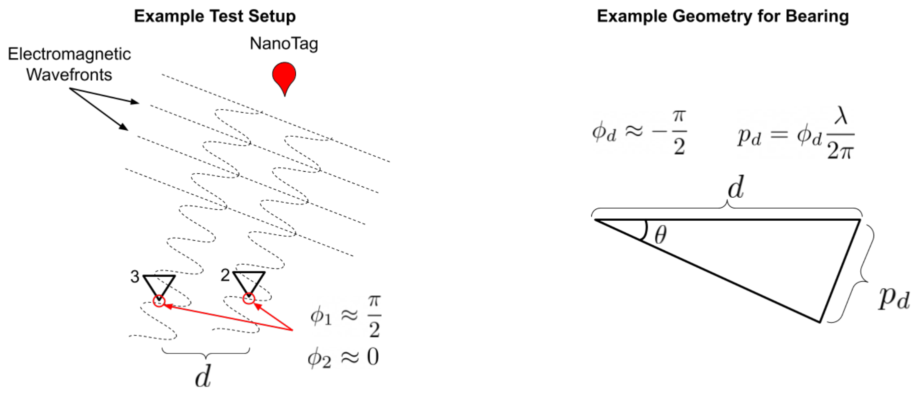
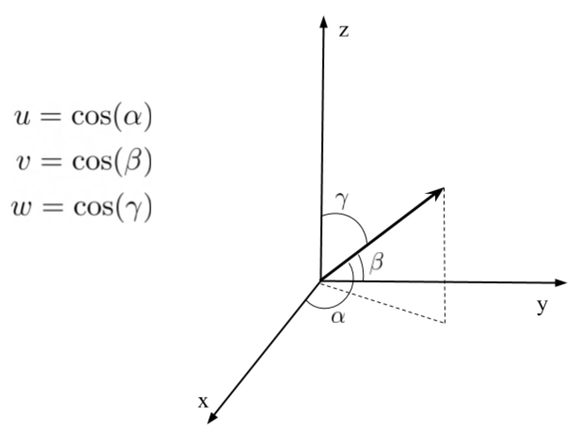

Example Test Setup
Electromagnetic Wavefronts
NanoTag
3
2
$\phi_1 \approx \frac{\pi}{2}$
$\phi_2 \approx 0$
d
Example Geometry for Bearing
$\phi_d \approx -\frac{\pi}{2}$
$p_d = \phi_d \frac{\lambda}{2\pi}$
d
θ
p_d

Figure 31: Visual representation of obtaining bearing estimations using 2 antennas.

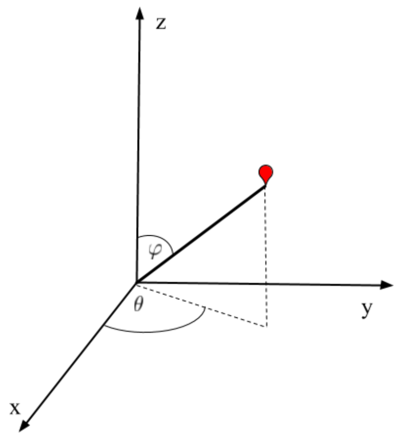
getDirectionOfArrival.m are used to estimate tag bearing. The function *getDOAFrom2Antennas.m* is implemented for calculations using only two antennas and is used as a benchmark testing method. These results will be explained in Chapter 6.

$$d_p = \phi_d \frac{\lambda}{2\pi} \tag{23}$$

$$\theta = 90 - \arcsin(d_p/d) \tag{24}$$



(a) Representation of bearing from direction cosines.



(b) Bearing and elevation angles with spherical angles.

Figure 32: Bearing and elevation schematics using direction cosines.

The main equations for function *getDOAFrom2Antennas.m* are outlined in Equations 23 where d_p is path difference and ϕ_d is phase difference and 24 where d is antenna distance. A visual is provided in Figure 31 for better understanding.

Function *getDirectionOfArrival.m* uses three antennas and is found by first solving for the direction cosines. Direction cosines are the angles that the vector to the NanoTag in space produces, shown in Figure 32a where the bearing and elevations angles can be calculated shown in Figure 32b.

This is done using the form $Ax = b$. With matrix A representing the geometry of the antenna array, vector b set as the phase differences multiplied by the inverse wave number and x set to the direction cosines. This equation is then solved by multiplying both sides of $Ax = b$ by the inverse of $A^T A$, shown in Equation 26.

$$x = (A^T A)^{-1} A^T b \tag{25}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \delta x_{21} & \delta y_{21} \\ \delta x_{31} & \delta y_{31} \\ \delta x_{32} & \delta y_{32} \end{bmatrix}^{-1} \begin{bmatrix} \delta\phi_{21} \\ \delta\phi_{31} \\ \delta\phi_{32} \end{bmatrix} \frac{\lambda}{2\pi} \tag{26}$$

The inverse tangent of the u and v of the direction cosine, seen in Equation 27 to get a bearing calculation.

$$\theta = \tan^{-1}(\frac{v}{u}) \tag{27}$$

Once these calculations were done, the bearing estimates were compared to the actual known bearing estimates based on the drone location calculated using the latitude and longitude positions of the drone and antenna positions.

CHAPTER 6

Estimating the Bearing to a Tag With Two Antennas

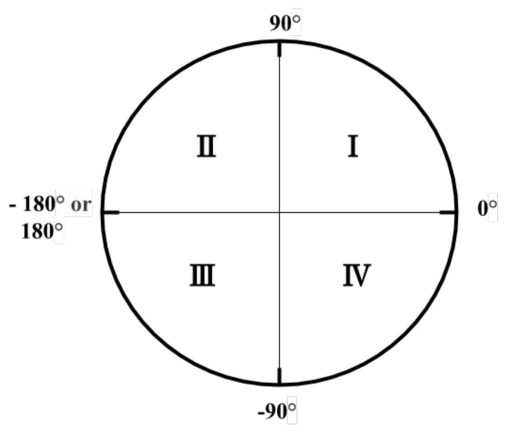
6.1 Method

Function *getDOAFrom2Antennas.m* (Appendix B), uses the phase difference between two antennas (e.g., #2 and 3) to estimate the bearing to a test nanotag. One limitation of using only two antennas is that it is uncertain exactly where the tag is located in relationship to the antenna (Figure 33).

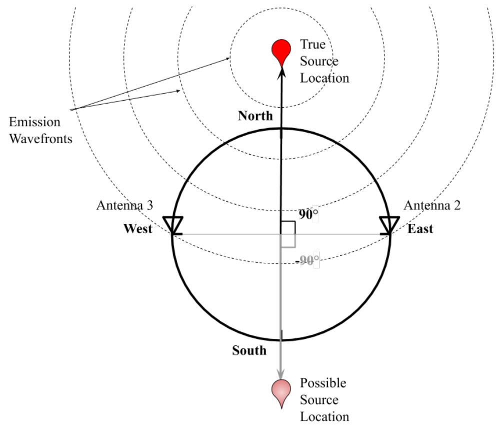
For example, although the estimated bearing was predicted to be in quadrant I/II, the tag could have been quadrant III/IV. This uncertainty makes a two-antenna system unable to get a full 360° bearing estimations without *apriori* knowledge. There were both drone coordinates and notes taken during the drone flights for manually changing the angles to the correct side (Appendix A). Although this method is applicable for testing, a two antenna system would not be practical for bird tracking.

One of the challenges during testing was that the FUNCube dongles drifted in frequency (i.e., ±1.5 ppm). This was evident after performing a FFT on the data recorded using SDR with a center frequency set to 166,385,000 Hz, which was the same frequency as the calibration source. The peak was off center (discussed and shown in Chapter 5). The test tag frequency also differed from the manufacturers specifications but processing accounted for these drifts using the function *getCalibrationSourceFrequencyOffset.m*, (Appendix A).

To determine the reliability of the system with two antennas, Flight 2 was used for processing. This was a helpful benchmark as the phase differences were expected



(a) Defining the quadrant space for consistency.



(b) Visual representation of bearing angle uncertainty within the quadrants III and IV.

Figure 33: Visual understanding of the defining quadrant system and the downside of having a two antenna DF system.

to be or close to zero. This aided in the finalization of the code and steps necessary to achieve the correct phase difference for the bearing estimations.

6.2 Results

With the test tag carrier frequency shifting throughout the recording, different corrections were added to the test tag frequency for demodulation and filtering. However, the peak is difficult to distinguish in the FFT because of the short duration of a transmission and low energy transmissions. As a result the first correction, 420 Hz, was chosen by lining up the test tag frequency "bump" in energy to zero by eye. After this, a series of FFTs were taken for every emission with a wide windowed moving average filter to catch the upper average limit of the FFT. To ensure that the bump in energy was the test tag and not an unknown transmission, the FFT limited to a window of 1 kHz around the expected carrier frequency That peak was then recorded and averaged to get the second frequency correction of 394.0903 Hz. These corrections are subtracted from the advertised frequency of 166.380 MHz during processing.

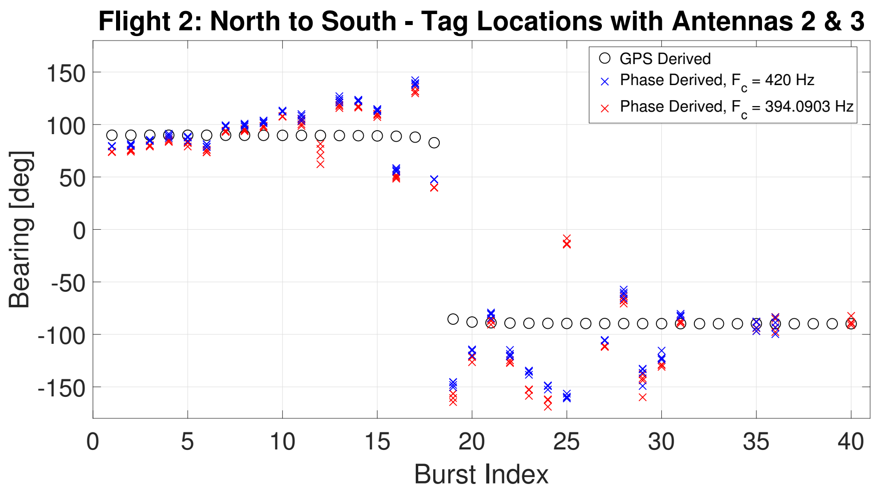


Figure 34: Comparison of bearing estimate of the known tag location (drone) compared to two estimates based on phase calculations using different carrier frequencies.

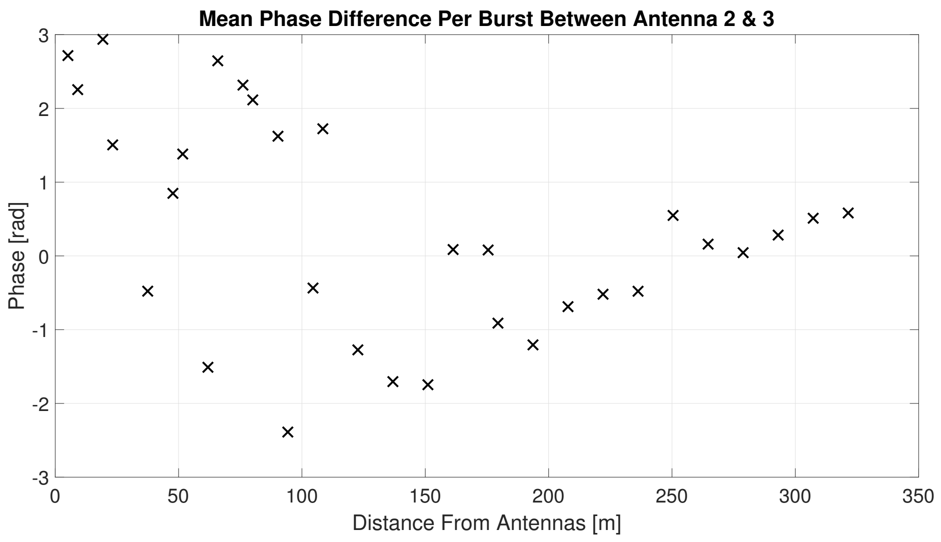


Figure 35: Measured error from the mean of the bursts bearing estimations from phase and the drone bearings with respect to distance from antennas.

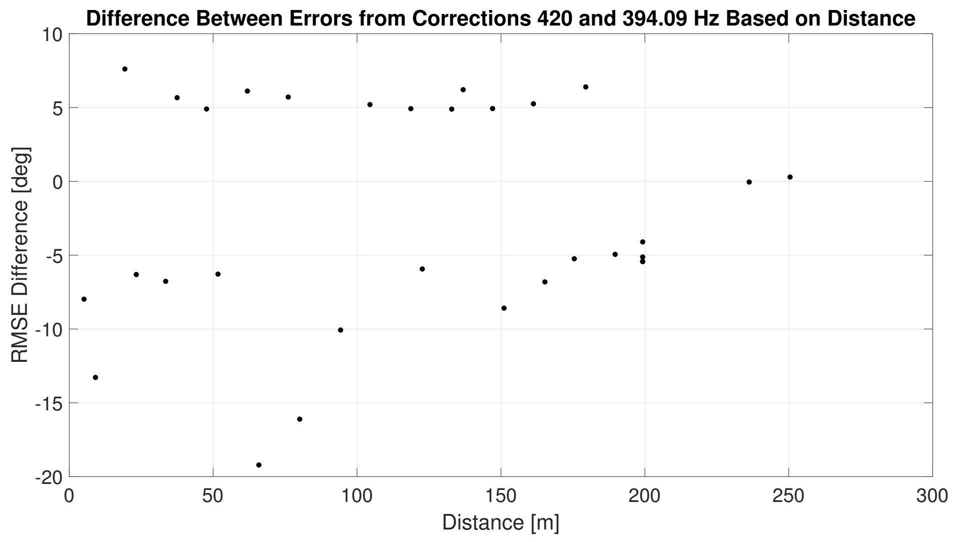


Figure 36: Measured error from the mean of the bursts bearing estimations from phase and the drone bearings with respect to distance from antennas.

Figure 34 includes the final results from Flight 2, showing the true bearing given by the GPS on the drone and the estimated bearings based on phase values for two different frequency corrections. For this test, it was expected that the phase difference equal zero because for a North to South transect the test tag would be equidistant from antennas 2 and 3. Figure 35 includes the phase differences averaged per burst versus distance. When the values are close to zero, the errors are small in this case.

By altering the carrier frequency of the test tag the errors also change. The two corrected values of 394.0903 Hz and 420 Hz are not consistent in accuracy (Figure 36). At some distances processing with one frequency performs better than the other but for both corrections the measured errors are at the largest the close the test tag is to the antenna. The estimated bearings show an error ranging from of 0.01° to 80° from the true bearing. After 175 m the measured errors are less than 20°. The errors from both the corrections seem to have no pattern when using looking at error versus distance.

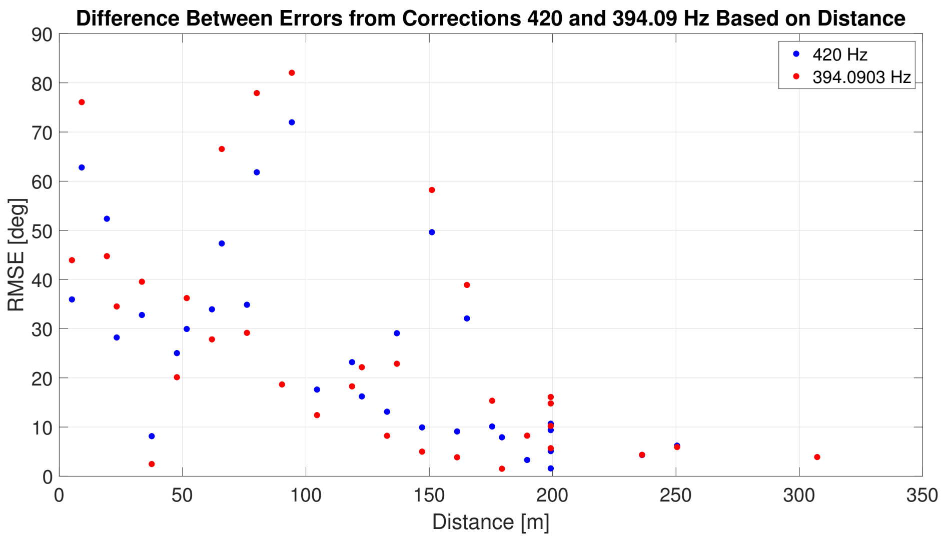


Figure 37: RMSE values over distance for the two correction frequencies.

Using the RMSE values demonstrate the distance and error relationship. In Figure 37, the majority of the RMSE values show a decreasing trend that is relatively linear until 200 m, where the errors begin to level out below 10°. At 175 m distance, bearing estimates are highly accurate (e.g., <15° at 225 m and the errors are <6° farther from the antennas), whereas when the tag approaches the antenna error estimates increase dramatically. The errors closer to the turbine are speculated to be effects of the beam patterns and being in near field where the electric field becomes unpredictable (<45 m). Overall, the average RMSE values for 420 and 394.04 Hz were 0.7750° and 0.8250° respectively.

Figure 38 compared the errors directly by taking the difference between the errors for both the measured errors and the RMSE values. With the exception of 177 seconds with a difference of 146°, the RMSE values show a more accurate representation of the differences between the frequency corrections. The reason being, is that the DOA estimate differences are based on the mean bearing of each burst, while the RMSE values take in the 4 bearing estimations per burst. Still, with error versus time there is no pattern showing consistency. However, at 120 seconds

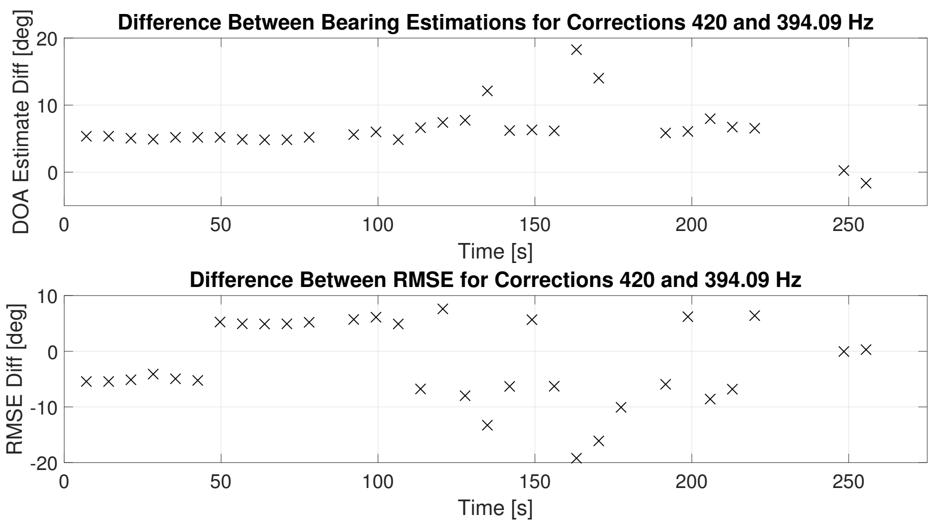


Figure 38: Differences in DOA estimations (upper panel) and the RMSE errors (lower panel) between frequency corrections 420 and 394.09 Hz. The upper panel excludes an outlier at burst index 25 (t= 177s), where the difference was 146°.

the test tag is above the antenna system and the differences in error begin to be more varied rather than the beginning 100 seconds where the errors had been more level.

Additionally, the FFT estimated carrier frequency varied greatly when there was low SNR. In Figure 39 the full fluctuations in the carrier frequency can be seen lined up with the ping groups. When the noise flow rises the variations in the approximations of the test tag carrier frequency increase.

Figure 40 shows a 55 second time span from Figure 39, which clearly show large variation in the carrier frequency, which makes estimating tag bearings based on the phase values less accurate (see Chapter 7).

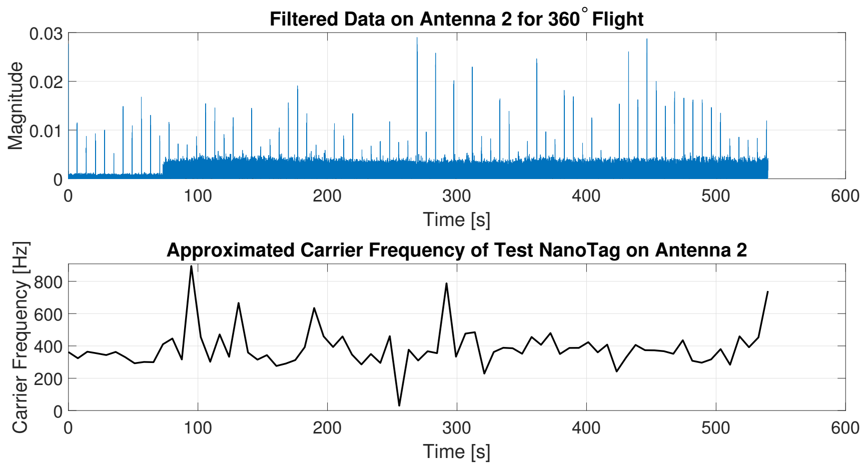


Figure 39: Example of the filtered signal on Antenna 2 for Flight 2 with carrier frequency calculations (lower panel) lined up with tag detections (upper panel) over a 55 sec time span.

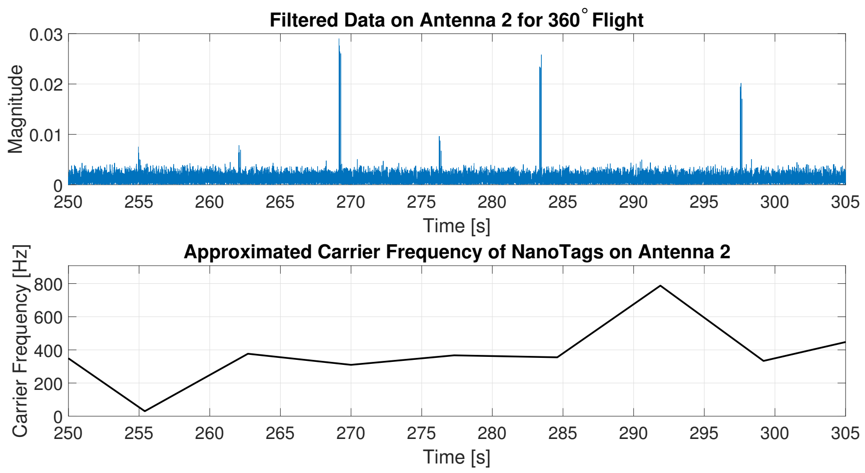


Figure 40: Example of the filtered signal on Antenna 2 for Flight 2 with carrier frequency calculations (lower panel) lined up with tag detections over a 55 sec time span.

CHAPTER 7

Conclusion: Sources of Error and Future Work

The DOA system was limited to two antennas for the results of this thesis. The main reason for this is due to data sets from one of the antennas having missing values of ping groups to orient data form other towers. Luckily the minimum number of antennas needed is two, if there is *apriori* information.

7.1 Direction Finding Error Sources

Errors have the ability to accumulate based on the four major categories of error sources: propagation-induced, environmental, instrumental and observational errors [1]. Propagation-induced errors refer to the signal traveling through different mediums, such as the ionosphere, that causes degradation in DOA, signal amplitude and can cause time dispersion effects. For VHF systems these errors are more likely to be a result of surface roughness. Over sea surfaces specifically include RF signal reflections thus causing decreased propagation distance.

The environmental errors can be any natural or man-made obstacle that can cause scattering or redirection of the electromagnetic wavefront causing deviation from the linear path. This error is divided into very near (which is within $\lambda/2\pi$), near (5 - 10 wavelengths), and far region (>5 - 10 wavelengths). Very near errors occur from the space on the platform the DF antennas are placed. An example error is antenna systems set up on the ground as it can hold dielectric constant and conductivity variations that can cause DOA errors. Near errors are caused by point and extended rereadiators and earth and terrain irregularities such as seasonal vegetation changes in an area which can affect the antennas received signals. Far

	Factors	**Reason**	**Result**
1	Low SNR	Internal receiver noise	Missed detections
2	Amplitude and phase unbalance	Poor design, Signal: amplitude and bandwidth, Physical: temperature and aging	Difficult to process
3	Time and frequency inaccuracies	Mistuned RF filters	Time based errors - errors in digital processing, Frequency offsets - mismatched phase shifts from multiple
4	Hardware imperfections and aging	Faults in hardware, aging	Limited dynamic range, amplitude and phase instability
5	Physical misalignment	Imprecise installation	Not optimal antenna setup, dynamic platform, spacing inaccuracy
6	Digital processing and algorithm imprecision	ADC create quantization noise, Time-sampling errors, Software algorithm imprecision, Software interpolations to reduce memory cost	Invalid assumptions and mathematical errors, Errors in digital data
7	Calibration inaccuracies	Measurement inaccuracies	Residual errors

Table 1: Equipment imperfections that cause instrumental errors and their effects on the phase detection system [from H. H. Jenkins 1991, p. 53].

region is anything beyond the near region and can consist of reradiators such as aircraft, objects that are far but reflective or also emitting signals. Overall a ground-based DF site must include the following to minimize environmental errors: set up on high, level and clear terrain, the ground must be uniform in conductivity and moisture, and removed from reradiating features such as man-made buildings, power lines and coastlines.

Instrumental errors are derived from imperfections in the equipment or amplitude and phase errors resulting from the use of multiple receivers. Examples and details can be seen in Table 1.

Observational errors apply when there is a display of DF results showing corrupted data, there tend to be large fluctuations that could be reduced by averaging. It could also be that the human factor caused some error if they are operating the system. In the case of this thesis, observational errors were not considered.

Looking at these errors relative to direction finding in general, and the errors

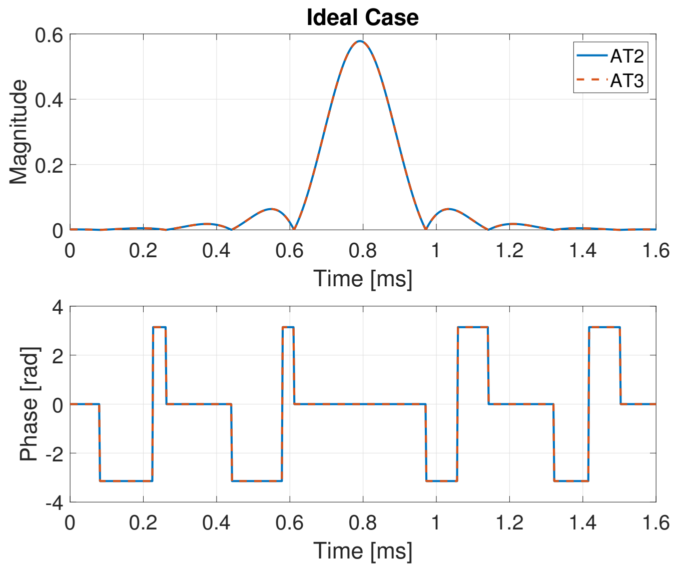
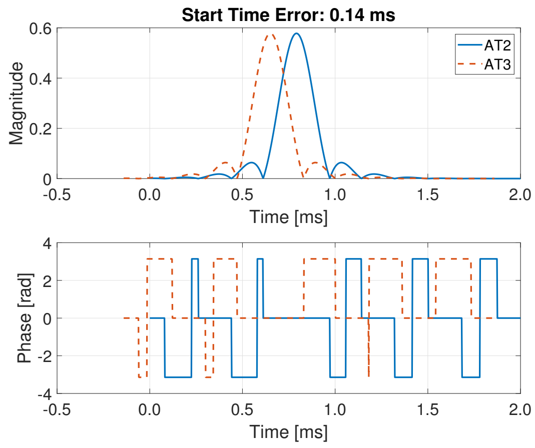


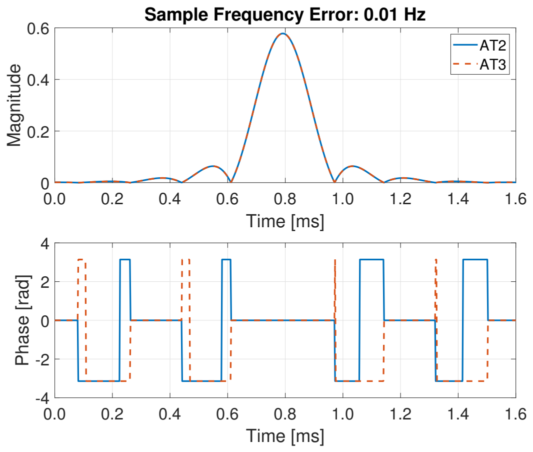
Figure 41: Ideal Case.

received on the post-processing front, there were clearly instrumental and environmental errors. The instrumental errors were the mostly due to the FUNCube dongles RF mixer and analog to digital converter. These errors are discussed in more depth below. The environmental errors were not as clearly visible but assumed to be there due to the area in which the testing took place. Metal buildings surrounded the antenna system and reradiation of the incoming electromagnetic waves are likely have occurred. However, still it is important to keep the other errors in mind when developing a more permanent system.

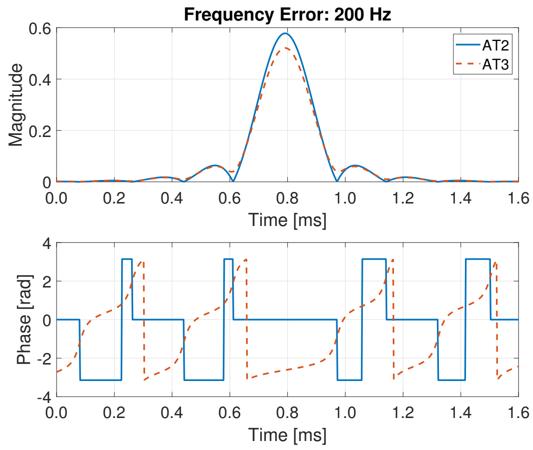
In addition to understanding common errors seen from literature review, modeled data of the system was also tested with the same functions developed for processing the test data to ensure that they worked correctly, shown in Figure 42. Each of the errors were added to the ideal case data (Figure 41) by itself before adding multiple error combinations. This was important to determine which errors had the greatest impact on the accuracy and precision of bearing estimates and how to fix these errors. For this exercise, Antenna 2 modeled data was unaltered, while Antenna



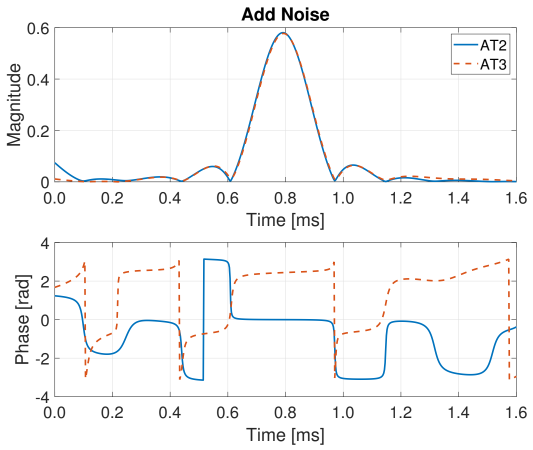
(a) Start Time Error of 0.14 ms Case.



(b) Sample Frequency Error of 0.01 Hz Case.



(c) Frequency Error of 200 Hz Case.



(d) Low SNR Case.

Figure 42: Different variables that were tested to observe the changes of the phase values in modeled data.

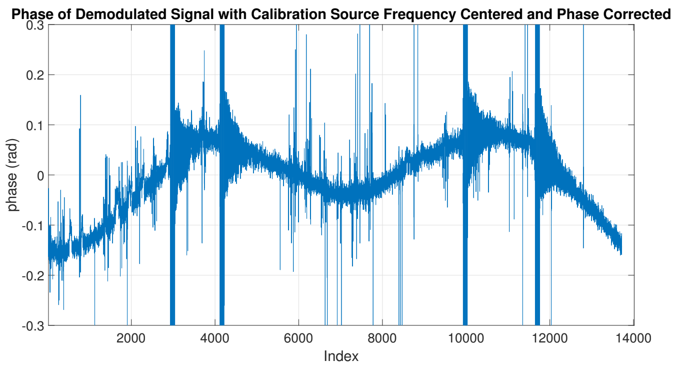


Figure 43: Phase of ping group 1 from using the RF Explorer as the calibration source.

3 data was modeled for errors. The more important errors contributing to phase inconsistencies were frequency errors and SNR (Figures 42c and 42d). Start time errors did not affect the phase values (Figure 42a) or did the sample rate errors 42b. The sample frequency error could create some inaccuracies if phase values of the transmission were sloped or the sample frequency error was much larger. However, the analysis performed here assumed that the error was small.

Finally, the reason that the calculations for two antennas was possible is due to the calibration source stability. In the beginning tests, the RF explorer was a calibration source with a frequency stability of ± 0.5 [2]. This is evident when looking at the phase inconsistencies in Figure 43

7.2 Conclusion

The FCDs were sources of large variation. Each FCD had a difference frequency offset by hundreds of hertz, which would change over time, thus needed to be constantly calculated. This source of was accounted for by including a precise calibration source, a rubidium clock with a stability of $<\pm 0.0001 ppm$ [3]. The calibration source allowed errors in frequency and phase to be corrected. A primary sources of the frequency and phase errors specifically were from the FCD local

Challenges	**Effect**
Clock Differences	Possibly lining up wrong peaks – phase difference errors
Software Memory Usage	Computer freezing – missing data, lining up wrong peaks
FUNCube Dongle Stability	Errors in frequency, affect frequency corrections
Nanotag Frequency Shifts	Frequency correction errors
Frequency Correction	Peak variability and phase correction errors
Peak Variability	Phase value errors
Phase Correction	Large phase errors
Signal-to-Noise-Ratio	Large phase errors
Uncertainties in Antenna and Drone Coordinates	Uncertainties of true bearing angles for phase bearing comparisons

Table 2: Table of all contributors of errors and their effects on the system.

oscillator drifting from the carrier frequency, the ADC timing not being precise enough, and propagation noise. These are errors that were documented in the literature review and were reviewed through modeled data in Figure 42.

This research showed there phase measurements are sensitive to error and can be corrected for through processing and higher stability equipment. With the rubidium clock calibration source, a majority of the frequency and phase errors from the FUNCube dongle were corrected. However, the Nanotag instability, data dropouts on the Antenna 1 system and general lack of coherency between systems lead to the calculation of bearing with two antennas. Despite the limited number of drone flights and using two antennas, this thesis provides an operational assessment of transmitter location.

7.3 Future Design

Suggested improvements for a next generation of the system are having a common LO to achieve phase coherence. The LO must be phase stable and have low phase noise and a reference clock synchronization to ensure all measurements are lined up.

The RSPduo and Low Jitter Precision GPSDO Reference Oscillator work well together for these SDR purposes, Figure 44.

(a) RSPduo Dual Tuner 14-bit SDR [4].

(b) Low Jitter Precision GPSDO Reference Oscillator [5]

Figure 44: Two main components to the new phase measuring system design.

The RSPduo specifications are the 0.5 ppm and has external clock input and output for easy synchronization of multiple RSPs or an external reference clock, in this case the Low Jitter Precision GPSDO. The oscillators important specifications are the stability of 0.001 ppm and a programmable frequency from 450 Hz - 800 MHz. When deploying this system the oscillator should be set to 166.385 MHz. The GPSDO Reference Oscillator will be input into one of the RSPDuos and outputted to the next RSPDuo and again with the last RSPDuo. Each RSPDuo will have its own respective Raspberry Pi 4. A simple schematic is shown in Figure 45 to show the connections.

From the research in this paper, having three separate computers made the processing difficult due to the FCD stability, having three separate local oscillators and depending upon the timing from the PCs that did not have enough memory to process the radio data. These errors should be accounted for by the new design. The RSPDuos will be timed on the same clock from the GPSDO Reference Oscillator rather than the Raspberry Pi clocks. The reason that three single board computers are used rather than one is because radio processing can be very intensive. In fact, the Raspberry Pis must be contained in a cooling case during

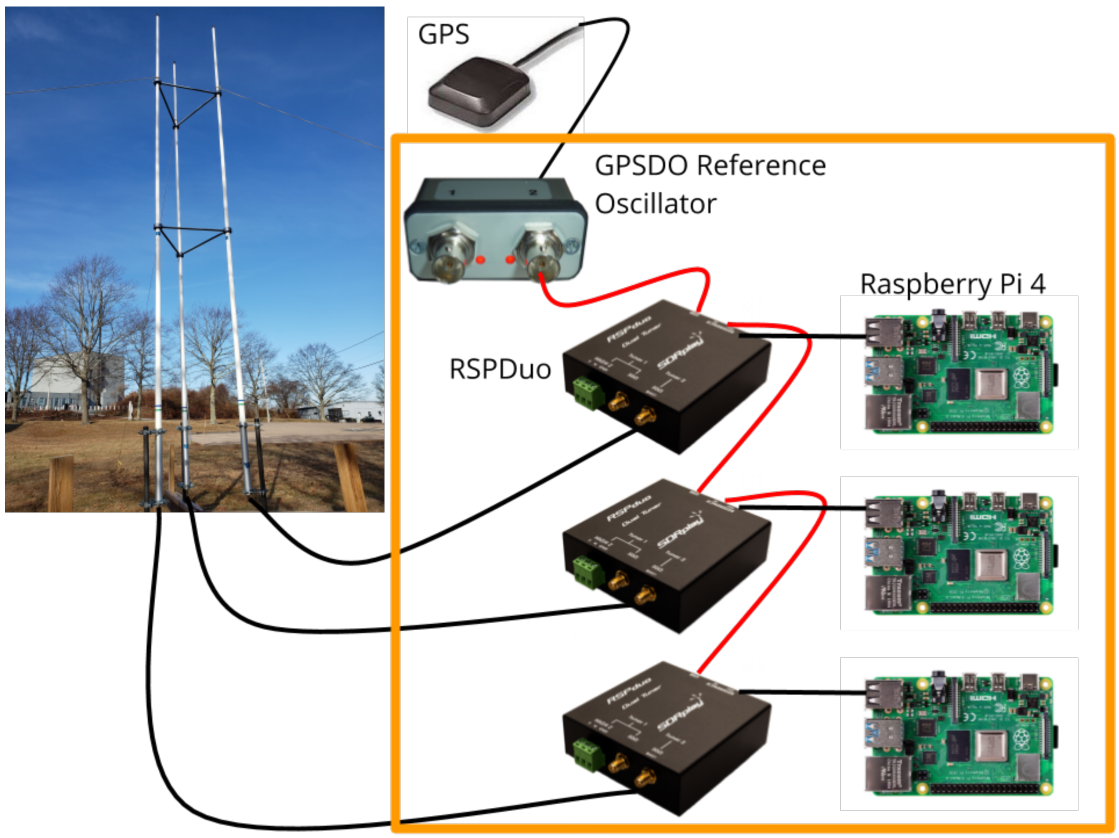


Figure 45: New system design schematic showing connections between the equipment inside deployable box.

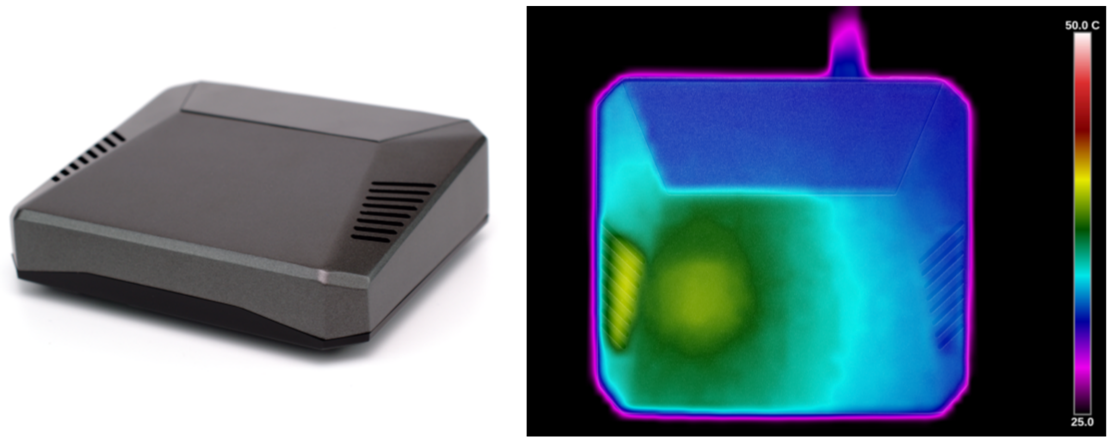


Figure 46: Argon ONE cooling case shown on the left with its respective thermal imaging after a heavy synthetic workload for 10 minutes [6].

radio processing. Radio data recording can be very intensive, especially over long duration of time and hot weather.

The Argon ONE shown in Figure 46 covers the Raspberry Pi to keep the debris out and kept the temperature below approximately 55°. All of these components pricing and sites are in Appendix E.

Future tests should include a drone test to determine the beam patterns of the antenna system experimentally. The omnidirectional antennas used in this work are Franklin arrays and can be complex. Additionally, a study on the stability of the Nanotag.

List of References

[1] H. H. Jenkins, *Small-Aperture Radio Direction-Finding.* Massachusetts, United States of America: Artech House Inc., 1991.

[2] *RF Explorer Signal Generator*, Digi-Key Electronics, 2019, updated to Firmware Version 1.31. [Online]. Available: http://j3.rf-explorer.com/40-rfe/article/132-rf-explorer-signal-generator-rfe6gen-specification

[3] S. R. Systems, "Rf signal generators: Sg380 series," 2018, [Online; accessed Nov 8, 2018]. [Online]. Available: https://www.thinksrs.com/products/sg380.html

[4] *RSPduo Dual Tuner 14-bit SDR*, SDRplay, 05 2018, v0.6.

[5] "Precision gps reference clock." [Online]. Available: http://www.leobodnar.com/shop/index.php?main_page=product_info&products_id=234

[6] G. Halfacree, "Group test: Best raspberry pi 4 thermal cases tested and ranked," Feb 2020. [Online]. Available: https://magpi.raspberrypi.org/articles/group-test-best-raspberry-pi-4-thermal-cases-tested-and-ranked

Appendix A: Bill Of Materials

Materials needed to construct a 3-antenna tracking array

Items	Unit Price	Quantity	Cost
Galvanized Metal Mast Clamp Set for Style 476 Antenna	$144.99	3	$434.97
Low-Carbon Steel Round Tube 0.12" Wall Thickness 2" OD	$48.74	3	$146.22
2 in. x 4 in. x 4 ft. Premium Ground Contact Pressure-Treated Lumber	$3.17	6	$19.02
Low-Carbon Steel Rectangular Tube 1/8" Wall Thickness, 1-1/2" x 3" Outside	$19.82	3	$59.46
Low-Carbon Steel Round Tube 1/4" Wall Thickness, 3-1/2" OD	$127.38	1	$127.38
PVC Rod	$13.75	6	$82.50
Thick PVC rod	$34.71	1	$34.71
1/4 in. Zinc Flat Washer (100-Pack)	$5.25	1	$5.25
1/4 in.-20 Zinc Plated Hex Nut (100-Pack)	$6.57	1	$6.57
#9 x 3 in. Star Flat-Head Wood Deck Screws (5 lbs. per Pack)	$29.97	1	$29.97
1/4 in.-20 x 4 in. Zinc Plated Hex Bolt	$0.31	18	$5.58
Guy wires			
		Total:	$951.63

Antenna Components and Instruments

Items	Unit Price	Quantity	Cost
476 21' Classic VHF Marine Band Antenna	$859.99	3	$2,579.97
TWS400-115 - 115' Jumper with TWS400 3/8", 50 Ohm braided cable with B	$149.00	3	$447.00
SG382/4, 2 GHz generator (w/ opt. 04, Rb timebase)	$5,400.00	1	$5,400.00
HP Stream - 11-ak1010nr	$199.99	3	$599.97
FUNCube Dongle	$155.88	3	$467.64
Nanotag - NTQB2-4-2	$210.00	1	$210.00
SMA male to BNC female adapter	$11.99	3	$35.97
2.4 / 5.8GHz / 2 dBi @ 2.4GHz, (RP-SMA) rubber duck WiFi Antenna	$6.99	1	$6.99
Maxmoral N Male to SMA Female Connector RF Coax Coaxial Adapter	$5.99	1	$5.99
		Total:	$9,753.53
		Total Overall:	$10,705.16

Site
https://www.westmarine.com/buy/shakespeare--galvanized-metal-mast-clamp-set-for-style-476-antenna--3844016
https://www.mcmaster.com/7767t54
https://www.homedepot.com/p/WeatherShield-2-in-x-4-in-x-4-ft-Premium-Ground-Contact-Pressure-Treated-Lumber-274324/300526750
https://www.mcmaster.com/6527k47-6527K471
https://www.mcmaster.com/7767t75-7767T753
https://www.mcmaster.com/8745k47
https://www.mcmaster.com/87025k67
https://www.homedepot.com/p/Everbilt-1-4-in-Zinc-Flat-Washer-100-Pack-800452/204276405
https://www.homedepot.com/p/Everbilt-1-4-in-20-Zinc-Plated-Hex-Nut-100-Pack-802582/204274090
https://www.homedepot.com/p/Deckmate-9-x-3-in-Star-Flat-Head-Wood-Deck-Screws-5-lbs-per-Pack-3DMT5/305418479
https://www.homedepot.com/p/Everbilt-1-4-in-20-x-4-in-Zinc-Plated-Hex-Bolt-800656/204633305

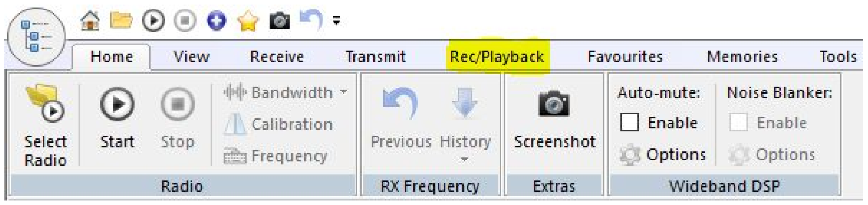
Site
https://www.westmarine.com/buy/shakespeare--476-21-classic-vhf-marine-band-antenna--297676
http://www.econ2way.com/
https://www.thinksrs.com/products/sg380.html
https://store.hp.com/us/en/pdp/hp-stream-11-ak1010nr
http://www.funcubedongle.com/?page_id=1113
https://www.lotek.com/products/nanotags/
https://www.amazon.com/Maxmoral-Female-Coaxial-Adapter-Connector/dp/B0114MWOD8/ref=sr_1_9?dchild=1&keywords=SMA%2Bmale%2Bto%2BBNC%
https://www.amazon.com/5-8GHz-2-4GHz-RP-SMA-rubber-Antenna/dp/B0093SGI0Q
https://www.amazon.com/Maxmoral-Female-Connector-Coaxial-Adapter/dp/B01NCAL4DR/ref=sr_1_4?crid=3B62KEGOSDFS5&dchild=1&keywords=n+type+

List of References

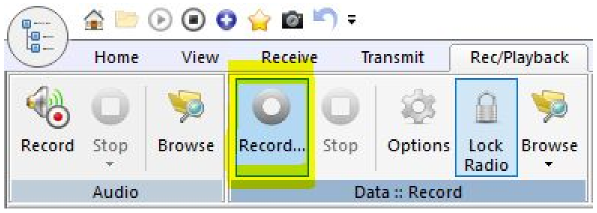
Appendix B: SDR-Radio SetUp

How To Set Up Recordings for Testing on SDR:

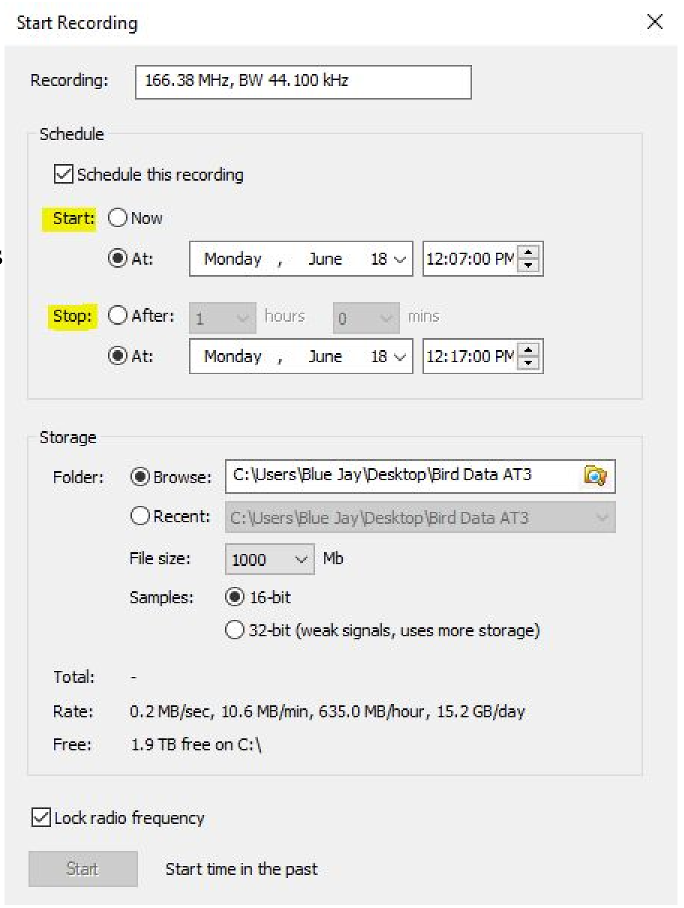
- Open SDR Radio Console Software and go to the Rec/Playback tab.



- In the Data Record section hit Record.



- Check the box labeled Schedule this recording.
- A menu will show up to set the start and stop time. You can change the directory if you would like. Currently it goes into a desktop folder labeled Bird data and the antenna it was attached to.
- Hit Start and the program will show a countdown for the scheduled recording.



Start Recording

Recording: 166.38 MHz, BW 44.100 kHz

Schedule

[✓] Schedule this recording

Start: ○ Now
● At: Monday , June 18 | 12:07:00 PM

Stop: ○ After: 1 hours 0 mins
● At: Monday , June 18 | 12:17:00 PM

Storage

Folder: ● Browse: C:\Users\Blue Jay\Desktop\Bird Data AT3
○ Recent: C:\Users\Blue Jay\Desktop\Bird Data AT3

File size: 1000 Mb

Samples: ● 16-bit
○ 32-bit (weak signals, uses more storage)

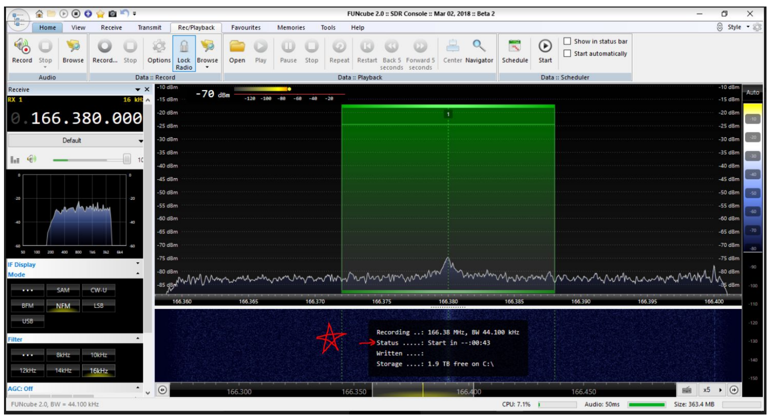
Total: -

Rate: 0.2 MB/sec, 10.6 MB/min, 635.0 MB/hour, 15.2 GB/day

Free: 1.9 TB free on C:\

[✓] Lock radio frequency

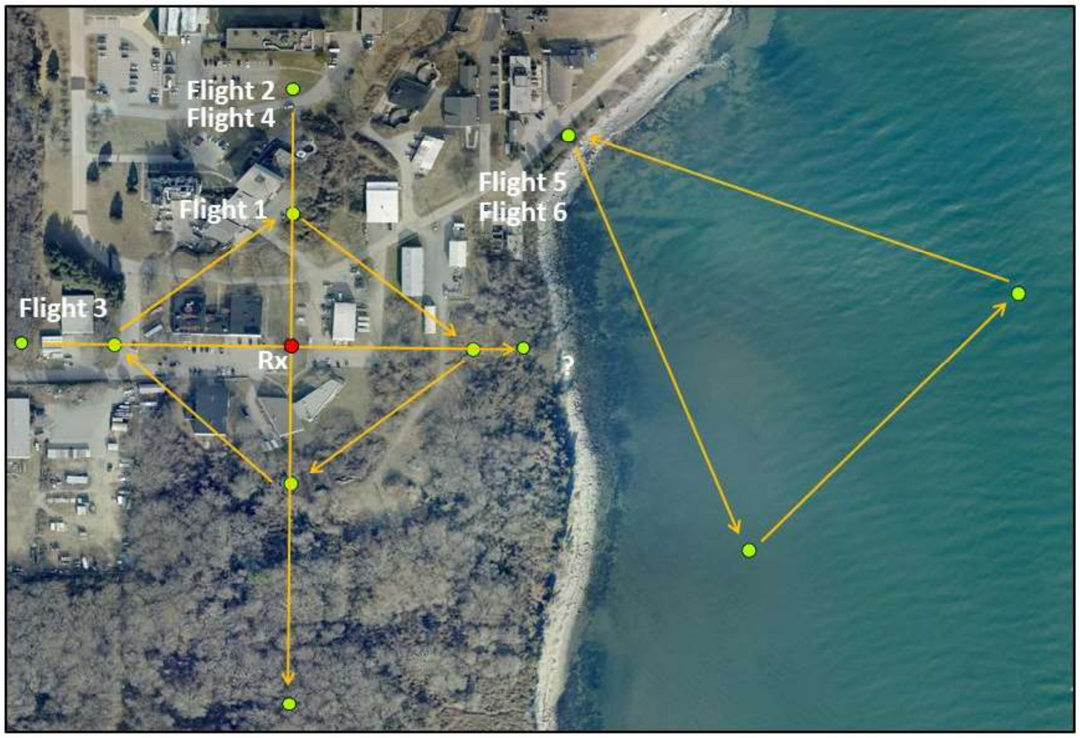
Start | Start time in the past



```
Recording ..: 166.38 MHz, BW 44.100 kHz
Status .....: Start in --:00:43
Written ....:
Storage ....: 1.9 TB free on C:\
```

List of References

Appendix C: Flight Plans



Every green dot represents a hovering point for 15 seconds.

Diamond:

Flight 1 has two parts, one with a speed of 2 m/s the other with 13 m/s

The pattern should lead to results of 360° of received transmissions. This could show is certain quadrants are weaker than others.

Straight flights on land:

Flight 2 and 4: Altitudes 60 and 120 respectively

Cross from North to South going over the antenna.

The goal is to have the results show constant North, tag disappears from being outside the beam, reappears showing constant South. It should help define the beam pattern limitations.

Flight over bay:

This flights will help to see the effects of range and small angle variation.

Flight #1:

Test at 11:32 AM

Diamond Test at Altitude 60 m and speed 2 m/s	
Time on File	**Notes**
0:00	Moving from North to East (therefore SE direction)
0:50	Halfway to East point
1:43	Reached East point, Noticed that AT2B the signal was very faint.
2:48	Halfway to South point, moving NE direction, strong gust of wind
3:50	Reached South point, waits for 15 seconds.
4:03	Gust of wind.
4:55	Halfway to West point.
5:43	Reached West point, waits for 15 seconds
6:56	Halfway to North point
8:08	Reached North point, waits for 15 seconds
Under Same File, Diamond at speed 13 m/s, with 15 s hovers	
9:09	Started moving towards East point
9:33	Started moving towards South point
10:07	Started moving towards West point
10:48	Started moving towards North point - But file stopped before reached North

General Notes	
North to East Leg	Tag seemed non existant to antennas, possibly due to hardware building in the way.
East to South Leg	Tag became more visible

Flight #2:

Test at 11:54

This test note section had weird notes.

North to South Transect at altitude 60 m and speed 2 m/s	
Time on File	**Notes**
0:00	Starting at North point, waits for 15 seconds. Then 18 seconds. (program error I believe)
0:33	Began transect.
2:05	Directly above antenna system.
> 2:05	All South after.

Flight #3:

Test at 12:03

West to East Transect at altitude 60 m and speed 2 m/s	
Time on File	**Notes**
0:00	Starting point, wait 15 second
1:59	Directly above antenna system
3:44	Reached East point
4:16	Flying back to North point

Flight #4:

Test at 12:10

North to South Transect at altitude 120 m and speed 2 m/s	
Time on File	**Notes**
0:00	Starting at North point, wait 15 second
2:01	Directly above antenna system. Windy*
4:49	Drone landing procedure

Flight #5:

Test at 12:25

Over bay test at altitude 60 m and speed 3 m/s	
Time on File	**Notes**
0:58	When flight begins, because the file started before the launch of the drone
2:25	75% to SE point of triangle
3:10	Reached the SE point, waits for 15 seconds
3:35	Started for NE point, however tag disappeared
4:33	Halfway to NE point
5:15	Reached Eastern point
6:40	Halfway to original location
?:??	Reached Northern point and hovered for last minute

Flight #6:

Test at 12:37

Over bay test at altitude 120 m and speed 3 m/s	
Time on File	**Notes**
0:24	Heading to SE point, *Very faint at 25% to SE
2:15	Wind caused sound floor to raise
2:33	Reached the SE point, waits for 15 seconds
3:00	Started for Eastern point, however tag disappeared
3:48	Halfway to Eastern point * tag disappeared
4:17	* tag reappeared
4:41	Reached Eastern point, waits 15 seconds
5:00	Flies due West, * tag disappears during this section
6:25	* tag reappeared
7:16	Reached Northern point
7:35	Drone landing procedure

List of References

Appendix D: MATLAB functions

```
clear all; clc; close all; warning off
%% Load constants
load('August2019_NanotagParameters.mat')
% global approximatedCorrection
% load ECUpdated.mat
extraCorrection = 350; %394.0903;
% extraCorrection = abs(mean(extraCorrectionsUpdated(:,2:3),2)); %; %420; %394.0903; %
    348.3514; 350
%% Decide which file you want to look at and the time interests

%Get data. ChooseFlight function holds the flight information.
flight_attempt = input('Pick a flight number 1-6: '); %Add flight descriptions?
[filename, flightDirectory] = chooseFlight(flight_attempt);

%Get time inputs and validate them
timeInterestStart = input('Start Time (seconds): ');
validateattributes(timeInterestStart,{'double'},{'integer','scalar'})

timeInterestEnd = input('End Time (seconds): ');
validateattributes(timeInterestEnd,{'double'},{'integer','scalar'})

timeCheck = [timeInterestStart, timeInterestEnd];
validateattributes(timeCheck,{'double'},{'integer','vector','increasing'})

%Make the time vector and a matching index vector. Make sure that there is
%an even index difference for calc_freq_diff.
durationInterest = timeInterestStart:timeInterestEnd;
idxTimeInterest = ((durationInterest(1):durationInterest(end))*SAMPLERATE)+1;
numberOfDetections = floor(length(durationInterest)/TIMEBETWEENPULSES);
pulseSampleIndex = TIMEBETWEENPULSES*SAMPLERATE;

%Colors used in subplots to match the main plot for presentaiton
color = {[0 .4470 .7410],[.8500 0.3250 0.0980],[0.9290 0.6940 0.1250]};

%% Read in the data and find detections

for fileNumber = 2:3

    filenameFull = strcat(flightDirectory, '\',  filename(fileNumber,:));
    [recordedData, SAMPLERATE]=audioread(filenameFull);
    time_seconds = (0:1/SAMPLERATE:(length(recordedData)-1)/SAMPLERATE)';
    timeSectionInterest = time_seconds(idxTimeInterest(1):idxTimeInterest(end));
    recordedDataSection = recordedData(idxTimeInterest(1):idxTimeInterest(end),:);
    complexSignal(:,fileNumber) = recordedDataSection(:,1) + 1j * recordedDataSection(:,2);

    [filteredSignal(:,fileNumber)] = getFilteredSignal(complexSignal(:,fileNumber), SAMPLERATE,
         ...
        CALIBRATIONFREQUENCY, timeSectionInterest, 100, 1, length(recordedDataSection), 420);



   %Plot the detections on the same plot
    figure(1)
    subplot(3,1,fileNumber)
    plot(timeSectionInterest, abs(filteredSignal(:,fileNumber)), 'Color', color{fileNumber})
    hold on
    title(['Filtered data for ', num2str(durationInterest(1)), ' to ', ...
        num2str(durationInterest(end)), ' seconds'],'FontSize',14)
    xlabel('Time (s)','FontSize',14)
    ylabel('Magnitude','FontSize',14)


    [indexPeakSearchStart, indexPeakSearchEnd, indexFreqDiffStart, indexFreqDiffEnd,
         pulseNumberDetected(fileNumber), peakThreshold(fileNumber)] ...
        = getInitialIdxBounds(filteredSignal(:,fileNumber), SAMPLERATE, pulseSampleIndex,
             PULSEDURATION);

    disp(['First detections found at ',num2str(timeSectionInterest(indexPeakSearchStart))]);

    for pulseNumber =  pulseNumberDetected(fileNumber):numberOfDetections-1

        [peakValues{pulseNumber,fileNumber}, peakIndex{pulseNumber,fileNumber},
             peakIndexSection{pulseNumber, fileNumber},...
            phaseDetected{pulseNumber,fileNumber}, timeDetected{pulseNumber,fileNumber}, ...
            timeUpdated{pulseNumber,fileNumber}, filteredSignalUpdated{pulseNumber,fileNumber},
                 phaseMeasurementsUpdated{pulseNumber,fileNumber},...
            frequencyOffet(pulseNumber, fileNumber)] ...
            = getCorrectedDetections(complexSignal(:,fileNumber), SAMPLERATE,
                 CALIBRATIONFREQUENCY, ...
            timeSectionInterest, CUTOFFFREQUENCY, PULSEDURATION, indexFreqDiffStart, ...
            indexFreqDiffEnd, indexPeakSearchStart, indexPeakSearchEnd, ...
            peakThreshold(fileNumber), pulseSampleIndex, extraCorrection);

%          meep(pulseNumber, fileNumber) = approximatedCorrection;

%        makePlots(timeUpdated, filteredSignalUpdated, phaseMeasurementsUpdated, ...
%            timeDetected, peakValues, phaseDetected, pulseNumber, fileNumber,
    numberOfDetections)

        [indexPeakSearchStart, indexPeakSearchEnd, indexFreqDiffStart, indexFreqDiffEnd] ...
            = getPeakIndexSearchLimits(peakIndex(:, fileNumber), pulseNumber, pulseSampleIndex,
                 indexPeakSearchStart, indexPeakSearchEnd, ...
            indexFreqDiffStart, indexFreqDiffEnd, SAMPLERATE);
    end

    [peakValuesMat(:,fileNumber), peakIndexMat(:,fileNumber), phaseDetectedMat(:,fileNumber),
         timeDetectedMat(:,fileNumber)] ...
        = addMissedDectectionsAsNaNs(pulseNumberDetected(fileNumber)-1, peakValues(:,fileNumber
             ), peakIndex(:,fileNumber), ...
        phaseDetected(:,fileNumber), timeDetected(:,fileNumber));

end

%% Calculations of for determining position of bird tag

%Get position matrix for calculations
[antennaDistanceMatrix] = getPositionMatrix(ANTENNADISTANCES);

%Phase differences between each antenna
```

```
phaseDifferences = [phaseDetectedMat(:,2)-phaseDetectedMat(:,1), ...
    phaseDetectedMat(:,3)-phaseDetectedMat(:,1), ...
    phaseDetectedMat(:,3)-phaseDetectedMat(:,2)]';

directionOfArrival  = getDirectionOfArrival(WAVELENGTH, antennaDistanceMatrix, phaseDifferences
    );



figure
plot(directionOfArrival, '*');

xlabel('Index')
ylabel('Angle [\circ]')
title('Direction of Arrival')
grid on
set(gca,'FontSize',20)


figure;
for i = 1:length(phaseDetected)
plot(ones(1,4)*i, phaseDetected{i,2}, 'r*'); hold on; plot(ones(1,4)*i,phaseDetected{i,3},'bs')
grid on; xlabel('Ping Group'); ylabel('Phase [rad]'); title('Phase Values Between Antenna 2 & 3
    ')
end

figure;
for i = 1:length(phaseDetected)
plot(ones(1,4)*i, [phaseDetected{i,2}-phaseDetected{i,3}], '*'); hold on;
grid on; xlabel('Ping Group'); ylabel('Phase [rad]'); title('Phase Differences Between Antenna
    2 & 3')
end
```

Listing 1: Main script to run the DOA processing.

```
function [filename, flightDirectory] = chooseFlight(flight_attempt)
%CHOOSEFLIGHT Summary of this function goes here
%   Detailed explanation goes here

codeDirectory = pwd;
cd ..\..\Summer2019\Aug9
flightDirectory = pwd;


switch flight_attempt
    case 1
        %First 10:52 11:00 10:50
        filename(1,:) ='Aug9 - AT1A\09-Aug-2019 113300.012 166.385MHz 000.wav';
        filename(2,:) ='Aug9 - AT2B\09-Aug-2019 113300.007 166.385MHz 000.wav';
        filename(3,:) ='Aug9 - AT3C\09-Aug-2019 113300.000 166.385MHz 000.wav';
        disp('Total Time = 10:50 minutes')

    case 2
        %Second 6:53 7:00 7:00
        filename(1,:) ='Aug9 - AT1A\09-Aug-2019 115500.016 166.385MHz 000.wav';
        filename(2,:) ='Aug9 - AT2B\09-Aug-2019 115500.018 166.385MHz 000.wav';
        filename(3,:) ='Aug9 - AT3C\09-Aug-2019 115500.014 166.385MHz 000.wav';
        disp('Total Time = 6:53 minutes')

    case 3
        %Third 6:00 6:00 6:00
        filename(1,:) ='Aug9 - AT1A\09-Aug-2019 120300.013 166.385MHz 000.wav';
        filename(2,:) ='Aug9 - AT2B\09-Aug-2019 120300.009 166.385MHz 000.wav';
        filename(3,:) ='Aug9 - AT3C\09-Aug-2019 120300.012 166.385MHz 000.wav';
        disp('Total Time = 6:00 minutes')

    case 4
        %Fourth 6:00 6:00 5:58
        filename(1,:) ='Aug9 - AT1A\09-Aug-2019 121000.015 166.385MHz 000.wav';
        filename(2,:) ='Aug9 - AT2B\09-Aug-2019 121000.009 166.385MHz 000.wav';
        filename(3,:) ='Aug9 - AT3C\09-Aug-2019 121000.005 166.385MHz 000.wav';
        disp('Total Time = 5:58 minutes')

    case 5
        %Fifth 9:00 9:00 9:00
        filename(1,:) ='Aug9 - AT1A\09-Aug-2019 122700.004 166.385MHz 000.wav';
        filename(2,:) ='Aug9 - AT2B\09-Aug-2019 122700.009 166.385MHz 000.wav';
        filename(3,:) ='Aug9 - AT3C\09-Aug-2019 122700.002 166.385MHz 000.wav';
        disp('Total Time = 9:00 minutes')

    case 6
        %Sixth 9:00 9:00 9:00
        filename(1,:) ='Aug9 - AT1A\09-Aug-2019 123700.007 166.385MHz 000.wav';
        filename(2,:) ='Aug9 - AT2B\09-Aug-2019 123700.019 166.385MHz 000.wav';
        filename(3,:) ='Aug9 - AT3C\09-Aug-2019 123700.006 166.385MHz 000.wav';
        disp('Total Time = 9:00 minutes')
    otherwise
        disp('There were only 6 flights, choose 1-6')
end

cd(codeDirectory)
end
```

Listing 2: Choose flight to analyze MATLAB scrip.t

```
function [indexPeakSearchStart, indexPeakSearchEnd, indexFreqDiffStart, indexFreqDiffEnd,
    pulseNumberDetected, peakThreshold] = getInitialIdxBounds(filteredSignal, sampleRate,
    pulseSampleIndex, pulseDuration)
%getInitialIdxBoundsForFreqDiff Summary of this function goes here
%   Inputs:
%       filteredSignal: the filtered signal that will be used for peak
%       picking
%       sampleRate: sampling frequency
%       pulseSampleIndex: the index between pulse duration
%       (sampleRate*pulseDuration)
%       pulseDuration: time between pulses
%   Outputs:
%       indexPeakSearchStart: the starting index for where the pulse starts
%       indexPeakSearchEnd: the ending index for where the pulse ends
%       indexFreqDiffStart: the starting index for calculating the
%       frequency offset
%       indexFreqDiffEnd: the ending index for calculating the
%       frequency offset
%       pulseNumberDetected: the number of the pulse that was found. In
%       otherwords, if a pulse was detected within relative time zero and
%       pulseDuration then the pulseNumberDetected is 1. The first pulse
%       detected was the first one expected, therefore there was no missed
%       detection, however if there was a missed detection, the
%       pulseNumberDetected will increase by one until there is a
%       detection.

skipSpike = floor(sampleRate/5); %Beginning and End have spikes
startSearchIndex = floor(skipSpike);
endSearchIndex =  floor(pulseSampleIndex)+sampleRate;%Plus a second of buffer

percentThreshold = 0.5; %June
peakThresholdMultiplier = 6; %6 August, 20 June
peakThreshold = mean(abs(filteredSignal))*peakThresholdMultiplier;
disp(['Peak threshold set to: ', num2str(peakThreshold)])
minimumPeakDistance = 0.01*sampleRate; %0.01 is the number of seconds

filteredSignalSection = abs(filteredSignal(startSearchIndex:endSearchIndex));
[peakValues, peakIndex, widths] = findpeaks(filteredSignalSection, 'MinPeakHeight',
     peakThreshold, 'MinPeakDistance', minimumPeakDistance, 'MinPeakWidth', 20);
    [peakValues, peakIndex, ~] = checkPeakDetections(peakValues, peakIndex, sampleRate, ...
    widths, minimumPeakDistance, [], filteredSignal, startSearchIndex, ...
    endSearchIndex, pulseDuration, peakThreshold, [], [], [], []);

pulseNumberDetected = 1;


while length(peakValues)~=4 || pulseDuration<(peakIndex(end)-peakIndex(1))/sampleRate
    startSearchIndex = startSearchIndex+pulseSampleIndex;
    endSearchIndex = endSearchIndex+pulseSampleIndex;

    if endSearchIndex>length(filteredSignal)
        error('Change the start and end time.')
    end

    filteredSignalSection = abs(filteredSignal(startSearchIndex:endSearchIndex));
%      peakThreshold = max(abs(filteredSignal(skipSpike:pulseSampleIndex)))*percentThreshold;
    [peakValues, peakIndex, widths] = findpeaks(filteredSignalSection, 'MinPeakHeight',
         peakThreshold, 'MinPeakDistance', minimumPeakDistance, 'MinPeakWidth', 50);
    [peakValues, peakIndex, ~] = checkPeakDetections(peakValues, peakIndex, sampleRate, ...
    widths, minimumPeakDistance, [], filteredSignal, startSearchIndex, ...
    endSearchIndex, pulseDuration, peakThreshold, [], [], [], []);

    pulseNumberDetected = pulseNumberDetected+1;
end



```

```

indexFirstPing = peakIndex(1)+startSearchIndex;
indexLastPing = peakIndex(end)+startSearchIndex;

indexSearchBuffer = round(sampleRate/10);
indexPeakSearchStart = indexFirstPing-indexSearchBuffer;
indexPeakSearchEnd = indexLastPing+indexSearchBuffer;

%Finds one second before and one second after
indexFreqDiffStart= indexFirstPing-sampleRate;
indexFreqDiffEnd = indexFirstPing+sampleRate-1;




end
```

Listing 3: Find the initial indices where the first ping group exists.

```
function [filteredSignal] = getFilteredSignal(complexSignal, sampleRate, frequencyCalibration,
    timeSection, cutoffFrequency, startSearchIndex, endSearchIndex, extraCorrection)
%GETFILTEREDSIGNAL takes the complex signal, finds the frequency difference
%generated by the dongles error. This funciton needs that the number of
%are even, using the getEvenIndexValue. Finds the demodulated signal with the
%frequency offset minused off. Then filters the demodulated signal with the
%reduce noise function to get the filtered signal that will be used for
%processing.
%   Inputs:
%       complexSignal: recorded data broken into real and imaginary
%       components
%       sampleRate: sampling frequency
%       frequencyCalibration: the calibration signals frequency above the
%       nanotags center frequency
%       timeSectionInterest: the time vector of the requested time
%       cutoffFrequency: the cut-off frequency used for filtering
%       startSearchIndex: where the complex signal starts to get the frequency difference
%       endSearchIndex: where the complex signal ends to get the frequency difference
%
%   Outputs:
%       filteredSignal: the filtered signal that will be used for peak
%       picking

%Currently doesnt work correctly?
% try
%     complexSignalSection = complexSignal(startSearchIndex:endSearchIndex);
% catch
%     %Not needed if FreqDiffStart = peakStart
%     if startSearchIndex < 0
%         complexSignalSection = complexSignal(1:endSearchIndex);
%     else
%         complexSignalSection = complexSignal(startSearchIndex:end);
%     end
% end

if length(startSearchIndex:endSearchIndex)<length(complexSignal)
```

```
    complexSignal = complexSignal(startSearchIndex:endSearchIndex);
    timeSection = timeSection(startSearchIndex:endSearchIndex);
end

frequencyOffset =  getFrequencyOffset(complexSignal, sampleRate) - frequencyCalibration -
    extraCorrection;
demodulatedSignal = demodulateSignal(complexSignal, timeSection, frequencyOffset); %Puts the im
     and real components into a complex equation to correct the frequency difference
filteredSignal = reducenoise(demodulatedSignal, cutoffFrequency, sampleRate); %Reduces noise of
     the demodulated signal using a butterworth filter or square filter


end
```

Listing 4: Filters complex signal using getFrequencyOffset, demodulateSignal, and reducenoise script.

```
function Xf = reducenoise(X,FC,FS)

[b,a] = butter(4,FC/(FS/2));
%b = ones(1,100)/100;
%a = 1;

Xf = filtfilt(b,a,X);
end
```

Listing 5: Applies a butterworth filter and filtfilt to complex data.

```
function [peakValues, peakIndexTrue, peakIndexSection, phaseDetected, timeDetected,
    timeSectioned, filteredSignalSection, phaseMeasurementsSection, frequencyOffset] =
    getCorrectedDetections(complexSignal, sampleRate, frequencyCalibration,
    timeSectionInterest, cutoffFrequency, pulseDuration, indexFreqDiffStart, indexFreqDiffEnd,
     indexPeakSearchStart, indexPeakSearchEnd, peakThreshold, pulseSampleIndex,
    extraCorrection)
%GETCORRECTEDDETECTIONS finds the newly filteres signal based on the
%frequency difference found one second before the pulse. The signal is then
%demodulated and filtered againw with the new frequency offset. From there
%the pulse of interest is found between the indexPeakSearchStart and the
%indexPeakSearchEnd index based on the peakThreshold and the hard coded
%peak width. From there it goes through the checkPeakDetections funciton
%and then either outputs the found detections or outputs NaNs if there were
%none.
%   Inputs:
%       complexSignal: recorded data broken into real and imaginary
%       components
%       sampleRate: sampling frequency
%       frequencyCalibration: the calibration signals frequency above the
%       nanotags center frequency
%       timeSectionInterest: the time vector of the requested time
%       cutoffFrequency: the cut-off frequency used for filtering
%       pulseDuration: maximum time of between the initial ping and the
%       last ping of a pusle
%       indexFreqDiffStart: the starting index for calculating the
%       frequency offset
%       indexFreqDiffEnd: the ending index for calculating the
%       frequency offset
%       indexPeakSearchStart: the starting index for where the pulse starts
%       indexPeakSearchEnd: the ending index for where the pulse ends
%       peakThreshold: the magnitude the peak must be above for detection
%       pulseSampleIndex: the index length between two pusles
%
%   Outputs:
%       peakValues: the magnitude value of the detected peaks
%       peakIndex: the index of the detected peaks
%       phaseDetected: the time of the detected peaks
%       timeDetected: the phase of the detected peaks
%       filteredSignal: the filtered signal used for peak detection
%       phaseMeasurements: the phase signal



%Finds peaks and locations
try
    timeSectioned = timeSectionInterest(indexPeakSearchStart:indexPeakSearchEnd);
catch
    disp('catch was used')
    if isnan(indexPeakSearchStart)
        keyboard
    end
    timeSectioned = timeSectionInterest(indexPeakSearchStart:end);
    indexPeakSearchEnd = length(timeSectioned)+indexPeakSearchStart-1;
end

skipSpike = 1000;
complexSignalSection = complexSignal(indexPeakSearchStart:indexPeakSearchEnd);
[filteredSignalSection, frequencyOffset] = processData(complexSignalSection, timeSectioned,
     sampleRate, cutoffFrequency, extraCorrection);
filteredSignalSection = filteredSignalSection(skipSpike:end);
timeSectioned = timeSectioned(skipSpike:end);

minimumPeakDistance = 0.01*sampleRate; %0.01 is the number of seconds


[peakValues, peakIndexSection, widths] = findpeaks(abs(filteredSignalSection), 'MinPeakHeight',
      peakThreshold, 'MinPeakDistance', minimumPeakDistance, 'MinPeakWidth', 23);


[peakValues, peakIndexSection, filteredSignalAdjusted, indexStartFull7Seconds] =
     checkPeakDetections(peakValues, peakIndexSection, sampleRate, ...
    widths, minimumPeakDistance, complexSignal, filteredSignalSection, indexPeakSearchStart,
         ...
    indexPeakSearchEnd, pulseDuration, peakThreshold, pulseSampleIndex, cutoffFrequency, ...
    frequencyCalibration, timeSectionInterest, extraCorrection);

%Final check to see if the code could not grab the peaks due to noise, but
%are pickable by eye.
```

```matlab

if isempty(peakValues)
    indexStartFull7Seconds = indexPeakSearchEnd-pulseSampleIndex;
    if indexStartFull7Seconds<0
        indexStartFull7Seconds=1;
    end
    complexSignal7Seconds = complexSignal(indexStartFull7Seconds:indexPeakSearchEnd);
    time7Seconds = timeSectionInterest(indexStartFull7Seconds:indexPeakSearchEnd);

    [filteredSignalFull7Seconds, frequencyOffset] = processData(complexSignal7Seconds,
         time7Seconds, sampleRate, cutoffFrequency, extraCorrection);
    filteredSignalFull7Seconds = filteredSignalFull7Seconds(skipSpike:end-1);
    [peakValues, peakIndexSection, filteredSignalAdjusted] = manualPeakPickingCheck(
         filteredSignalFull7Seconds, sampleRate, peakThreshold, minimumPeakDistance,
         pulseDuration);
end
peakIndexSection = round(peakIndexSection);




if ~isempty(peakIndexSection)
    peakIndexTrue = peakIndexSection+indexPeakSearchStart+skipSpike;%-1;


    %this if loop takes care of the signal did not hold the pulses, and needed
    %the window of search to be opened wide in the checkPeakDetections
    %function.
    if ~isempty(filteredSignalAdjusted)
        filteredSignalSection = filteredSignalAdjusted;
        peakIndexTrue = peakIndexSection+indexStartFull7Seconds;
        timeSectioned = timeSectionInterest(peakIndexTrue(1)-round(sampleRate/10):(
            peakIndexTrue(1)-round(sampleRate/10)+length(filteredSignalSection))-1);
        peakIndexSection = [round(sampleRate/10); (peakIndexTrue(2:end)-peakIndexTrue(1))+round
            (sampleRate/10)];
%
%         figure(28); plot(timeSectioned, abs(filteredSignalSection))
%         hold on; plot(timeSectionInterest(round(peakIndexTrue)), peakValues, 'o')
%         hold off;
    end
    timeDetected = timeSectioned(round(peakIndexSection));


    %
    phaseMeasurementsSection = angle(filteredSignalSection);
    phaseDetected = phaseMeasurementsSection(peakIndexSection);


elseif isempty(peakIndexSection)
    peakValues = NaN(4,1);
    peakIndexSection = NaN(4,1);
    peakIndexTrue = NaN(4,1);
    phaseDetected = NaN(4,1);
    timeDetected = NaN(4,1);
    phaseMeasurementsSection = [];
else
    error('getCorrectedDetections error on if loop')
end


% figure(9)
% plot(timeSeconds, abs(filteredSignalSection))
% hold on
% plot(timeDetected, peakValues, 'o')
% hold off
%
% figure(10)
% plot(timeSectionInterest, phaseMeasurements)
% hold on
% plot(timeDetected, phaseDetected, 'o')
% xlim([timeSeconds(1) timeSeconds(end)])
% hold off

end
```

Listing 6: Applies the frequency corrections to the data, finds the peaks to grab the phase values. Also uses checkPeakDetections and manualPeakPickingCheck

```
function [filteredSignal, frequencyOffset] = processData(complexSignal, time, sampleFrequency,
    cutOffFrequency, extraCorrection)
%processData Summary of this function goes here
%   Detailed explanation goes here
calibrationSourceFrequency = -5000;
tagOffset = calibrationSourceFrequency-extraCorrection;
[frequencyOffset, ~] =  getCalibrationSourceFrequencyOffset(complexSignal, sampleFrequency);
% global approximatedCorrection
% calibrationSourceFrequency = -5000;
% [frequencyOffset, nanotagFrequency] =  getCalibrationSourceFrequencyOffset(complexSignal,
    sampleFrequency);
% tagOffset = -(frequencyOffset + (44100-nanotagFrequency));

% if tagOffset < calibrationSourceFrequency-400 || tagOffset > calibrationSourceFrequency-280
%     %No nanotag detected on FFT, use default correction
%     tagOffset = calibrationSourceFrequency-340;
% end
% approximatedCorrection = tagOffset - calibrationSourceFrequency;

%Frequency Corrected
demodulatedSignal = demodulateSignal(complexSignal, time, frequencyOffset);
%Phase Corrected
demodulatedSignal = demodulatedSignal*exp(-1j*median(angle(demodulatedSignal)));

%Shifts tag frequency to zero
demodulatedSignal = demodulateSignal(demodulatedSignal, time, tagOffset);
%
% signalLength = length(demodulatedSignal);
% NFFT = 2^nextpow2(signalLength*32);
% frequencySpectrum = fft(demodulatedSignal, NFFT)/signalLength;
% dt = 1/sampleFrequency;
% T = dt*NFFT;
% df = 1/T;
% frequencyVector = 0:df:(NFFT-1)*df;
% figure; plot(frequencyVector, fftshift(abs(frequencySpectrum)));


%Filtering with Cut Off Frequency
filteredSignal = reducenoise(demodulatedSignal, cutOffFrequency, sampleFrequency);

end
```

Listing 7: Applies the frequency correction to the data to demodulate at the carrier frequency and corrects the phase.

```
function [frequencyOffset, nanotagFrequency] = getCalibrationSourceFrequencyOffset(signal,
    sampleFreq)
%UNTITLED10 Summary of this function goes here
%   Detailed explanation goes here

signalLength  = length(signal);
NFFT = 2^nextpow2(signalLength*32);
frequencySpectrum = fft(signal, NFFT)/signalLength;
% frequencyVector = linspace(0, 1, NFFT-1)*sampleFreq;

dt = 1/sampleFreq;
T = dt*NFFT;
df = 1/T;
frequencyVector = 0:df:(NFFT-1)*df;
frequencyVectorPlot = -((NFFT-1)*df)/2:df:((NFFT-1)*df)/2;


intensityVector = 1:length(frequencyVector);

[PeakFTs,Idx] = max(abs(frequencySpectrum(intensityVector))*2);
frequencyOffset = frequencyVector(Idx);

% [envHigh, ~] = envelope(abs(frequencySpectrum),1500,'peak');
% outsideBand = envHigh((length(envHigh)/2):end);
% [val, nanotagFrequencyIndex] = max(outsideBand);
% newFrequency  = frequencyVector((length(envHigh)/2):end);
% nanotagFrequency = newFrequency(nanotagFrequencyIndex);
% %
% figure(66); plot(frequencyVector/1000, abs(frequencySpectrum));
% hold on; plot(frequencyVector/1000, envHigh); hold off

lowerLimit = 38*length(frequencySpectrum)/44;
upperLimit = 40.2*length(frequencySpectrum)/44;
[envHigh, envLow] = envelope(abs(frequencySpectrum(lowerLimit:upperLimit)), 1500, 'peak');
skipSpike = 1000;
outsideBand = envHigh(skipSpike:end-skipSpike);
[val, nanotagFrequencyIndex] = max(outsideBand);
```

```
newFrequency  = frequencyVector(lowerLimit:upperLimit);
nanotagFrequency = newFrequency(nanotagFrequencyIndex+skipSpike-1);
%
figure(66); plot(newFrequency/1000, abs(frequencySpectrum(lowerLimit:upperLimit)));
hold on; plot(newFrequency/1000, envHigh); hold off


% figure
% plot(frequencyVectorPlot, fftshift(abs(frequencySpectrum(intensityVector))*2))
% grid; xlabel('Frequency [Hz]'); ylabel('Magnitude'); title('FFT of Ping Group 1 on Antenna
    2')
% text(frequencyVector(Idx)+370, PeakFTs, sprintf('Calibration Source Frequency: %.3f Hz',
    frequencyVector(Idx)), 'HorizontalAlignment','left', 'VerticalAlignment','top', 'FontSize
    ', 20);
% text(-22904, .005, sprintf('Nanotag Frequency: -5140.914 Hz'), 'HorizontalAlignment','left',
    'VerticalAlignment','top','FontSize', 20);
% set(gca, 'FontSize', 20)
end
```

Listing 8: Uses an FFT to get the frequency offsets of the dongles and corrects it.

```
function frequencyOffset = getFrequencyOffset(complexSignal, sampleRate)
%GETFREQUENCYOFFSET: finds the highest frequency in the spectrum. It is
%assumed to be the calibration signal.
%   Inputs:
%       complexSignal: recorded data broken into real and imaginary
%       components
%       sampleRate: sampling frequency
%   Outputs:
%       frequencyOffset: the frequency highest in the spectrum

lengthSignal = length(complexSignal);
if mod(lengthSignal,2)~=0
    complexSignal = complexSignal(1:lengthSignal-1);
end

spectrum = fftshift(fft(complexSignal));

%f = linspace(-FS/2, FS/2, N);
deltaFreq = sampleRate/lengthSignal;
frequency = -(lengthSignal/2-1)*deltaFreq:deltaFreq:lengthSignal/2*deltaFreq;

%plot(f,abs(Spec))

[~, indexMax] = max(abs(spectrum));

frequencyOffset = frequency(indexMax);

end
```

Listing 9: Finds the frequency offset only for plotting pruposes.

```
function demodulatedSignal = demodulateSignal(complexSignal, time, frequencyOffset)
```

```
%DEMODULATEDSIGNAL uses frequency offset to calculate the original pulse
%from the nanotags using the
%   Inputs:
%       complexSignal: recorded data broken into real and imaginary
%       components
%       time: the duration of the signal
%       frequencyOffset: the frequency from getFreqeucnyOffset based on the
%       calibration frequency and the dongle frequency drift
%   Outputs:
%       demodulatedSignal: original message signal

demodulatedSignal = complexSignal.*exp(-1j*2*pi*frequencyOffset*time);
end
```

Listing 10: Demodulates the signal.

```
function [peakValues, peakIndex, filteredSignalSectionBuffer, bufferStart] =
    checkPeakDetections(peakValues, peakIndex, sampleRate, widths, minimumPeakDistance,
    complexSignal, filteredSignalSection, indexPeakSearchStart, indexPeakSearchEnd,
    pulseDuration,  peakThreshold, pulseSampleIndex, cutoffFrequency, frequencyCalibration,
    timeSectionInterest, extraCorrection)
%CHECKPEAKDETECTIONS takes in the peak detections found form
%getCorrectedDetections and checks if they fit the criteria that
%constitutes the peaks as the programmed pulse rather than a spike of
%noise. It ouputs the updated peakValues and peakIndex if the qualities of
%the peaks did not fit that of the nanotags pulses, or it ouputs the
%original detections if they seem to be right.
%   Inputs:
%       peakValues: the magnitude value of the detected peaks
%       peakIndex: the index of the detected peaks
%       sampleRate: sampling frequency
%       minimumPeakDistance: minimum distance between pings within the
%       pulse
%       filteredSignal: the filtered signal used peak detection
%       indexPeakSearchStart: the starting index for where the pulse starts
%       indexPeakSearchEnd: the ending index for where the pulse ends
%       pulseDuration: maximum time of between the initial ping and the
%       last ping of a pulse
%       peakThreshold: the magnitude the peak must be above for detection
%       pulseSampleIndex: the index length between two pusles
%
%   Outputs:
%       peakValues: the magnitude value of the detected peaks
%       peakIndex: the index of the detected peaks



%Checks to see if there are too few peaks detected, if so it then lowers
%the threshold level by 0.003 and searches again.

% % % %Consider getting rid of this one.
% % % if length(peakValues)<4 && ~isempty(peakValues)
% % %     disp('Threshold adjusted.')
% % %     peakThreshold = peakThreshold -.003;
% % %      [peakValues, peakIndex, widths] = findpeaks(abs(filteredSignalSection), 'MinPeakHeight', peakThreshold, 'MinPeakDistance', minimumPeakDistance, 'MinPeakWidth', 30);
% % %
% % %      if length(peakValues)<4
% % %          peakValues = [];
% % %          peakIndex = [];
% % %      end
% % % end

%ADDED FOR DEBUGGING
if length(peakValues)<4
% %      disp('Threshold adjusted NEW CODE.')
    peakThresholdNew = peakThreshold -.0035;
    [peakValues, peakIndex, widths] = findpeaks(abs(filteredSignalSection), 'MinPeakHeight', peakThresholdNew, 'MinPeakDistance', minimumPeakDistance, 'MinPeakWidth', 30);

    if ~isempty(peakIndex)
        if pulseDuration<(peakIndex(end)-peakIndex(1))/sampleRate
            peakIndex = [];
            peakValues = [];
        end
    end

end

%Checks to see if too many peaks have been detected and attemps to filter
%out the false detections.

filteredSignalSectionBuffer = [];
bufferStart = [];
if length(peakValues)<4 && ~isempty(pulseSampleIndex)
    %Try sliding the window, this means that the peak threshold is too
    %high, this may not work.
    bufferStart = indexPeakSearchEnd-pulseSampleIndex;
    if bufferStart < 0
        bufferStart = 1;
    end
%     [filteredSignalSectionBuffer] = getFilteredSignal(complexSignal, sampleRate, frequencyCalibration, timeSectionInterest, cutoffFrequency, bufferStart, indexPeakSearchEnd);
    [filteredSignalSectionBuffer] = processData(complexSignal(bufferStart:indexPeakSearchEnd), timeSectionInterest(bufferStart:indexPeakSearchEnd), sampleRate, cutoffFrequency, extraCorrection);
    [peakValuesBuffer, peakIndexBuffer, widths] = findpeaks(abs(filteredSignalSectionBuffer), 'MinPeakHeight', peakThreshold, 'MinPeakDistance', minimumPeakDistance, 'MinPeakWidth', 23);

    if isempty(peakValuesBuffer)
        filteredSignalSectionBuffer = [];
    else
        peakIndex = peakIndexBuffer;
        peakValues = peakValuesBuffer;
    end
```

```

end


%Finds maximum peak, and uses the index as a reference index
if length(peakValues)>4
    %Finds the max peak, and assumes that it is true.
    %      [~, maxPeakIndex] = max(peakValues);
    %      peakMaxIndexTrue = peakIndex(maxPeakIndex);
    %      peakMatchIndexLower = peakMaxIndexTrue-maximumPeakDistanceIndex;
    %      peakMatchIndexHigher = peakMaxIndexTrue+maximumPeakDistanceIndex;
    %      peakMatchIndex = peakIndex>=peakMatchIndexLower & peakIndex<peakMatchIndexHigher;
% %      disp('Too many peak values.')
    %If loop to accomodate the ignoring of the initial spike in the
    %getInitialIdxBoundsForFreqDiff
    if isempty(pulseSampleIndex)
        peakIndex = peakIndex+floor(sampleRate/5);
    end

    %Widths
    [~, maxWidthIdx] = maxk(widths, 4);
    sortedWidthIdk = sort(maxWidthIdx, 'ascend');
    peakIndexUsingWidths = peakIndex(sortedWidthIdk);
    peakValuesUsingWidths= peakValues(sortedWidthIdk);

    %Diffs
    relativeDiffs = abs((peakIndex - peakIndex')./peakIndex;
    sumDiffRows = sum(relativeDiffs,2);
    [~, idxsortSumDiffRows] = sort(sumDiffRows);
    peakIndexUsingDiffs = peakIndex(idxsortSumDiffRows(1:4));
    peakValuesUsingDiffs = peakValues(idxsortSumDiffRows(1:4));
    [peakIndexUsingDiffs, indexSortDiffs] = sort(peakIndexUsingDiffs);
    peakValuesUsingDiffs = peakValuesUsingDiffs(indexSortDiffs);

    %Maxes
        %Finds highest peak values and then sorts them according to index
        [peakValuesSorted, indexMaxPeakSort] = sort(peakValues, 'descend');
        indexSortedByPeaks = peakIndex(indexMaxPeakSort);

        peakIndex4Max = indexSortedByPeaks(1:4);
        peakValues4Max = peakValuesSorted(1:4);
       [peakIndexUsingMax, indexSort] = sort(peakIndex4Max);
        peakValuesUsingMax = peakValues4Max(indexSort);

    if pulseDuration>(peakIndexUsingWidths(end)-peakIndexUsingWidths(1))/sampleRate ||
         pulseDuration>(peakIndexUsingDiffs(end)-peakIndexUsingDiffs(1))/sampleRate
        if ~isempty(filteredSignalSectionBuffer)
            figure(100); plot(abs(filteredSignalSectionBuffer)); xlim([0 length(
                 filteredSignalSectionBuffer)])
        else
            figure(100); plot(abs(filteredSignalSection)); xlim([0 length(filteredSignalSection
                 )])
        end
        hold on; plot(peakIndexUsingWidths, peakValuesUsingWidths, 'r*')
        plot(peakIndexUsingDiffs, peakValuesUsingDiffs, 'ko');
        plot(peakIndexUsingMax, peakValuesUsingMax, 'gs', 'MarkerSize', 9); hold off
        xlabel('Index'); ylabel('Power'); title('Check if the peaks found are right.')
        xlim([min([peakIndexUsingWidths; peakIndexUsingDiffs])-5000 max([peakIndexUsingWidths;
            peakIndexUsingDiffs]+5000)])
        legend('Signal', 'Widths', 'Differences', 'Max')
        disp('Set the peakIndex and peakValues by hand, check the plot.')
        keepIndex = input('1 for Widths, 2 for Difference, 3 for Max, or 4 for neither: ');
        if keepIndex == 1
            peakIndex = peakIndexUsingWidths;
            peakValues = peakValuesUsingWidths;
        elseif keepIndex == 2
            peakIndex = peakIndexUsingDiffs;
            peakValues = peakValuesUsingDiffs;
        elseif keepIndex == 3
            peakIndex = peakIndexUsingMax;
            peakValues = peakValuesUsingMax;
        else
            peakIndex = [];
            peakValues = [];
        end

    end


    %Just in case:
    if ~isempty(peakIndex)
        if pulseDuration<(peakIndex(end)-peakIndex(1))/sampleRate
            peakIndex = [];
            peakValues = [];
        end
    end



    %      if pulseDuration<(peakIndex4Max(1)-peakIndex4Max(end))/sampleRate
    %          figure(21); plot(abs(filteredSignalSection)); hold on; plot(peakIndex4Max-
        indexPeakSearchStart, peakValues4Max, 'o')
    %          disp('Wrong peak detected, either first or last peak, replaced last peak index
        with next highest peak value index')
    %          peakIndex4Max(4) = indexSortedByPeaks(5);
    %          peakValues4Max(4) = peakValuesSorted(5);
    %          figure(22); plot(abs(filteredSignalSection)); hold on; plot(peakIndex4Max-
        indexPeakSearchStart, peakValues4Max, 'o')
    %      end

    %      [peakIndex, indexSort] = sort(peakIndex4Max);
    %      peakValues = peakValues4Max(indexSort);
    %%
    %%      if length(peakValues)<4
    %%          disp(['Sketptical.']);
    %%          peakValues = [];
    %%          keyboard
    %%      else
    % %         %         disp('False detections were found and removed')
    % %     end
end

%LAST CHECK!
%Sometimes before it gets into this it only has one value found, and if so,
%it keeps it throughout the process, we want to discard it. Example: File
%1, time 198
if length(peakIndex)<4
    peakIndex = [];
    peakValues = [];
elseif length(peakIndex) == 4 && ~isempty(filteredSignalSectionBuffer)
    %Redines the filteredSignalBuffer to the same starting point as if it were
    %not changed so that unwrapping the phase has the same amount of samples
    %before the ping every time
    try
        filteredSignalSectionBuffer = filteredSignalSectionBuffer(peakIndex(1)-round(sampleRate/10):peakIndex(end)+round(sampleRate/10));

    catch
        filteredSignalSectionBuffer = filteredSignalSectionBuffer(peakIndex(1)-round(sampleRate/10):end);
    end
%     peakIndex = [round(sampleRate/10); (peakIndex(2:end)-peakIndex(1))+round(sampleRate/10)];

end


end
```

Listing 11: Uses a series of checks to ensure that the peaks are the NanoTag transmissions and not noise spikes.

```
function [peakValues, peakIndex, filteredSignalFull7Seconds] = manualPeakPickingCheck(filteredSignalFull7Seconds, sampleRate, peakThreshold, minimumPeakDistance, pulseDuration)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

[peakValues, peakIndex] = findpeaks(abs(filteredSignalFull7Seconds), 'MinPeakHeight', ...
    peakThreshold, 'MinPeakDistance', minimumPeakDistance, 'MinPeakWidth', 23);


fig100 = figure(100);
plot(abs(filteredSignalFull7Seconds));
hold on; plot(peakIndex, peakValues, 'o'); hold off
xlabel('Index'); ylabel('Power'); title('Use datatip the peaks from the tag')
legend('Signal', 'Peak Options'); axis auto

pickIndex = input(['1 for set the peakIndex and peakValues by hand as column vector, 2 for not existant',...
    '\nIf you choose 1, hold alt while using the datatip selector on Fig 100. ']);
if pickIndex == 1
    [peakIndex, peakValues] = pickPeaks(fig100);
    while length(peakIndex)~=4
        disp('Chose too many or few peaks, try again')
        clear peakIndex peakValues
        pickIndex = input(['1 for set the peakIndex and peakValues by hand as column vector, 2
            for not existant',...
    '\nIf you choose 1, hold alt while using the datatip selector on Fig 100. ']);
        [peakIndex, peakValues] = pickPeaks(fig100);
    end

    try
        filteredSignalFull7Seconds = filteredSignalFull7Seconds(peakIndex(1)-round(sampleRate
            /10):length(filteredSignalFull7Seconds)+round(sampleRate/10));
    catch
        %The pulses might be too close to the end of the data section,
        %therefore the size needed to be (index:end)
        filteredSignalFull7Seconds = filteredSignalFull7Seconds(peakIndex(1)-round(sampleRate
            /10):end);
    end

    %Redefines the filteredSignalBuffer to the same starting point as if it were
    %not changed so that unwrapping the phase has the same amount of samples
    %before the ping every time
%     peakIndex = [round(sampleRate/10); (peakIndex(2:end)-peakIndex(1))+round(sampleRate/10)];

    if pulseDuration<(peakIndex(end)-peakIndex(1))/sampleRate
        disp('Probably chose the wrong peaks, the sample index between the first and last ping
            are wrong')
        peakIndex = [];
        peakValues = [];
        filteredSignalFull7Seconds = [];
    end
else
    peakIndex = [];
    peakIndex = [];
    peakValues = [];
    filteredSignalFull7Seconds = [];
end


end
```

Listing 12: A way to check for peaks one last time if the algorithm could not find them.

```
function [] = makePlots(time, filteredSignal, phaseMeasurements, timeDetected, peakValues,
    phaseDetected, pulseNumber, fileNumber, numberOfDetections)
%UNTITLED3 Summary of this function goes here

if ~isempty(peakValues{pulseNumber, fileNumber}) & ~isnan(peakValues{pulseNumber, fileNumber})
xLimitStart = timeDetected{pulseNumber, fileNumber}(1) -0.01;
xLimitEnd = timeDetected{pulseNumber, fileNumber}(end)+0.01;

figure(fileNumber+1)
```

```
subplot(4,round(numberOfDetections/4), pulseNumber)
plot(time{pulseNumber, fileNumber}, abs(filteredSignal{pulseNumber, fileNumber}))
hold on
plot(timeDetected{pulseNumber, fileNumber}, peakValues{pulseNumber, fileNumber}, 'ko')
hold off
xlim([xLimitStart xLimitEnd])
xlabel('Time (s)')
ylabel('Magnitude')
title(['#', num2str(pulseNumber)])
%title(['Energy over time, Detection #', num2str(pulseNumber)], 'FontSize',14)


figure(fileNumber+4)
subplot(4,round(numberOfDetections/4), pulseNumber)
plot(time{pulseNumber, fileNumber}, phaseMeasurements{pulseNumber, fileNumber})
hold on
plot(timeDetected{pulseNumber, fileNumber}, phaseDetected{pulseNumber, fileNumber}, 'ko')
hold off
xlim([xLimitStart xLimitEnd])
xlabel('Time (s)')
ylabel('Phase (rad)')
title(['#', num2str(pulseNumber)])
%title(['Phase angle over time, Detection #', num2str(pulseNumber)], 'FontSize',14)



else
    disp('No peaks found, plot empty.')

    figure(fileNumber+1)
    subplot(4,round(numberOfDetections/4), pulseNumber)
    title(['#', num2str(pulseNumber)])

    figure(fileNumber+4)
    subplot(4,round(numberOfDetections/4), pulseNumber)
    title(['#', num2str(pulseNumber)])

end

end
```

Listing 13: Makes plots of the magnitudes and phases to check results.

```
function [indexPeakSearchStart, indexPeakSearchEnd, indexFreqDiffStart, indexFreqDiffEnd] =
    getPeakIndexSearchLimits(peakIndex, pulseNumber, pulseSampleIndex, indexPeakSearchStart,
    indexPeakSearchEnd, indexFreqDiffStart, indexFreqDiffEnd, sampleRate)
%GETPEAKINDEXSEARCHLIMITS Summary of this function goes here
%   Inputs:
%
%   Ouputs:
%       indexPeakSearchStart: the starting index for where the pulse starts
%       indexPeakSearchEnd: the ending index for where the pulse ends
%       indexFreqDiffStart: the starting index for calculating the
%       frequency offset
%       indexFreqDiffEnd: the ending index for calculating the
%       frequency offset



if ~isnan(peakIndex{pulseNumber})
    peakIndex = peakIndex{pulseNumber};
    factor = 1;
else
    peakIndex = peakIndex(~cellfun('isempty',peakIndex));
    matPeakIndex = cell2mat(peakIndex);
    firstDetectionIndex = matPeakIndex(1:4:length(matPeakIndex));
    pulseNumberLastDetectedLogicial = ~isnan(firstDetectionIndex);
    pulseNumberLastDetected =  find(pulseNumberLastDetectedLogicial==1,1,'last');
    factor = length(firstDetectionIndex)-pulseNumberLastDetected+1;
    peakIndex = peakIndex{pulseNumberLastDetected};
end

pulseSampleIndex = pulseSampleIndex*factor;
indexFirstPing = peakIndex(1)+pulseSampleIndex;
indexLastPing = peakIndex(end)+pulseSampleIndex;

indexSearchBuffer = round(sampleRate/10);
indexPeakSearchStart = indexFirstPing-indexSearchBuffer;
indexPeakSearchEnd = indexLastPing+indexSearchBuffer;

%Finds one second before and one second after
indexFreqDiffStart = indexFirstPing+pulseSampleIndex-sampleRate;
indexFreqDiffEnd = indexFirstPing+pulseSampleIndex+sampleRate-1;


% %     indexPeakSearchStart = indexPeakSearchStart+pulseSampleIndex;
% %     indexPeakSearchEnd = indexPeakSearchEnd+pulseSampleIndex;
% %     indexFreqDiffStart = indexFreqDiffStart+pulseSampleIndex;
% %     indexFreqDiffEnd = indexFreqDiffEnd+pulseSampleIndex;

% if isnan(peakValues)
%     %Update the search index bounds
%     indexPeakSearchStart = indexPeakSearchStart+pulseSampleIndex;
%     indexPeakSearchEnd= indexPeakSearchEnd+pulseSampleIndex;
%     indexFreqDiffStart= indexFreqDiffStart+pulseSampleIndex;
%     indexFreqDiffEnd= indexFreqDiffEnd+pulseSampleIndex;
% else
%     indexPeakSearchStart = peakIndex(end)+pulseSampleIndex;
%     indexPeakSearchEnd= peakIndex(end)+pulseSampleIndex;
%     indexFreqDiffStart= indexFreqDiffStart+pulseSampleIndex;
%     indexFreqDiffEnd= indexFreqDiffEnd+pulseSampleIndex;
% end

end
```

Listing 14: Finds the indices to limits for windowing to find the NanoTag transmission.

```
function [peakValues, peakIndex, phaseDetected, timeDetected] = addMissedDectectionsAsNaNs(
    pulseDetectedNumber, peakValues, peakIndex, phaseDetected, timeDetected)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

peakValues = cell2mat([num2cell(nan(4,pulseDetectedNumber), 1)'; peakValues]);
peakIndex = cell2mat([num2cell(nan(4,pulseDetectedNumber), 1)'; peakIndex]);

phaseDetected = cell2mat([num2cell(nan(4,pulseDetectedNumber), 1)'; phaseDetected]);
timeDetected = cell2mat([num2cell(nan(4,pulseDetectedNumber), 1)'; timeDetected]);

end
```

Listing 15: Adds missed detections as nans so the phase values don't alter the differences.

```
function [bearing] = getDOAFrom2Antennas(phase2, phase3)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
wavelength = 1.803101334294987;
distance = wavelength/2;


phaseDifference = phase2-phase3;
for i = 1:length(phaseDifference)
    if phaseDifference(i)>pi
        phaseDifference(i) = phaseDifference(i)-(2*pi);
    elseif phaseDifference(i)<-pi
         phaseDifference(i) = phaseDifference(i)+(2*pi);
    end
end
pathDifference = phaseDifference/(2*pi)*wavelength;

bearing = 90 - asind(pathDifference/distance); %if positive 1st quadrant, negative 2nd quadrant



end
```

Listing 16: Find DOA with two antennas.

```
function [directionOfArrival] = getDirectionOfArrival(wavelength, antennaDistanceMatrix,
    phaseDifferences)
%GETDIRECTIONOFARRIVAL Summary of this function goes here
%   Inputs:
%       wavelength: speed of light/frequency [m]
%       positionMatrix: matrix of the distances from local xyz origin fo
%       antennas
%       phaseDifferences: matrix of 4x3 of phase differences between antennas
%       2-1, 3-1, and 3-2.
%
%   Outputs:
%       directionOfArrival: Angle from North (Antenna 1) towards target
%        (Nanotag)



% [~, col] = find(isnan(phaseDifferences))
% replaceColumnNumbers = unique(col)
% phaseDifferences(:, replaceColumnNumbers)= []


%%%%%%%%
directionCosines = pinv(antennaDistanceMatrix)*phaseDifferences*wavelength/(2*pi); %pinv is
    pseudoinverse (creates inverse of G) %Y THEN X
for idx = 1:length(directionCosines)
    normalizedCosines(:,idx) = norm(directionCosines(:,idx));
end

directionCosinesMagnitude = directionCosines./normalizedCosines;


directionOfArrivalRadians = atan(directionCosinesMagnitude(2,:)./directionCosinesMagnitude(1,:)
    );%uses location of calculated u and v the new point relative to the origin and calculates
     the angle from North tan(theta)=Opposite/Adjacent

% directionOfArrivalRadians = pi + atan(directionCosinesMagnitude(2,:)./
    directionCosinesMagnitude(1,:));%uses location of calculated u and v the new point
    relative to the origin and calculates the angle from North tan(theta)=Opposite/Adjacent

% greaterThanPi = directionOfArrivalRadians>pi;
% lessThanPi = directionOfArrivalRadians<-pi;
% directionOfArrivalRadians(greaterThanPi) = directionOfArrivalRadians(greaterThanPi)-(2*pi);
% directionOfArrivalRadians(lessThanPi) = directionOfArrivalRadians(lessThanPi)+(2*pi);

directionOfArrival = rad2deg(directionOfArrivalRadians);




%%%%%%%%
% directionOfArrival = [directionOfArrival(1:replaceColumnNumbers(1)-1), NaN, NaN, NaN, NaN,
    directionOfArrival(replaceColumnNumbers(4)+1:replaceColumnNumbers(5)), NaN, NaN, NaN, NaN,
    directionOfArrival(replaceColumnNumbers(end)+1:end)];
end
```

Listing 17: Find DOA with three antennas.

Appendix E: New Design Materials

Number	Items	Unit Price	Quantity	Cost
1	RSPduo Dual Tuner 14-bit SDR	$279.95	3	$839.85
2	Low Jitter Precision GPSDO Reference Oscillator	$185.44	1	$185.44
3	Raspberry Pi 4 4GB	$35.00	3	$105.00
4	Argon ONE Pi 4 Raspberry Pi Case	$25.00	3	$75.00

Total: $1,205.29

Number	Site
1	https://www.sdrplay.com/rspduo/
2	http://www.leobodnar.com/shop/index.php?main_page=product_info&products_id=234
3	https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
4	https://www.argon40.com/catalog/product/view/id/52/s/argon-one-raspberry-pi-4-case/

"Precision gps reference clock." [Online]. Available: http://www.leobodnar.com/shop/index.php?main_page=product_info&products_id=234

"Specifications." [Online]. Available: http://www.funcubedongle.com/?page_id=1201

2020. [Online]. Available: https://www.marine.com/products/11-10612/shakespeare-476-21-vhf-antenna

"Raimondo sets goal for 100[Online]. Available: https://www.ri.gov/press/view/37527

Barthelmie, R. J., "A brief review of offshore wind energy activity in the 1990's," *Wind Engineering*, pp. 265–273, 1998.

Beason, R. C., Nohara, T. J., and Weber, P., "Beware the boojum: caveats and strengths of avian radar," *Human–Wildlife Interactions*, vol. 7, no. 1, p. 4, 2013.

Bridge, E. S., Thorup, K., Bowlin, M. S., Chilson, P. B., Diehl, R. H., Fléron, R. W., Hartl, P., Kays, R., Kelly, J. F., Robinson, W. D., *et al.*, "Technology on the move: recent and forthcoming innovations for tracking migratory birds," *BioScience*, vol. 61, no. 9, pp. 689–698, 2011.

Burger, J., Gordon, C., Lawrence, J., Newman, J., Forcey, G., and Vlietstra, L., "Risk evaluation for federally listed (roseate tern, piping plover) or candidate (red knot) bird species in offshore waters: A first step for managing the potential impacts of wind facility development on the atlantic outer continental shelf," *Renewable Energy*, vol. 36, no. 1, pp. 338–351, 2011.

David, S.-M., Meca-Meca, F. J., Martín-Gorostiza, E., and Lázaro-Galilea, J. L., "Snr degradation in undersampled phase measurement systems," *Sensor*, vol. 1, no. 10, p. 1772, 2016.

Desholm, M., Fox, A., Beasley, P., and Kahlert, J., "Remote techniques for counting and estimating the number of bird–wind turbine collisions at sea: a review," *Ibis*, vol. 148, pp. 76–89, 2006.

Desrochers, A., Tremblay, J. A., Aubry, Y., Chabot, D., Pace, P., and Bird, D. M., "Estimating wildlife tag location errors from a vhf receiver mounted on a drone," *Drones*, vol. 2, no. 4, p. 44, 2018.

RF Explorer Signal Generator, Digi-Key Electronics, 2019, updated to Firmware Version 1.31. [Online]. Available: http://j3.rf-explorer.com/40-rfe/article/132-rf-explorer-signal-generator-rfe6gen-specification

Esteban, M. D., Diez, J. J., López, J. S., and Negro, V., “Why offshore wind energy?” *Renewable Energy*, vol. 36, no. 2, pp. 444–450, 2011.

Fox, A., Desholm, M., Kahlert, J., Christensen, T. K., and Krag Petersen, I., “Information needs to support environmental impact assessment of the effects of european marine offshore wind farms on birds,” *Ibis*, vol. 148, pp. 129–144, 2006.

Furness, R. W., Wade, H. M., and Masden, E. A., “Assessing vulnerability of marine bird populations to offshore wind farms,” *Journal of environmental management*, vol. 119, pp. 56–66, 2013.

Halfacree, G., “Group test: Best raspberry pi 4 thermal cases tested and ranked,” Feb 2020. [Online]. Available: https://magpi.raspberrypi.org/articles/group-test-best-raspberry-pi-4-thermal-cases-tested-and-ranked

Hallworth, M. T. and Marra, P. P., “Miniaturized gps tags identify non-breeding territories of a small breeding migratory songbird,” *Scientific reports*, vol. 5, p. 11069, 2015.

Hamadi, V., Brosnan, U., Loftus, I., and Montgomery, G., “Offshore substation design: High-level overview of the industry best practices,” *IEEE Power and Energy Magazine*, vol. 17, no. 4, pp. 67–74, 2019.

Hüppop, O., Dierschke, J., EXO, K.-M., Fredrich, E., and Hill, R., “Bird migration studies and potential collision risk with offshore wind turbines,” *Ibis*, vol. 148, pp. 90–109, 2006.

Jenkins, H. H., *Small-Aperture Radio Direction-Finding.* Massachusetts, United States of America: Artech House Inc., 1991.

Kolz, A. L., “Radio frequency assignments for wildlife telemetry: A review of the regulations,” *Wildlife Society Bulletin (1973-2006)*, vol. 11, no. 1, pp. 56–59, 1983.

López-López, P., “Individual-based tracking systems in ornithology: welcome to the era of big data,” *Ardeola*, vol. 63, no. 1, pp. 103–136, 2016.

Loring, P., “Evaluating digital vhf technology to monitor shorebird and seabird use of offshore wind energy areas in the western north atlantic,” Ph.D. dissertation, UMass Amherst, 2020.

Loring, P., Paton, P., McLaren, J., Bai, H., Janaswamy, R., Goyert, H., and Sievert, P., "Tracking offshore occurrence of common terns, endangered roseate terns, and threatened piping plovers with vhf arrays," *Sterling (VA): US Department of the Interior, Bureau of Ocean Energy Management. OCS Study BOEM*, vol. 17, p. 140, 2019.

NanoTags (Coded VHF), Lotek, 2020, rev. 1.2. [Online]. Available: https://www.lotek.com/wp-content/uploads/2017/10/NanoTags-Spec-Sheet.pdf

MacCurdy, R. B., Gabrielson, R. M., and Cortopassi, K. A., "Automated wildlife radio tracking," *SA Zekavat & RM Buehler (red). Handbook of position location: theory, practice, and advances. Wiley, Hoboken, NJ*, 2011.

Marques, A. T., Batalha, H., Rodrigues, S., Costa, H., Pereira, M. J. R., Fonseca, C., Mascarenhas, M., and Bernardino, J., "Understanding bird collisions at wind farms: An updated review on the causes and possible mitigation strategies," *Biological Conservation*, vol. 179, pp. 40–52, 2014.

Nickolas, C. Digi-Key Electronics. "The basics of mixers." [Online; accessed Apr 4, 2020]. 2011. [Online]. Available: https://www.digikey.com/en/articles/the-basics-of-mixers

Proakis, John G, M. D. K., *Digital Signal Processsing (4th Edition)*. Prentice-Hall, Inc., USA, 2006.

RSPduo Dual Tuner 14-bit SDR, SDRplay, 05 2018, v0.6.

Severson, J. P., Coates, P. S., Prochazka, B. G., Ricca, M. A., Casazza, M. L., and Delehanty, D. J., "Global positioning system tracking devices can decrease greater sage-grouse survival," *The Condor*, vol. 121, no. 3, p. duz032, 2019.

Shouman, E. R. M., "Global prediction of wind energy market strategy for electricity generation," in *Modeling, Simulation and Optimization of Wind Farms and Hybrid Systems*. IntechOpen, 2020.

Staffell, I. and Green, R., "How does wind farm performance decline with age?" *Renewable energy*, vol. 66, pp. 775–786, 2014.

Systems, S. R., "Rf signal generators: Sg380 series," 2018, [Online; accessed Nov 8, 2018]. [Online]. Available: https://www.thinksrs.com/products/sg380.html

Taylor, P., Crewe, T., Mackenzie, S., Lepage, D., Aubry, Y., Crysler, Z., Finney, G., Francis, C., Guglielmo, C., Hamilton, D., *et al.*, "The motus wildlife tracking system: a collaborative research network to enhance the understanding of wildlife movement," *Avian Conservation and Ecology*, vol. 12, no. 1, 2017.

Taylor, P., Crewe, T., Mackenzie, S., Lepage, D., Aubry, Y., Crysler, Z., Finney, G., Francis, C., Guglielmo, C., Hamilton, D., *et al.*, “The motus wildlife tracking system: a collaborative research network to enhance the understanding of wildlife movement,” *Avian Conservation and Ecology*, vol. 12, no. 1, 2017.

Time and Date, “Past weather in narragansett, rhode island, usa — august 2019,” cited 2020, [Available online at ”https://www.timeanddate.com/weather/usa/narragansett/historic?month=8&year=2019”].

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

Wickert, M. A., “Lab 6: Software defined radio and the rtl-sdr usb dongle,” Lab Practical ECE4670, 2000. [Online]. Available: http://www.eas.uccs.edu/~mwickert/ece4670/lecture_notes/Lab6.pdf

Xue, Q. and Liao, S., “High frequency and high gain two-element collinear antenna array,” in *2015 International Symposium on Antennas and Propagation (ISAP)*. IEEE, 2015, pp. 1–3.