

6-12-2018

RF Roaming System Locator: A Modular Omnidirectional Antenna System

George Stathakis

Santa Clara University, gstathakis@scu.edu

Josh Sullivan

Santa Clara University, jnsullivan@scu.edu

Christian Ayscue

Santa Clara University, cayscue@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/idp_senior



Part of the [Computer Engineering Commons](#), and the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Stathakis, George; Sullivan, Josh; and Ayscue, Christian, "RF Roaming System Locator: A Modular Omnidirectional Antenna System" (2018). *Interdisciplinary Design Senior Theses*. 43.

https://scholarcommons.scu.edu/idp_senior/43

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Interdisciplinary Design Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

DEPARTMENT of COMPUTER ENGINEERING

Date: June 12, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY
SUPERVISION BY

Christian Ayscue

ENTITLED

RF Roaming System Locator

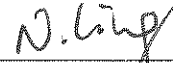
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



THESIS ADVISOR



DEPARTMENT CHAIR

SANTA CLARA UNIVERSITY

Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

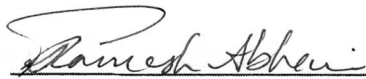
George Stathakis, Josh Sullivan

ENTITLED

RF Roaming System Locator

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

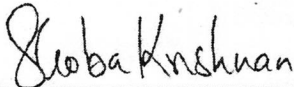
**BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING**



Thesis Advisor Signature

6/13/2018

date



Department Chair Signature

6/12/2018

date

RF Roaming System Locator:

A Modular Omnidirectional

Antenna System



By

George Stathakis, Josh Sullivan, Christian Ayscue

Abstract

Drones are increasingly common in everything from entertainment to delivery services. Currently, GPS and wireless triangulation are the two main methods of tracking drones and collision avoidance. The main problem with these two methods are accuracy and mobility. This thesis goes into building a system that is able to solve the two main drawbacks, by using a completely contained antenna system. The system is robust and can work in a variety of environments and situations. While the thesis focuses on drone tracking, the system is built with modularity in mind and can be adapted to track other RF signals with proper RF down conversion modules. This thesis will go in depth in how this aforementioned system was conceptualized, designed, built, and evaluated.

Table of Contents

Chapter 1: Introduction.....	5
1.1. Problem Objectives.....	6
1.2. Project Requirements.....	7
Chapter 2: Background and Problems with Current Technology.....	8
2.1 Current Innovations and Research.....	12
Chapter 3: Senior Design Project	14
3.1 Design Process.....	14
3.2 Components Used In All Iterations.....	15
3.2.1 Spiral Antenna.....	15
3.2.2 GUI.....	16
3.3 Iteration 1.....	17
3.3.1 RF A to D Converter.....	17
3.4 Iteration 2.....	18
3.4.1 Mixer.....	19
3.4.2 ADC.....	20
3.5 Final Design.....	21
3.5.1 Power Detector.....	22
3.6 Comparison.....	23
Chapter 4: Software Implementation.....	24
4.1 Use Cases.....	24
4.2 Training the Location Model.....	25
4.3 Machine Learning Algorithm.....	26
4.3.1 K-Nearest-Neighbors Algorithm.....	26
4.3.2 Neural Network.....	27
4.4 Pipeline Architecture for Deployment.....	28
4.5 Smoothing Algorithm.....	29
4.6 Software Packages.....	30

Chapter 5: Test Plan and Results.....	31
5.1 Spiral Antenna: 915 MHz.....	31
5.2 Power Detector: ZX47-40-S+.....	32
5.3 ADC: MCP 3008.....	33
5.4 GUI	33
Chapter 6: Final Project Analysis.....	34
6.1 Objectives Met.....	34
6.2 Gantt Chart.....	35
6.2.1 Proposed.....	35
6.2.2 Actual.....	35
6.3 Future Work.....	36
Chapter 7: Professional Issues and Constraints.....	36
7.1 Ethics.....	36
7.2 Science Technology and Society & Usability.....	37
7.2.1 Social Sustainability Benefits for User.....	37
7.3 Civic Engagement.....	37
7.4 Political Impact.....	38
7.5 Economic Impact.....	38
7.6 Health and Safety.....	39
7.7 Manufacturability.....	39
7.8 Usability.....	39
7.9 Environmental Sustainability.....	40
7.9.1 Materials Used.....	40
7.10 Lifelong Learning.....	41
7.11 Compassion.....	41
Chapter 8: Conclusion.....	42
Bibliography.....	43
Picture References.....	44

Appendices..... 46

 Appendix A: Machine Learning Algorithm..... 46

 Appendix B: Location Tracking App and GUI..... 49

 Appendix C: Data Collection Script..... 56

 Appendix D: Table of Components and Cost..... 58

1. Introduction

We live in a world that is full of wireless electronic devices and with the explosive growth in IoT and autonomous vehicles, the need for accurate RF signal tracking becomes more imperative. Currently, the drone industry is in the most immediate need for a system that can accurately track the drone's location. An article in the Rutgers Computer & Technology Law Journal specifically focuses on Amazon, and its need for delivery drone [1].



Figure 1: conceptual design of a delivery drone[12]

1.1 Project objectives

This project is meant to address this issue and make it so objects can be more accurately tracked. GPS currently is very limited on accuracy since it can be precise down to several meters. For most of the applications listed above accuracy of inches or less is needed. Our project is a continuation of a project from 2017 and the goal of this year's phase is to improve the accuracy and move the detection range to 360 degrees. Our plan is to accurately track a beacon in all 3 sectors with great precision while also keeping the system small and compact.

With this project, one of the initial goals was to allow the system to remain modular and portable. This would allow it to outshine competitive products such as WiFi based systems that require big triangulation areas. This will all be accomplished by using an RF based location tracking system that will allow greater accuracy and precision and a smaller footprint than other current tracking systems.

In the current design, the RF system will have an operating frequency of 915 MHz. This signal will be received by 3 different sectors split by 3 different spiral antennas. This will allow us to obtain the power and determine the position based on the amplitude.

This position will be displayed on a GUI, with the interface design being based off of figure 2. The project currently detects object at the 915 MHz range but only in a 120 degree window. The goal of our senior design is to first bring the project to 360 degrees by using 3 antennas, then to convert the individual analog signals to digital, and finally to use machine learning to estimate and display the location. Currently this comparison is done using hardware which will be too complicated. By accomplishing these things we will greatly improve and build upon the current project.

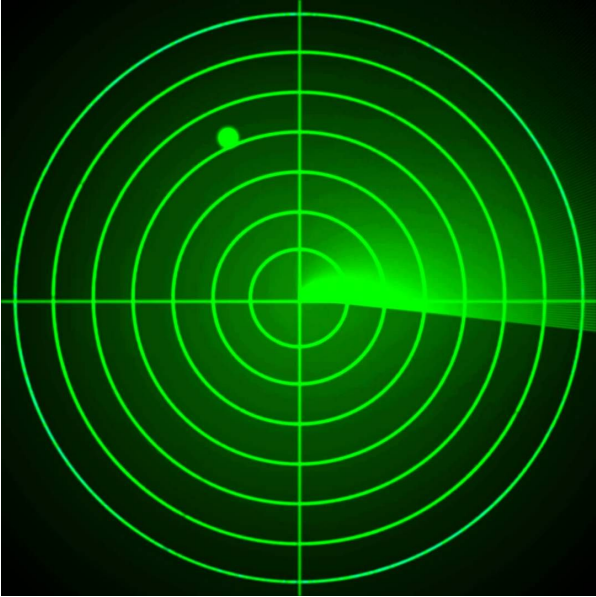


Figure 2: initial GUI concept[13]

1.2 Project requirements – marketing and engineering

- Accurate
- Lightweight
- Cost effective
- Small
- Rugged
- Compact
- Fast
- Professional

We need our project to be accurate so that it is a viable solution to tracking a drone. If it is not accurate enough then there is no reason for customers to buy our product over GPS or similar products. It needs to be lightweight and rugged so that it can operate in a variety of environments. This is a very important consideration for search and rescue



Figure 3: SAR drone prepping for delivery of first aid [14]

(SAR) operations. Most SAR will be conducted in disaster zones that are not conducive to fragile parts. Figure 3 shows one such SAR drone with delivery of first aid in mind. For our product to be competitive in this market, it must be cost effective. If our product is better than the leading competitor, but costs ten times more, less customers will be inclined to buy it.

Being small and compact are also major considerations. The primary purpose behind these considerations is so that it can be setup and relocated easily to fit the needs of the operator. Fast refers to a couple of things. We need it to be fast in regard to real time usage. Our GUI must be able to update quick enough so that the operator can see the drones position in real time. It also needs to be able to have a rapid setup to maximize SAR time. Our product needs to be professional so that customers want to buy it. Aesthetics is a major consideration in any product and a professional demeanor will help to make our product more appealing to our audience.

2 Background and Problems with Current Technology

The location tracking industry is a big business. It is deeply rooted within our society and it plays a major role in our economy. Today, GPS makes up most of the tracking services and is integrated into almost every hand held device. From cell phones to airplanes, GPS is the most commonly used solution. Most companies use GPS to integrate location services into their products.

GPS started in 1973 by the United States Government. Called the Navstar GPS system, several prototype Navstar satellites were launched beginning in 1978, and by the end of 1993 a complete set of 24 operational

satellites were in place. A picture of the basics of this system is shown in figure 4 right. Full operational capability was achieved by mid-1995. By late 2009 there were 35 satellites in orbit in the GPS “constellation,” including several extra satellites as a reserve[6]. A GPS satellite is basically an extremely accurate clock in orbit. It broadcasts coded signals representing time. A receiver determines its distance from a satellite by measuring the lapsed time

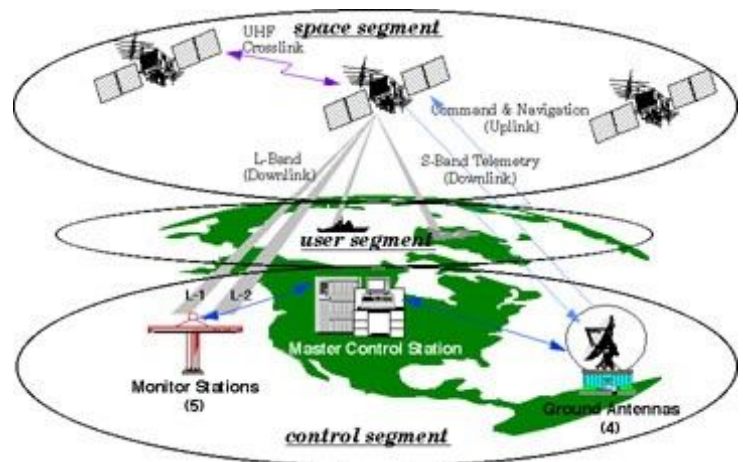


Figure 4:basic layout of Navstar GPS[15]

between transmission and reception. The receiver then takes similar measurements relative to at least three other satellites and uses this family of measurements to calculate its latitude, longitude, altitude, and often other parameters, such as velocity, direction, and orientation, as well [6].

Authorized users with receivers capable of reading the encrypted Pulse-per-second (PPS) code can typically determine their horizontal position with an accuracy of 5 to 10 m (roughly 15 to 33 ft). PPS signals are transmitted in two frequency bands; their comparison makes it possible to correct and compensate for the effects of ionospheric distortion. Civilian users, who use single-frequency standard positioning services (SPS), are guaranteed the capability to resolve their position within 36 m (120 ft) at least 95 percent of the time; in actuality, resolution as good as 10 to 20 m (33 to 65 ft) is commonly obtained. Formerly, the GPS signal provided to civilians was intentionally degraded, resulting in an accuracy of only 100 m (330 ft); that feature, known as selective availability, was discontinued in 2000[6].

According to GPS.gov, the official government website about GPS, the GPS for cellphone users is at best accurate within 16ft [3]. New technology such as PPS or dual frequency GPS receivers can improve accuracy greatly but require more infrastructure.[3] Table 1 below, shows a comparison between single and dual frequency precise point positioning. To add to the lack of resolution, GPS does not work inside buildings or where objects obstruct the signal. These problems inhibit the usefulness of GPS in modern technology. To solve these limitations the newest and most common solution is the use of WiFi triangulation through access points.

	Latency	Updates	Sample interval	Orbit RMS	Clock RMS	St.Dev.
Broadcast	Real-time	--	daily	~100 cm	~5 ns	~75 cm
Ultra-Rapid (predicted half)	Real-time	03, 09, 15, 21 UTC	15 min	~5 cm	~3 ns	~45 cm
Ultra-Rapid (observed half)	3 - 9 hours	03, 09, 15, 21 UTC	15 min	~3 cm	~150 ps	~1.5 cm
Rapid	17 - 41 hours	17 UTC daily	15 min / 5 min	~2.5 cm	~75 ps	~7.5 mm
Final	12 - 18 days	every Thursday	15 min / 30s	~2.5 cm	~75 ps	~6 mm

TABLE 1. Accuracy and latency of IGS products <<http://igs.cb.jpl.nasa.gov/components/prods.html>>, retrieved June 26, 2012. Orbit accuracies are 1D mean RMS values over the three XYZ geocentric components; clock accuracies are expressed relative to the IGS timescale. The standard deviation (St.Dev.) values in the last column are clock standard deviation values converted into range units. These are computed by removing a separate bias for each satellite and station clock, whereas this is not done for the clock RMS values. The St.Dev. values are a more realistic representation of the ephemerides errors in PPP (which solve for the phase ambiguities) than the individual orbit and clock RMS. The first row, with the broadcast ephemerides, is not an IGS product and is only included for comparison.

Table 1 [26]

However, there is a major drawback to using standard triangulation. One must set up specialized equipment such, as a triangulation tower or smart antenna array, in a fixed

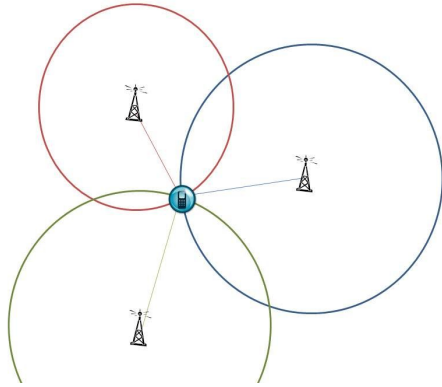


Figure 5: location triangulation utilizing towers [16]

This allows the system to perform angle estimates within the line of sight of the object, which is not always entirely possible [7]. Wifi triangulation is also very infrastructure heavy and static. This is what led us to RF location based tracking. This type of tracking was originally used for pinpoint object location in an indoor setting. There was originally a heavy military focus in location tracking, like the RTLS which was developed by NASA [4]. This system's main purpose was track lunar rovers, robots, and astronauts [4]. This was because there are no GPS satellites around the moon, so a less infrastructure dependant system was needed. This system is still in use today and was most recently used in the Mars missions [4]. This was one of the main driving factors behind the RF location tracking technology we see today, and a main source of inspiration for our project.

Microsoft's RADAR project was one of the first big initial pushes to use RF based location tracking for consumer use. The team that initially started the project used a very similar process that is now being used today in RF location tracking technology.

One RADAR team member expands on how the system works:

“The RADAR system works using a radio map. A radio map is a lookup table that holds collections of packet signal strengths and the building locations where these signals were measured. To locate the user's position, the user's wireless device measures the signal strength from the APs within its range and then searches the radio map to determine the signal strength entry that best matches the measured signal strength. To improve its estimate, RADAR takes into account the recent movement history of the user and dynamic changes such as temperature, the number of people present, and any other environmental factors which will affect the radio map [8].”

This project highlights how RF location tracking has been implemented in a practical situation. The latest implementation of this RF tracking technology is now being used for tracking the location of objects within buildings and in areas where GPS fails. Figure 6 below, shows how different materials affect signal strength. Even in Microsoft's RADAR project, the team was able to track the users in a different room or in other scenarios in which a typical GPS or Wifi system would fall short. Current company innovation is now trying to push towards a more precise, less infrastructure RF base location tracking system. This is where our project takes over and hopes to fill the current void.

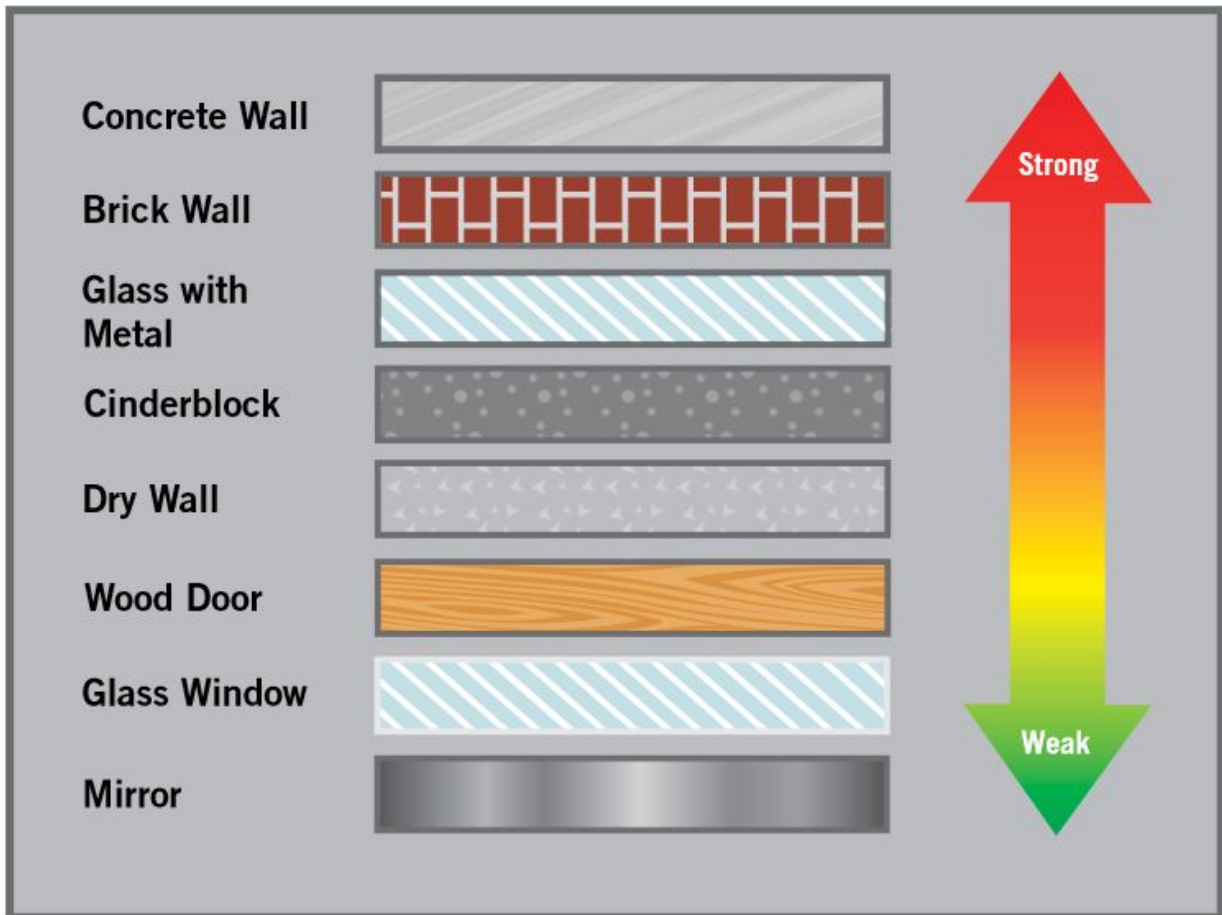


Figure 6: signal absorption rate of different materials [17]

2.1 Current innovations and research

Amazon is the current leader in drone distribution and implementation. According to their latest patent filing, Amazon is planning several different ways of creating drone

“hubs” where the drone can recharge and pick up their next package to deliver [2]. This latest patent details several different mobile ways a drone Hub can be constructed. Each drone Hub is created from a steel shipping container [2]. The patent filed picture to the left, figure 7, shows the design for a train centered mobile platform [2]. This patent goes into great detail about how the shipping container would be converted to serve the drone center efficiently

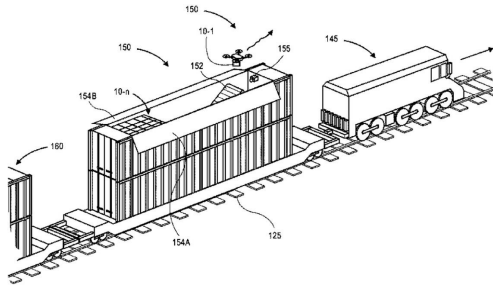


Figure 7: train centered drone hub and how it would be implemented on a train.

The image to right, figure 8, shows another way Amazon could create a drone mobile platform [2]. In figure 8 a cargo ship would be used to carry each shipping container to its destination. Both the train and cargo ship would be huge technological achievements and change how drone are distributed and tracked.

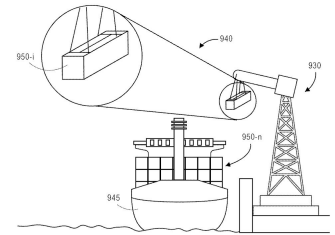


FIG. 9A

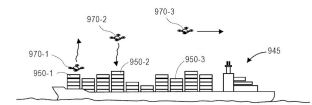


FIG. 9B

Figure 8: cargo ship centered drone hub

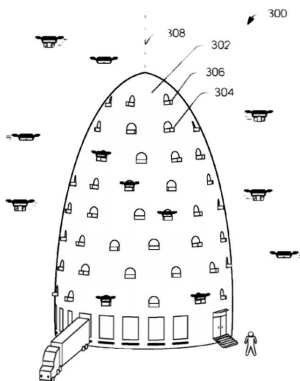


Figure 9: building centered drone hub

Amazon's latest patent is looking to create these Beehive like structures that can function as drone distribution centers, shown in figure 9 to the left.[9] All three hubs (train, cargo ship, and beehive) require very precise tracking capabilities for operation [9].

A start up company called Dedrone is a current market leader in RF location tracking. They actually use the same basic principles that are used in our project. A simple

representation of the company model is shown below in figure 10. “Dedrone RF Sensors detect drones and remote controls by their radio frequency (RF) signatures. Commercial, hobbyist, and homebrew drones are detected, including the entire DJI product line” [10].

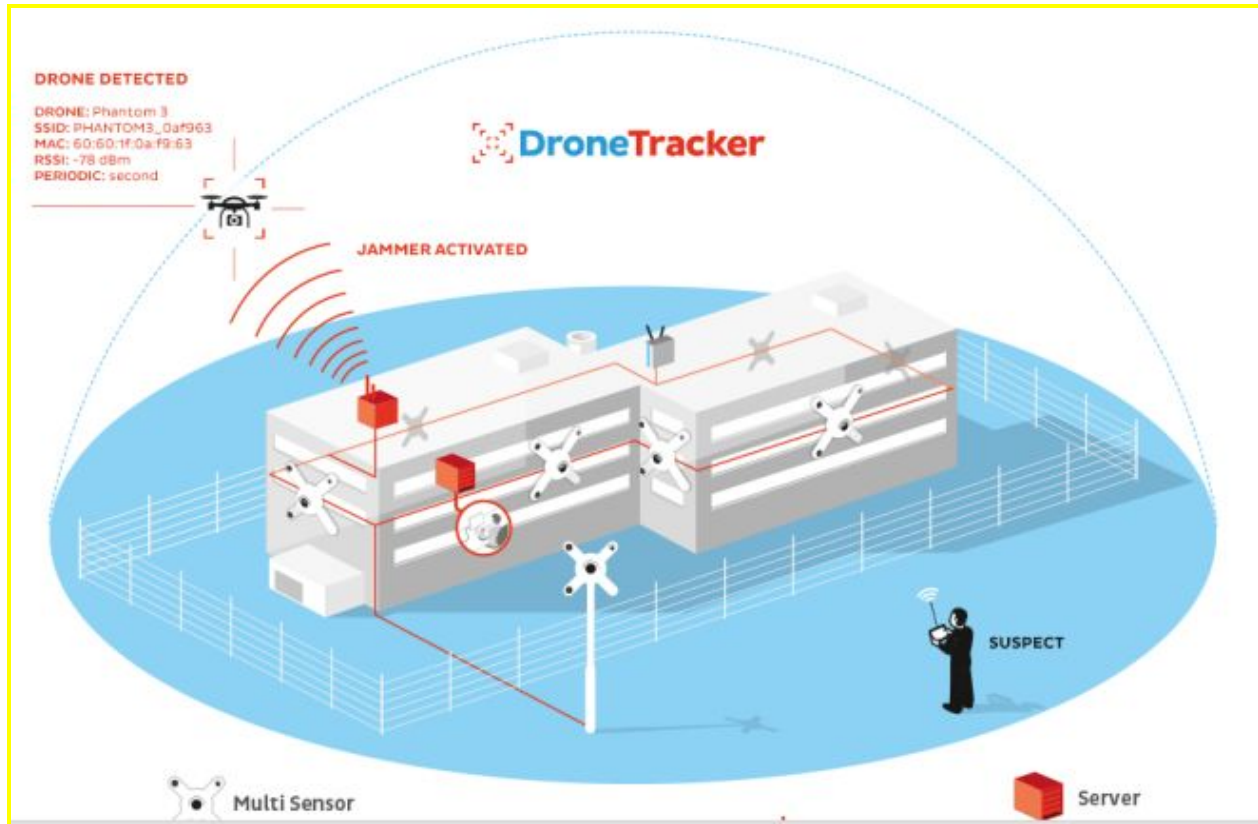


Figure 10: Dedrone company model [18]

3 Senior Design Project

3.1 Design Process

The block diagram below, figure 11, gives a general overview of our design process. We started by researching into what the previous year's team accomplished and looked at what we wanted to improve on/change in their system. Then we went into our first iteration which consists of just using an RF A to D converter. Through various problems discussed later on in this paper, we decided to forgo this iteration and moved into our second iteration which involved using a regular A to D converter and a mixer. We decided that we could optimize this iteration and moved into our final design which consisted of a power detector and a regular A to D converter. Finally, we tested our final design and compiled a list of results for our findings.

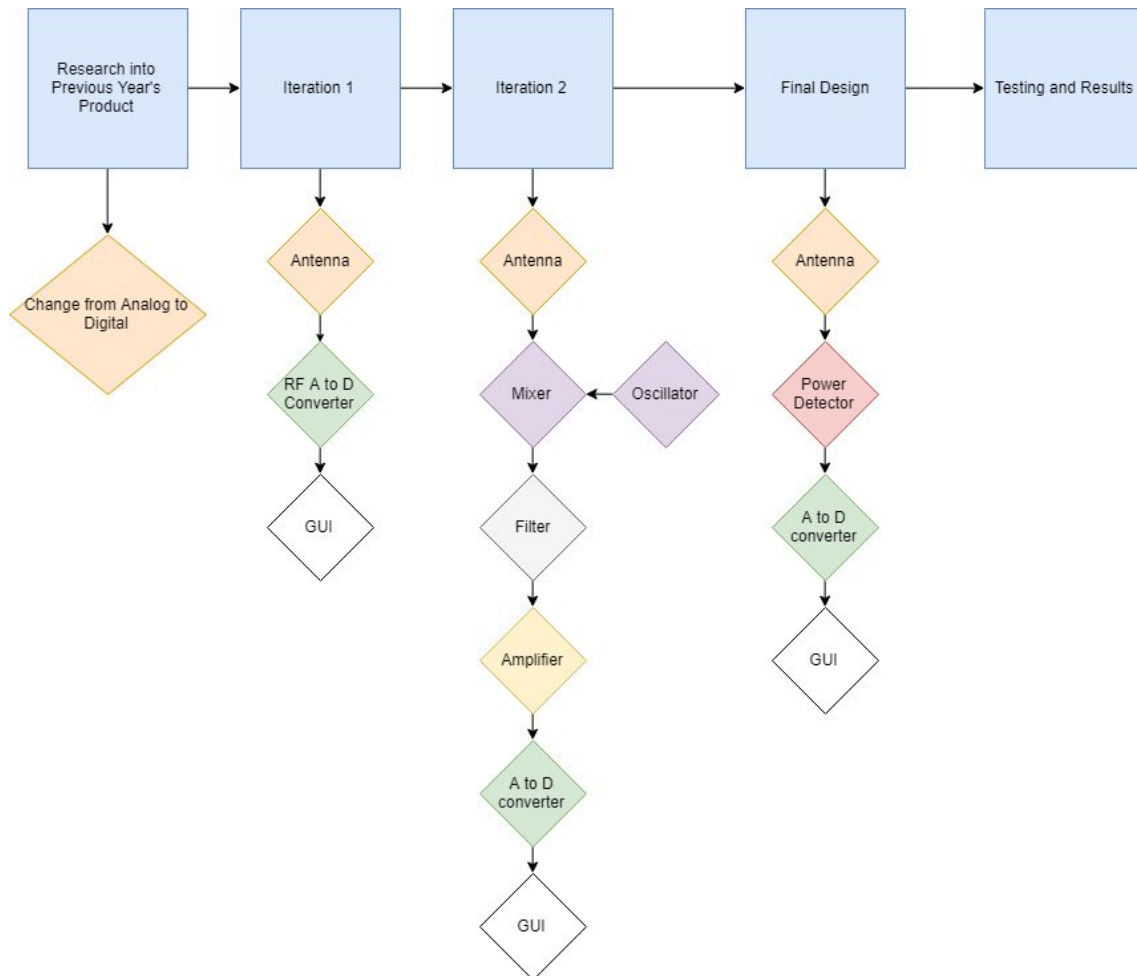


Figure 11: block diagram of design process

3.2 Components Used in All Iterations

3.2.1 Spiral Antenna



Figure 12: a picture of the spiral antennas

The spiral antennas used would be situated as shown above in figure 12. They are situated by a 3D-printed stand that is holding them 120 degrees from each other. This allows full coverage of area in which it is placed, while cutting down on the number of antennas needed. If more antennas were used then a more accurate picture could be generated, but it would increase the cost to fabricate and another set of backend would be needed for each subsequent antenna. We decided to forgo the use of the dipole antenna because it would require another set of equipment to use. The spiral antennas used were composed of FR4 material, for the Printed Circuit Board (PCB), and copper for the antenna spiral itself. The three antennas are connectorized and use SubMiniature version A (SMA) gold connections to connect with the rest the backend of the project. The antennas were borrowed from the electrical engineering department. They come from basic kits used to do experiments at the 915 MHz range. They are widely used and have good track record of being reliable.

3.2.2 GUI

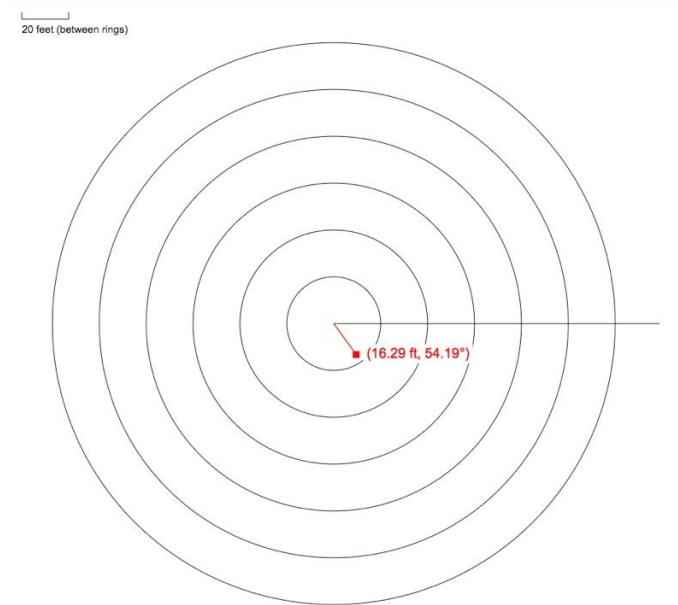


Figure 13: a picture of the GUI

The GUI used was based off a Raspberry Pi interface, shown in figure 13 above. We choose to use a Raspberry Pi GUI because the previous year's project team purchased the Raspberry Pi and screen. By using their setup, we were able to save budget costs and also improve on their code. The Raspberry Pi offers an easy to use program that allows the user to make a professional looking GUI. Another added plus was the extensive experience our Computer Engineering partner had in using the Raspberry Pi. This allowed fast debugging of problems and efficient implementation of code.

Our GUI uses a machine learning algorithm to triangulate the location of the drone. An article in the International Journal of Advanced Research in Computer Science used machine learning to extract information from a database. We used this article to critique our own technique and optimize performance [5]. We compiled a large database of data of previous known locations of the drone. Our algorithm will essentially peruse this database and compare the drones current location data from the ADC to our database to determine its location in relation to our setup. By using machine learning, we will be able to track the drone very accurately against empirical data.

3.3 Iteration 1

The block diagram below, figure 14, shows the initial design plan for our antenna system. We started with the idea to just use an RF A to D converter to transfer the drone signal straight to the GUI. This would have simplified the whole process and allowed us to work with the complete unaltered signal right in a digital framework.



Figure 14: iteration 1 block diagram

3.3.1 RF A to D Converter

The RF A to D converter theoretically used was the ADC08D1520, shown below in figure 15. This ADC would have allowed us to convert the drone signal straight to a digital representation. The major problems with this iteration was the cost of the ADC, ~\$1500 total, and the need for an FPGA board to integrate with our GUI. This design was very nice because it only required one component between the antenna and the GUI. However, because of the cost associated with ADC and its need for a dedicated FPGA we moved onto our next iteration.



Figure 15: iteration 1 block diagram [19]

3.4 Iteration 2

The block diagram below, figure 16, shows the second design plan for our antenna system. Due to the constraints on the RF ADC in iteration 1, we focused on finding an ADC that would work for our application and building a support system around it. We found that to work with a regular ADC we would need to downconvert the 915 MHz RF signal down to something a regular ADC could handle. We did this by using an oscillator and mixer in conjunction with a filter and amplifier to give us a clean signal that our ADC could handle. This iteration required four more components than iteration 1. The cost was reduced to a total cost of ~\$468. However, due to the large increase in the amount of components the power usage of the whole system increased and four DC power supplies were needed to power all the components.

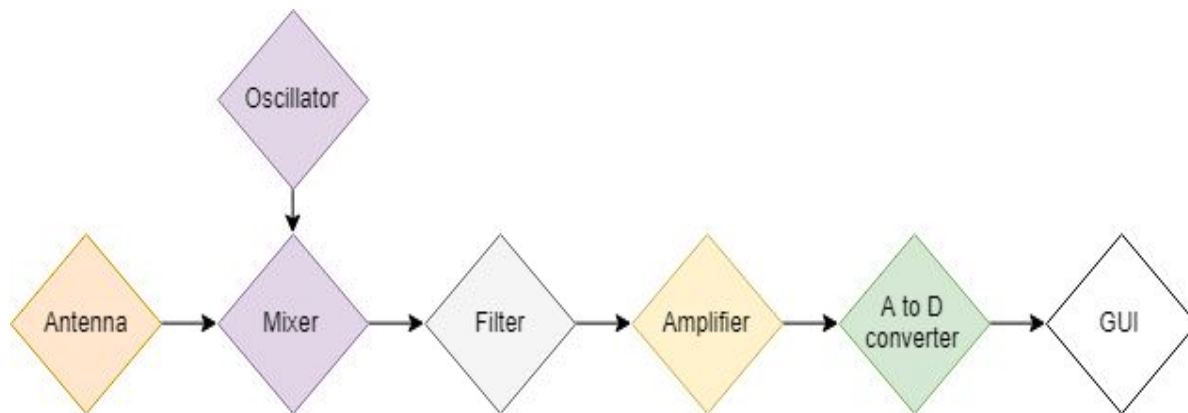


Figure 16: iteration 2 block diagram

3.4.1 Mixer

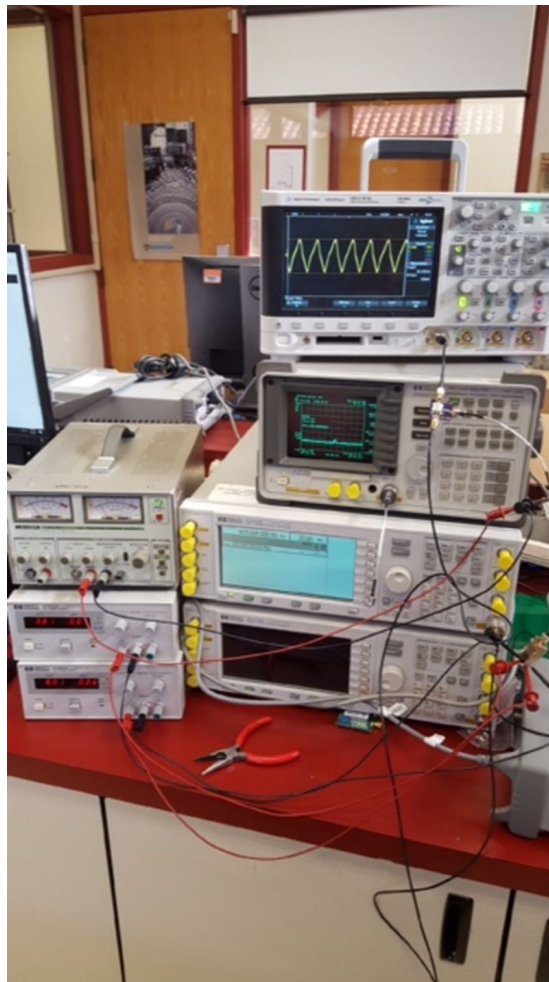


Figure 17: a picture of the mixer converting the signal to 60 KHz

The mixer used in our application was the SYM-63LH+. We choose this mixer for a multitude of reasons. The most important was its ability to bring the RF signal to the KHz range. Our ADC, which will be highlighted in the next section, is only able to handle at most 100 KHz. For this reason, the mixer had to be able to bring the RF to the DC range. This is shown in figure 17 above. Another consideration was the connectability of the mixer. SMA connections were desired for all parts and this specific part had the ability to use SMA connections for the RF, LO, and IF ports. Finally, cost factored into choosing this particular mixer. Minicircuits, the company this part was purchased from, had Gerber files for this mixer. This allowed us to fabricate our own board and save hundreds of dollars.

3.4.2 ADC

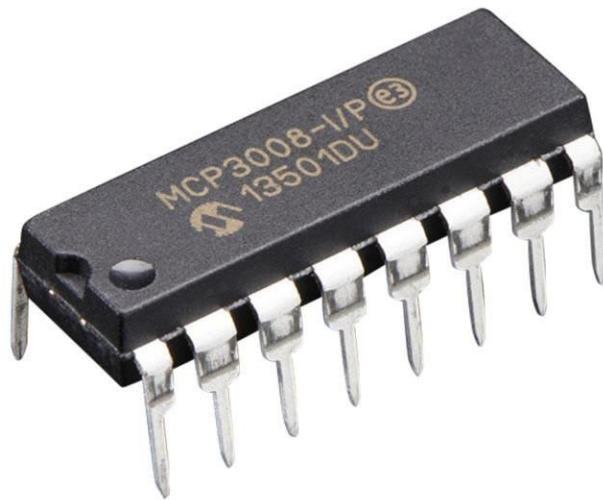


Figure 18: a picture of the ADC [20]

The ADC used was the MCP3008, shown in figure 18 above. This particular ADC was chosen because of its accessibility. The major problem encountered during constructing this project was the ADC. Getting an RF ADC that is able to sample at the MHz range would have required thousands of dollars and a dedicated Field Programmable Gate Array (FPGA). This would have expanded the scope of our project immensely if we were determined to use an RF ADC. Because of this, we decided to pick an easy to use ADC and build all the other parts around it.

The two most used ADC's with the Raspberry PI are the MCP3008 and the ADS1x15 series. The ADS1 offered more bit precision, but a lower sampling rate. Sampling rate was the biggest concern, as that is the difference between this ADC and an RF ADC. We learned from our ELEN 110 class that the Nyquist theorem states that you need at least a 2 times conversion rate between the sampling rate and the maximum frequency in the signal. In other words, your sampling rate can at least be 2 times larger than your operating frequency. The max sampling rate of the ADS1 is 80 sps. Our f_c after downconversion of RF is 60KHz. The MCP3008 has the sampling rate of 200 ksp. This made the MCP3008 a much better choice for our application.

3.5 Final Design

The block diagram below, figure 19, shows the final design plan for our antenna system. Now that we were successfully able to get a signal to the ADC we concentrated on optimizing the system to fit our original project objectives. We found the the cost of iteration 2 was still a little to high and the power requirements made the system not robust or portable enough to fulfill our requirements. Because of this, we decided to reduce the amount of components between the antenna the the ADC. This led our team to a power detector that was able to take a wide range of RF signals and convert them to DC values that our ADC was able to read. This reduced the cost and power requirements, but caused us to lose the ability to work with the phase of the drones signal.

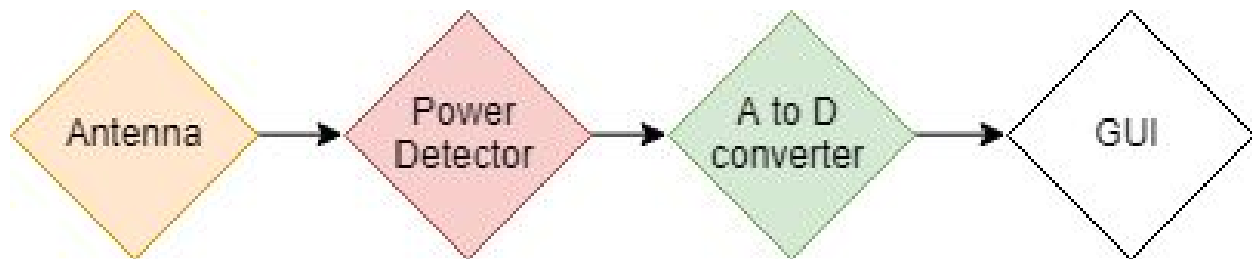


Figure 19: final design block diagram

3.5.1 Power Detector



Figure 20: a picture of the power detector used[21]

The power detector used was the ZX47-40-S+. It takes an input of a wide range of signals from 10 MHz to 8 GHz. It outputs a DC voltage that linearly corresponds to the power received from the signal. Figure 21 below shows how the input power corresponds to the DC output voltage. The relationship is inversely proportional and linear for powers greater than -40 dBm. This power detector vastly increased the amount of signals our system could process at the cost of no longer getting phase data from the signal. It also only cost \$90 a piece which considerably cut down on the cost of the entire system. By outputting a DC voltage, we also did not have to worry about the ADC sampling rate.

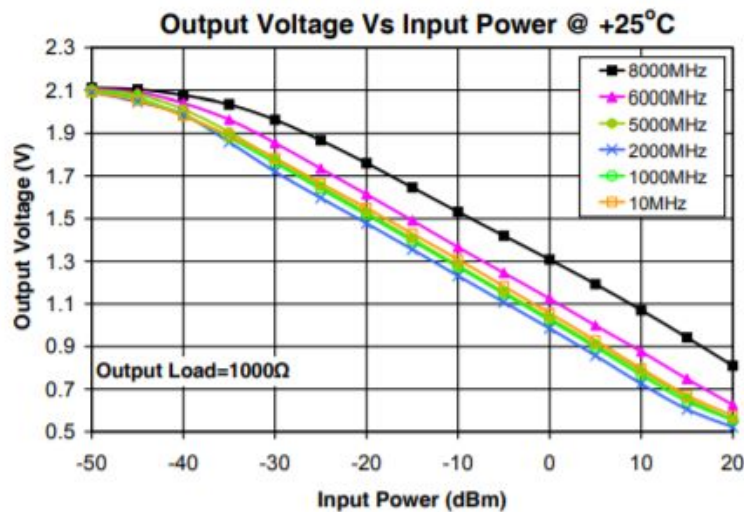


Figure 21: a picture of input power vs. output voltage[21]

3.6 Comparison

Through our entire design process we were able to both reduce the cost and power requirements of our antenna system. Iteration 1 allowed for the least amount of components used, but had the highest cost. Iteration 2 significantly reduced the cost of the system, but significantly increased the amount of components and power needed. The final design reduced the amount of components needed and again reduced the cost and power required, at the cost of not being able to use the phase of the signal. We found that the final design optimized the system to such an extent that we were willing to lose the data associated with the phase. Table 2 below, gives a comparison of the various iterations against the final design.

	Iteration 1	Iteration 2	Final Design
# of Components	1	5	2
Power Requirements	?	4 DC Power Supplies	1 DC Supply
Total Cost	\$1500+	\$468	\$286

Table 2: a comparison of the different design iterations

4 Software Implementation

This Chapter will cover the software needed to track a drone frequency. We went through a number of algorithms to find the most efficient and accurate way to parse through a large amount of data. The main focus of the software was to accurately predict the location of the drone based on previous data collected.

4.1 Use Cases

The primary use case of the project is to track the location of a 915MHz signal, usually from a drone. The use case diagram (Figure 22) is therefore fairly sparse. The other possible use case is to find signals in a surrounding area, however the way in which the user interacts with the system is identical for both of these use cases, i.e. by starting and stopping the tracking.

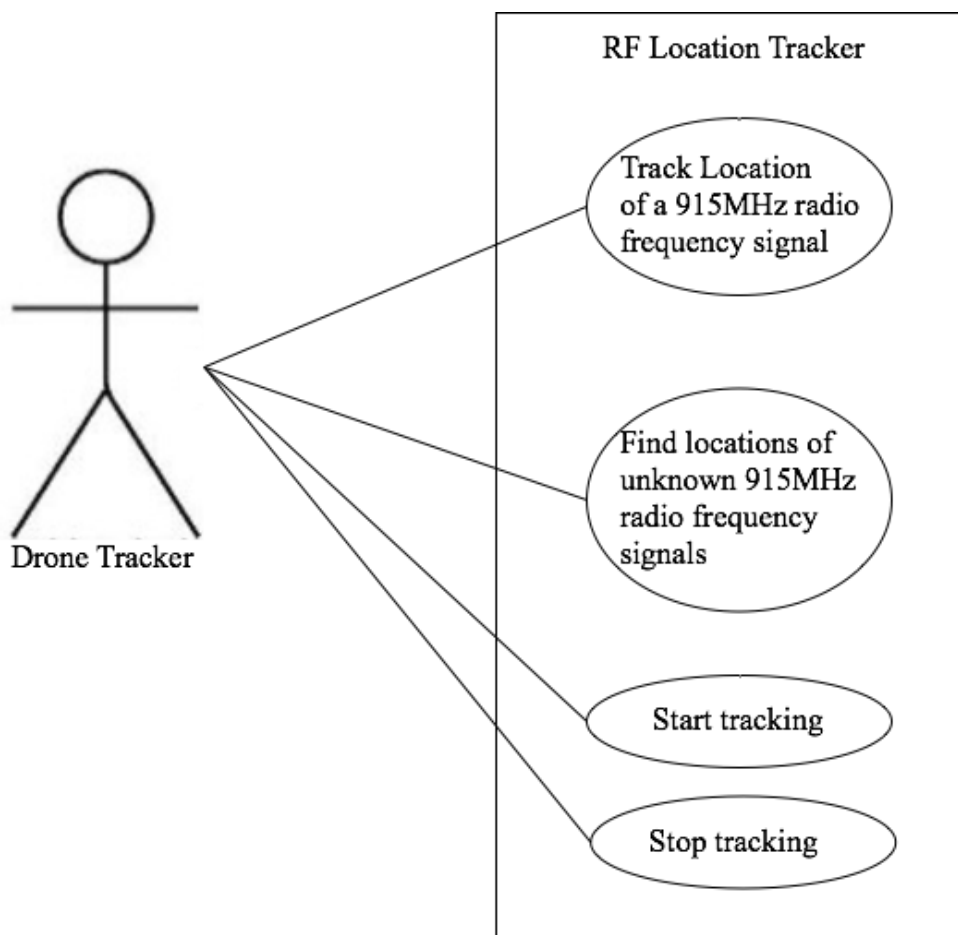


Figure 22: use case diagram

4.2 Training the Location Model

Because an actual RF environment is difficult to model, an equation to calculate distance or angle of the signal cannot be predefined. Thus, the receiver was built first and then two models were trained. One model calculate the distance of the signal, and the second will calculate the angle.

There are three signals that will be received - one from each of three spiral antennas. The reading from the adc from the power detector is used as the input of the model.

To train the model, data was collected at different distances and angles. The distance and angles will manually be recorded as the labels for the training data. We knew that the farther away from the receiver that we collected training data points, there would be exponentially more area to collect points in because $Area = \pi * radius^2$. To fix this, we set up a raspberry pi application that used stepper motors to move the receiver away from the signal and rotate the receiver at exact angles. We moved the receiver away from the signal in 3 inch increments down an 5 foot track. At each distance, we measured the signals at 102 different angles, each approximately 3.5° appart, spanning 360° . The final data collector robot is shown in Figure 23.

We ran multiple data collection runs, and iterated on the design of this robot to get it to run smoothly. In a final run, we let it run overnight for 14 hours. In the morning, we had approximately 26,000 data points. However, each lap down the track collects only $(102 \text{ angle measurements}) * ((5 \text{ feet} / 3 \text{ inch increments}) \text{ distance measurements}) = 2040$ points. So, each distance, angle pair was measured 13 times.

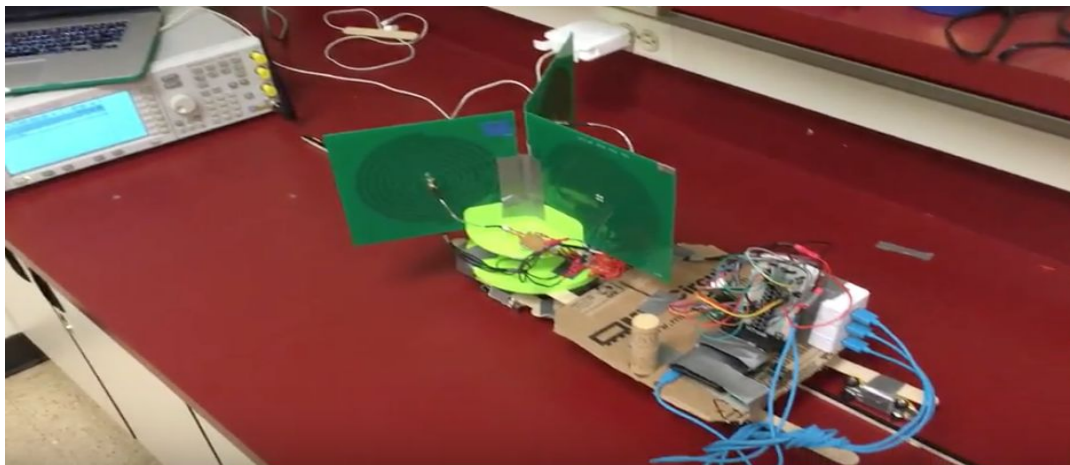


Figure 23: training system

The code that does collects the training data is shown in Appendix C.

4.3 Machine Learning Algorithms

4.3.1 K-Nearest-Neighbors Algorithm

We used two machine learning algorithms to try to most accurately predict the location of the signal. The first one we tried is the k-nearest-neighbors algorithm, and the second is a neural network.

The k-nearest neighbors algorithm essentially looks at all of the training data each time it predicts a point. It finds the K nearest (euclidean distance) neighbors to the current signals it is receiving, and averages the labels for those training points. This average is the prediction. We found that the algorithm had the most accuracy using 10 data points. Remembering that we measured the signals at each (angle, distance) pair 13 times, we end up average fewer than that number of test points to find the most accurate location of the signal. Because the K nearest neighbors iterates over so all the data points, it is generally a difficult algorithm to scale to larger sets of data points. For our application, however, we did not notice it cause any performance issues. An, example of the k-nearest neighbors algorithm is shown below in figure 24 below.

Example:

Training Set: Signals → Position [3,4,5] → (10in, 15°) [2,5,2] → (20in, 225°) [2,1,4] → (30in, 130°)	Running the tracker: K = 1 Signal Received: [4,4,5] Prediction: (10in, 15°)
--	--

Figure 24: example of k-nearest neighbor algorithm

Neural Network

The neural network we used had three input nodes (representing the three signals from the ADC), 4 hidden layers with 60 nodes each, and 2 output nodes (representing the distance and angle). An example image of the neural network is shown in Figure 25.

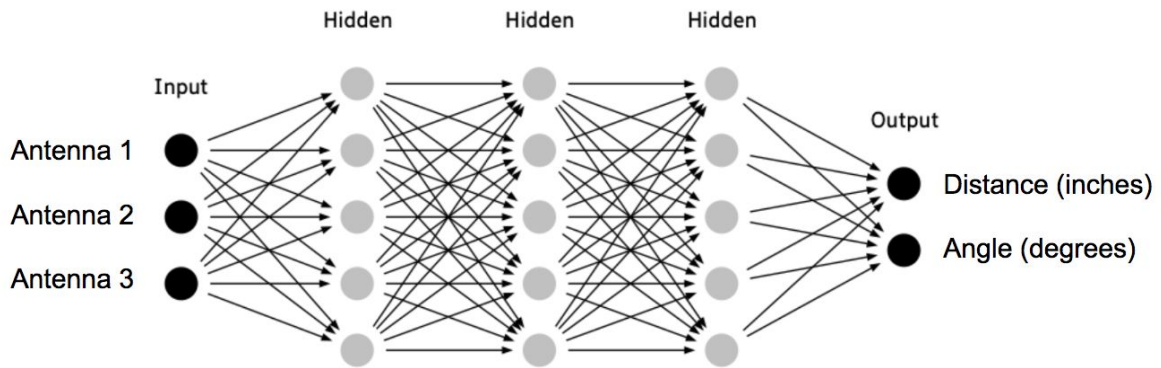


Figure 25: neural network

To our surprise, the K nearest neighbors algorithm and the neural network performed about the same, though we thought the neural network would be better. The code for the machine learning algorithms is shown in Appendix A.

4.4 Pipeline Architecture for Deployment

The deployment application (in contrast to the application used to train the model), will have a pipeline architecture with timed loops, each running on separate threads. The software receive a data stream from the ADC representing the power received by each of the three antennae. The three signals will be fed to the pre-trained model. Given the three inputs, the model will produce polar coordinates for the signal (distance, angle). This will then be displayed in the GUI. If the signal is close to the receiver, the location map in the GUI automatically scales down to show more detail. Likewise, if the signal is far away, the map scales out to show this location. A dynamic scale key for the map is be shown. The pipeline is shown in Figure 27.

```
1.598 | 0.013 | 0.01 | 1.595 | 0.013 | 0.01 | 0.01 | 1.56 |
1.598 | 0.013 | 0.013 | 1.595 | 0.013 | 0.01 | 0.01 | 1.553 |
1.598 | 0.016 | 0.016 | 1.595 | 0.013 | 0.01 | 0.01 | 1.55 |
1.598 | 0.013 | 0.013 | 1.595 | 0.013 | 0.01 | 0.01 | 1.557 |
1.595 | 0.013 | 0.013 | 1.592 | 0.006 | 0.01 | 0.01 | 1.55 |
1.598 | 0.013 | 0.013 | 1.595 | 0.013 | 0.01 | 0.01 | 1.553 |
1.598 | 0.013 | 0.013 | 1.592 | 0.006 | 0.006 | 0.01 | 1.553 |
1.592 | 0.013 | 0.016 | 1.595 | 0.013 | 0.01 | 0.01 | 1.557 |
1.598 | 0.013 | 0.013 | 1.592 | 0.013 | 0.003 | 0.01 | 1.557 |
1.595 | 0.013 | 0.013 | 1.592 | 0.013 | 0.01 | 0.01 | 1.557 |
1.595 | 0.013 | 0.016 | 1.595 | 0.013 | 0.023 | 0.019 | 1.557 |
CTraceback (most recent call last):
  File "tests/simpletest.py", line 43, in <module>
    time.sleep(0.5)
KeyboardInterrupt
pi@raspberrypi:~/Desktop/rft $ ^C
pi@raspberrypi:~/Desktop/rft $ kill 1117
```

Figure 26: a picture of example DC voltages that are fed into the model

4.5 Smoothing Algorithm

The previous team's implementation had a fair amount of jitter on the location map, which we intended to diminish in this iteration of the project. A smoothing algorithm is used to partially achieve this. Because the data is coming to the program continuously, there is an abundance of data to use to pinpoint the location as best as possible. The GUI is updated every 500ms, which means that 500ms of signal data can be used to determine the most likely position of the signal. So, every 100ms, the angle and distance is calculated from the preceding 100ms of data. We are able to do this 5 times for every update of the GUI. With these five polar coordinates, the final point to be displayed is the average of the three which have the lowest Euclidean distance to the previous point displayed on the GUI. This has the effect of reducing jitter by choosing a point close to the previous point and by averaging readings. The entire software process is shown in Figure 27.

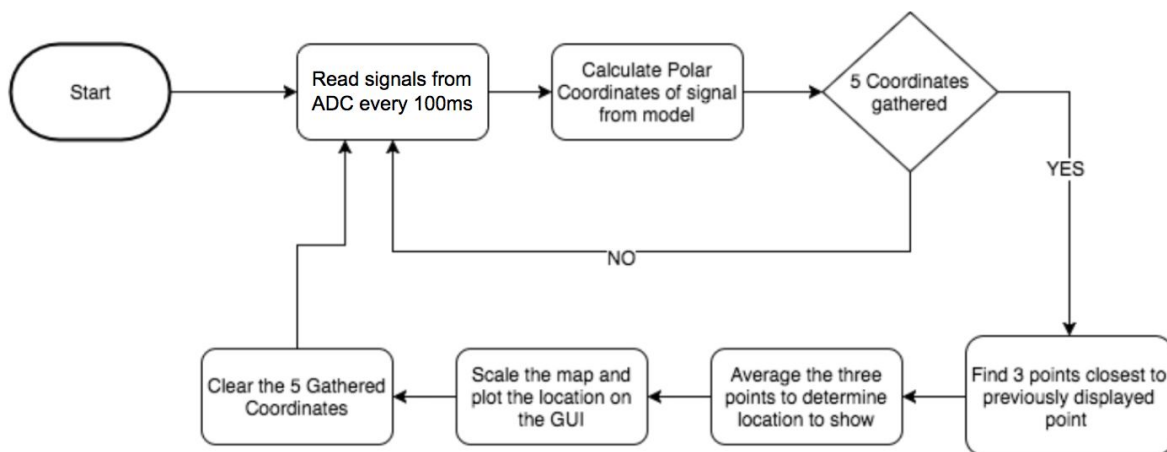


Figure 27: location prediction flow diagram

The code for the location tracking and GUI is included in Appendix B.

4.6 Software Packages

The language used is Python. The application runs on the Raspberry Pi's Raspbian operating system (fork of Linux). The following software libraries are used:

- HTML Canvas - GUI framework
- Django - web app / server
- numpy, scipy, tensorflow, pandas - running the neural network and k-nearest-neighbors algorithm
- Adafruit_Python_MCP3008 - Interface with MCP3008 Library

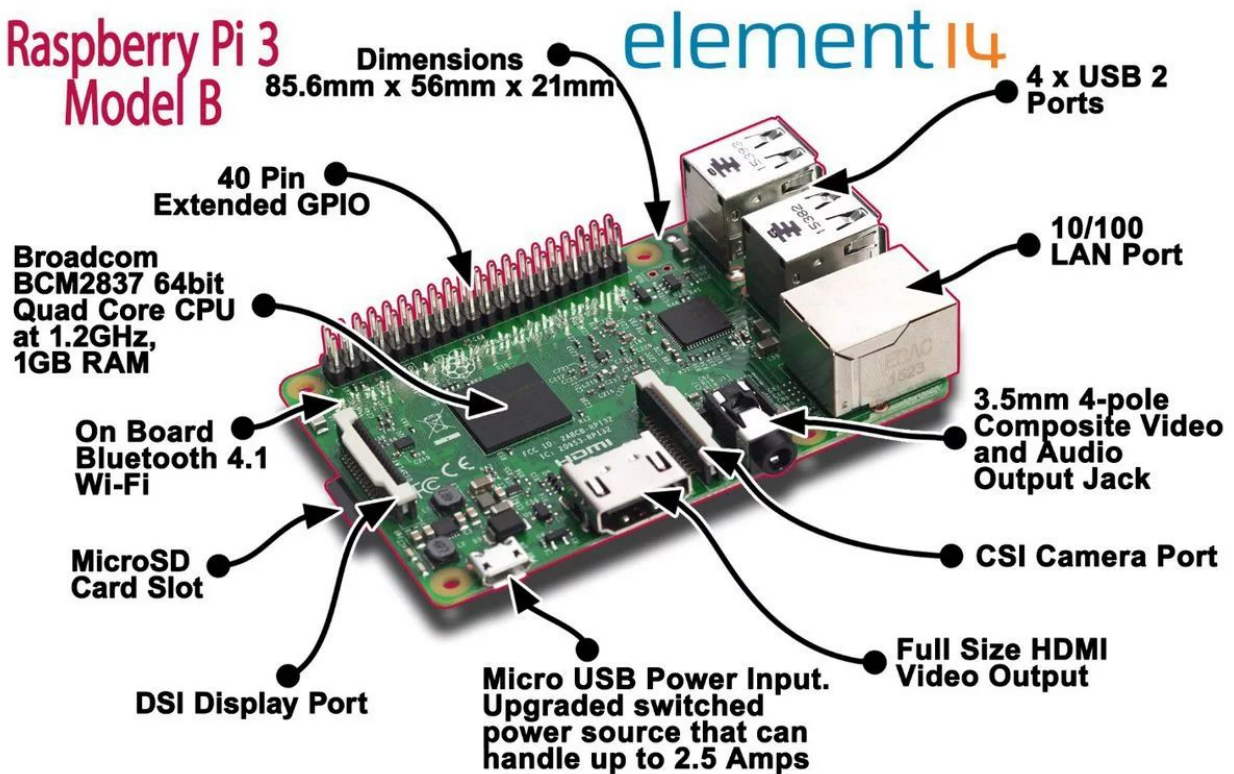


Figure 28: picture of the raspberry pi used[22]

5 Test Plan and Results

5.1 Spiral Antenna: 915 MHz

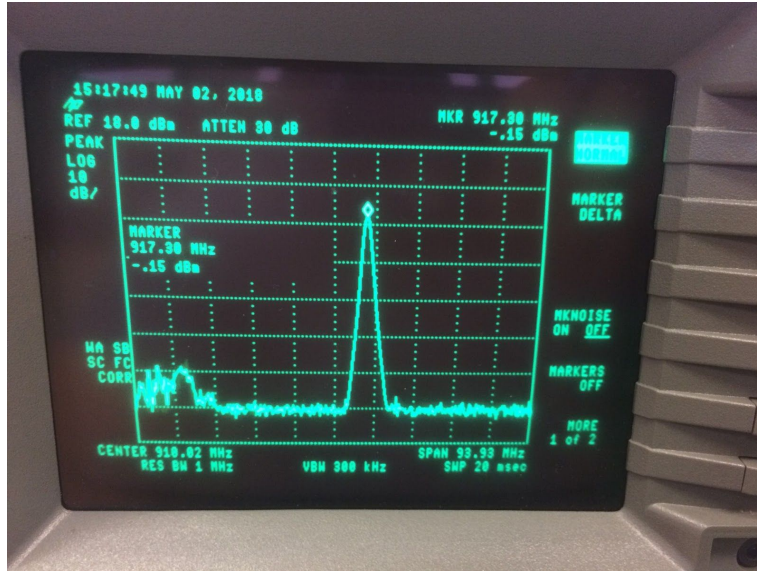


Figure 29: using a spectrum analyzer to test spiral antenna

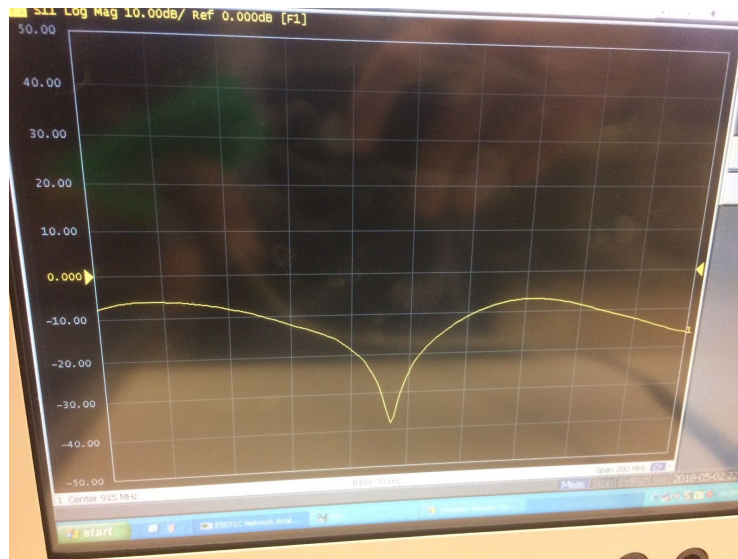


Figure 30: picture of S11 parameters of spiral antenna

We tested the S11 parameters of the spiral antennas using a VNA and checked the received RF signal power density using a spectrum analyzer . We found them to be working at ~915 MHz which validated the integrity of the antennas used.

5.2 Power Detector: ZX47-40-S+

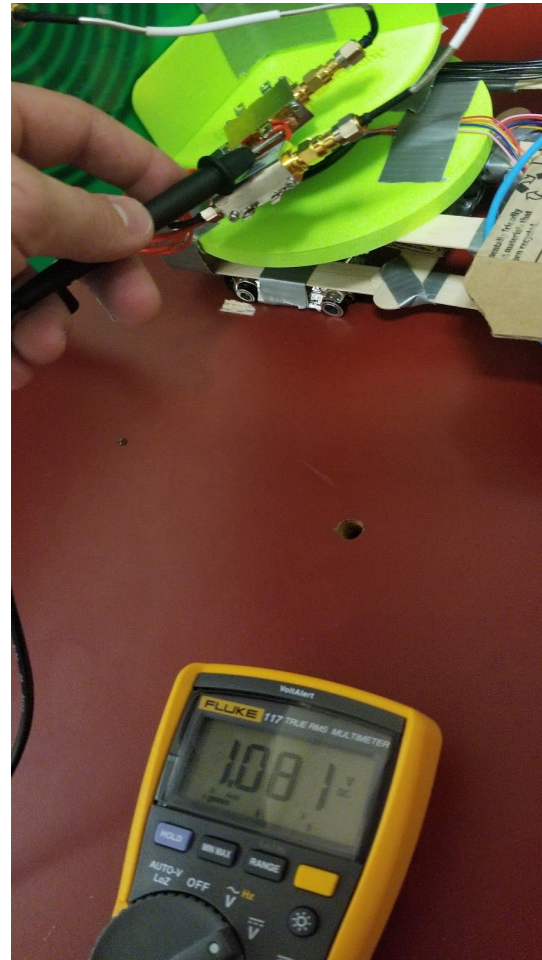
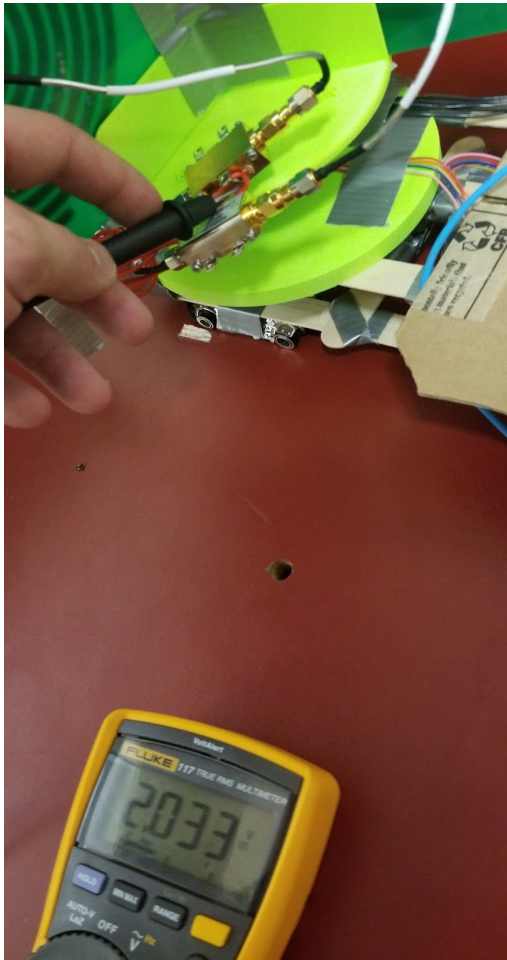


Figure 31 (left) and 32 (right): picture of the DC voltage output of the power detectors

We tested the power detectors at no signal, shown in figure 31, and with a high power signal, shown in figure 32. The output DC voltages we received were as expected from the datasheet with small adjustments for real world variables.

5.3 ADC: MCP3008

We tested the ADC by varying the input signal it was receiving and checking to make sure that it linearly changed the output 0 to 1024 data bit that the ADC was outputting, with a larger signal corresponding to a larger data bit. Figure 26 shows the final output once the data bit was converted to a corresponding voltage reading.

5.4 GUI

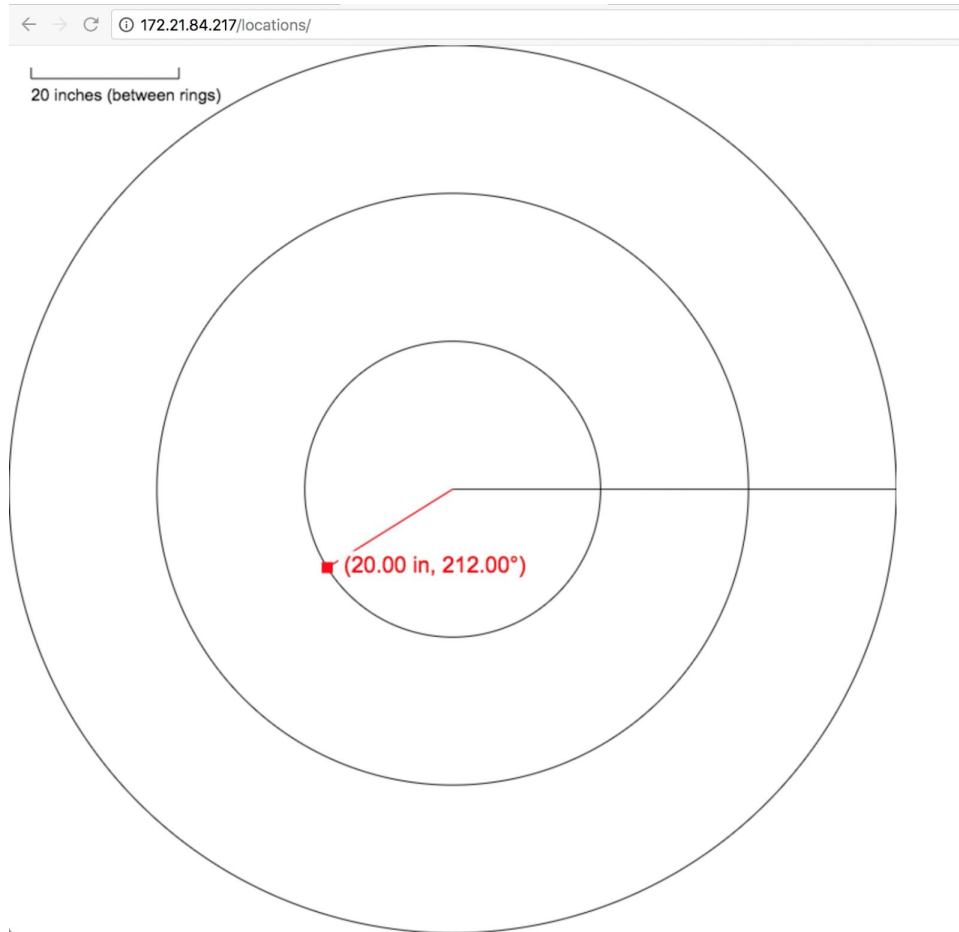


Figure 33: picture of GUI used to display drone position

We tested the GUI by changing the location of the transmitter and timing how long it took to update the GUI. It updated as expected with no discernable delay between what was expected and actual.

6 Final Project Analysis

In this chapter we analyzed the results from our test procedure. We used techniques from our electrical engineering courses ELEN 104 and ELEN 105 to analyze the RF side of the project. We used techniques from ELEN 50, ELEN 100, and ELEN 164 to analyse the circuit layout and the results from the components. ELEN 115 and ELEN 116 helped us to parse through datasheets and check key parameters in the components. However, we needed to supplement our learning with online videos and lectures for the mixer, ADC, and component setup as it was not covered in our classes.

6.1 Objectives met

Expected	Actual	Met?
Track a 915 MHz Signal	Track a signal from 10 MHz to 8 GHz	Yes
360 Degree Coverage	360 Degree Coverage	Yes
Range of 15 ft	15 ft	Yes
Max angle of less than 5 degrees	Max angle of less than 15 degrees	No
Distance resolution of 1 foot	Distance resolution of 3 inches	Yes
Real Time Display	Real Time Display	Yes
Portable/Rugged	Portable	Yes
2 Hour Operating Time	4 Hour Operating Time	Yes

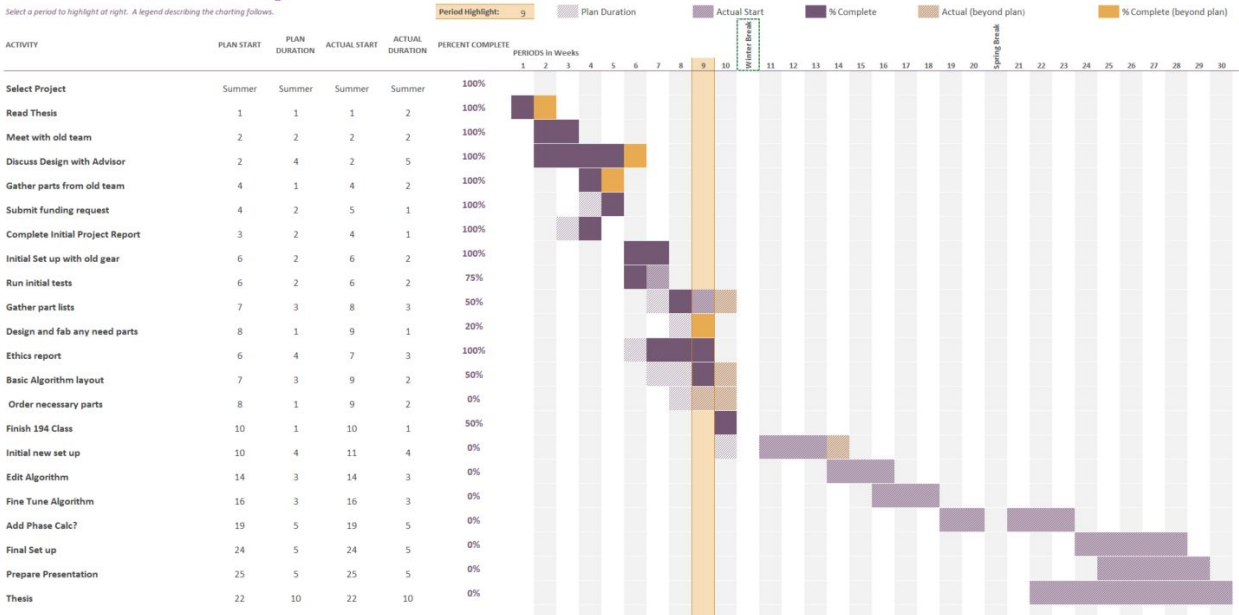
Table 3: data of objectives met

6.2 Gantt Chart

6.2.1 Proposed

Drone Tracker Project Planner

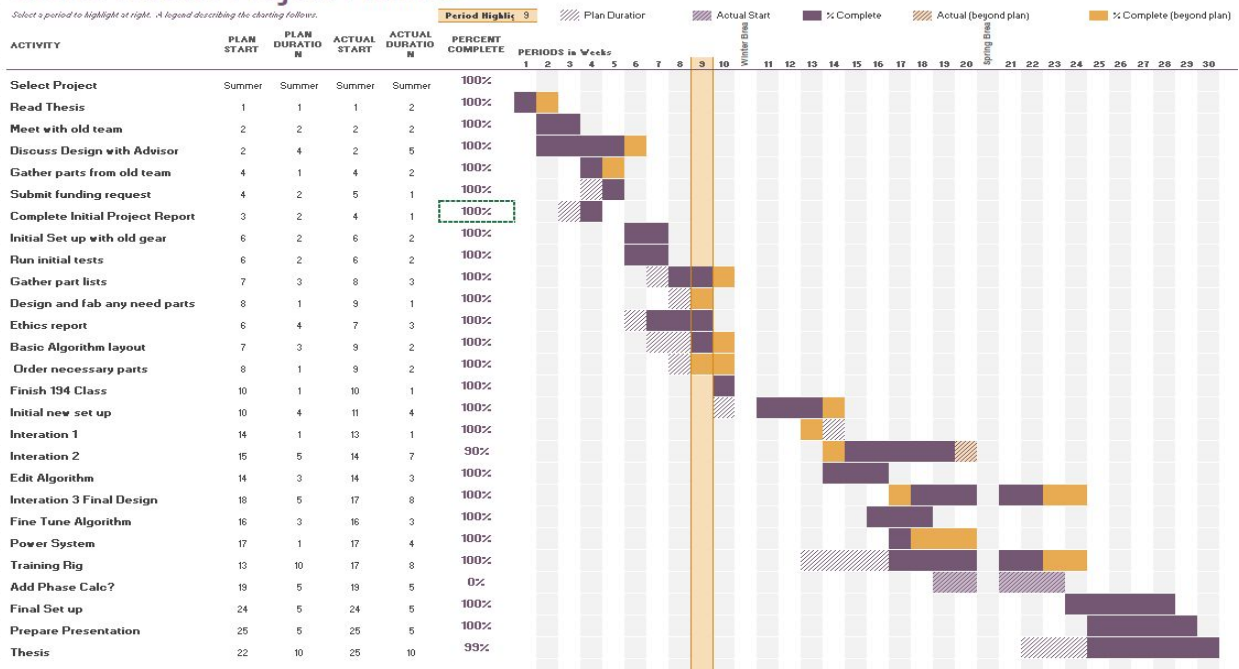
Select a period to highlight at right. A legend describing the charting follows.



6.2.2 Actual

Drone Tracker Project Planner

Select a period to highlight at right. A legend describing the charting follows.



6.3 Future Work

For future work the major expansion that could be taken is to expand our project into 3-dimensional space. That would vastly improve the system and make it much more applicable to real life situations. Two other factors would be in improving accuracy. This would be through both working on neural network to increase accuracy readings and building a database for different environments. These two factors would improve the accuracy and usefulness of the product.

7 Professional Issues and Constraints

7.1 Ethics

The biggest ethical question that our project raises is personal and business security. Our project actually helps answer this question. Today Drones create a big ethical conflict as Drones can invade people personal privacy. Our product could both help drone operators to stay out of no fly areas and track rogue drones. If a customer is worried about the possibility of a drone flying on their property, they could use our product to alert and track a possible drone intrusion.



Figure 34: picture of professional considerations[23]

7.2 Science Technology and Society & Usability

Social Sustainability Benefits for User

1. Portability
2. Ruggedness
3. Usability
4. Precision
5. Modularity
6. Aesthetic
7. Affordable

Our product directly satisfies a specific need in the industry. Right now, GPS or wireless triangulation is used to track drones. The major drawbacks to GPS are precision and the major drawback to wireless triangulation is portability. Our product will cover both needs in an easy to use package that is affordable for both commercial use and industry. We built our GUI and product to be easy to use and robust. The modularity by making most of the backend digital allows the user to change our product to fit their needs. They don't necessarily have to use our product for drones as it is just tracking a signal. If they wanted to, they could change the setup minimally and be able to track any signal for any application. There are no major negative impacts on a user's health or welfare. Our product does not interact with the user in any way that could be harmful on the signal bases. It also does not include any moving parts, so there is no risk of harm to the user's self. The only concern for our project is not operational, but more of an ethical question. As our product can be adapted to track any signal, there could be issues with privacy if our product is used in a nonconventional way. However, this is not a special circumstance with our product as anything can be used incorrectly or for morally wrong reasons.

7.3 Civic Engagement

The following regulatory agencies and professional societies all have compliance regulations which were taken into account in our project.

1. UL - electrical compliance
2. FAA - Drone compliance
3. IEEE - EE research and funding
4. DoD - Drone security
5. FSC - Spectrum use and registration

7.4 Political Impact

This system will allow people to track drones better, having political impact in our streets at home and in the military. Drones will hopefully be monitored closely by a system like ours in every home. Drones that violate property laws can be flagged and prosecuted. In the military, our tracker will help our allies fly drones with better coordination, and will help us find and destroy enemy drones.

7.5 Economic Impact

Economic Sustainability Current Market Size

1. Delivery
 - a. Users - 300 million [11]
 - b. \$17 Billion with drones alone [11]
2. Search & Rescue
 - a. 400 units in the United States alone [11]
 - b. \$113 Billion [11]

The Economic sustainability of our product is quite high. Our product has very high demand and a very wide use case. There are plenty of potential users to support the future delivery of our product. Not only does our product reduced the need for other products, it is very cost effective it self. Since our product can replace other inefficient products we can save customer and producers materials and labor. One example of this is our products ability to replace the current industry standard WiFi triangulation system. The main problem and biggest set back of this system is that it need multiple access points to work. This takes up space and materials along with all the labor to build this access points. Our system one requires one point system and uses much less material then all those access points. This make our product economic sustainability from a material and labor standpoint. Our product is also economically sustainable from a profit cycle standpoint. Our current system cost is around \$300 since we did not have antenna board fabrication and used the ones available in the lab. If this were to go into production this cost could be dropped by more than 50%. At current market conditions a product of this caliber could sell for 2-3 time its current production cost. This means many more can be produced in a viable way.

7.6 Health and Safety

Health and Safety are two of the primary uses of the RF system locator. The product can promote safety in the home by detecting unwanted drones. Furthermore, our product can be used to guide drones through buildings without crashing into walls, another safety feature. The product does not pose any health risks, and we have made sure to insulate our circuitry to prevent electrocution, through the voltages and currents our product uses are low. One safety issue yet to be improved on is the pointy corners on the antennas. These could be rounded or protected by enclosing the system in a case.

7.7 Manufacturability

Our design has not been refined to be manufactured easily yet. Nonetheless, the electrical components used could easily be reduced into a more compact system. A few of the components would need to be put in place and connected by a human, but many of the parts that are now separate could be condensed. Our product was a proof of concept.

7.8 Usability

The RF system locator was built around usability. The device was made to be easy to use and allow anyone to operate. The easy to use GUI allows users to easily track their drone or AV and then monitor its real time location. Also for a future build out and manufacture of the product we would include easy to use environment packages that could be uploaded to the design. This would allow users to set up our product in any environment without having to rerun the environment setup. For more advanced users they will be able to run custom environment calibration and upload those designs to the locator. This will allow for greater accuracy. Also all of the power systems are plug and run format. They can be charged off of a standard USB port and can then be plugged straight into the locator.

7.9 Environmental Sustainability

7.9.1 Materials Used

- Plastic- 3D-printing, case for components, part of stepper motors, wheels/legos
- Wood- Track for calibration setup
- Metals- Components, attachments, cables, traces, holdings, solder, servo motors
- Silicons- PCB boards, antennas
- Rubber- Wires and connections
- Chemicals- Battery

There are not any alternatives for some of the components. The silicons, metals, rubber, and chemicals are all resources that cannot be substituted for anything else. The one major resource that could possibly be changed would be the 3D-printing holder to the antenna system. We could use a wooden holder, which would be more sustainable. The only problem is that we would not be able to rapid prototype as well. Thankfully, most of the components that we used do not have a major impact on the environment besides the chemicals from the battery. The silicons and rubbers are somewhat of a concern because of how hard it is to recycle them. We decided that the advantages of having a self-contained system and being able to rapid prototype outweighed the environmental impacts that came from the use of those resources.

The energy resources needed for the raw materials are mostly in the processing stage. Making the plastic and silicon wafers are specialized processes that require a robust infrastructure to acquire. The manufacturing of the product could be mass produced and cut down on the amount of energy needed to build our product. The energy needed to bring the product to the customer would primarily come through the transportation of the product to the stores. This would be a cost of gas which is not a renewable resource as does detriment the environment. The resources needed for operation would primarily come from the battery/bank of batteries that would be needed to power our product. We choose components that are as low energy as possible, but still need some power. Our product does not produce any ambient pollution so no energy would be needed to offset this.

Our product is design to have a very long life cycle. All of the build components are composed from very high grade part that are meant to last. Because of the high build quality a production build should cost between 1500- 5000 dollars. The energy required for operation is extremely low, around 3W. This means that our product could be operated on a battery for several hours. The product life is expected to be 5-10 years. The main reason for the life expectancy cap of 5-10 years is because of the technology becoming outdated. The components should last well beyond this life and could even be

recycled after the product is done being used. Because the some of the product could be recycled or reused the impact on the environment will be minimal. The biggest problem would be disposal of any battery that is used to power the product, but there are battery recycling facilities that could take care of the disposal.

7.10 Lifelong Learning

Learning is truly done by oneself. This project made our team do deep research, find parts and software packages, and construct a complex system without much hand-holding from our advisor. The project taught us independence, and that we can never rely on anyone but ourselves to get a job done. The most powerful thing that the development of the internet brought to the world is the abundance of information. New technologies can be developed at lightning pace compared to before, and civilization will advance at new speeds.

7.11 Compassion

A main objective of this project is to aid in search and rescue missions. Better coordination of drones will allow drones to become more effective in relief of natural disasters. If our technology were good enough for such use cases, we would license it for free to such causes, while still maintaining profit in commercial applications. Technology is only good when it helps people, and our technology is aimed at doing so and providing a safer world with the advent of drones.

Com-pas-sion [kuhm-pash-uhn]

noun : a feeling of deep sorrow for another who is stricken by misfortune, accompanied by a strong desire to alleviate the suffering.



Figure 35: picture of definition of compassion[24]

8 Conclusion

GPS and wireless triangulation are two implementations that are currently being used to track drones. The two biggest limitations of these systems are accuracy and portability. In our paper, we proposed a solution to these problems in the form of a wireless RF location tracking system. Our system is able to track a 915 MHz signal and display the position of the signal on a GUI. It is accurate to within a foot, able to withstand normal wear and tear in an outside environment, and built with the ability to track any signal. Our system is composed of a Spiral Antenna, Mixer, ADC, and GUI which are set up in series to translate the signal from analog to digital very quickly. The GUI utilizes a machine learning algorithm to estimate the location of the drone from a large database of information. Figure 36, shows a conceptual design of our finished product.

Some future work that a new year's team could work on in regards to our project are twofold. The first is the power system. An efficient power system composed of an isolated source of power would help with the portability of our product. Currently our project requires a plug in to the house/outside power supply. If an internal system was built then the system would be much more compact. Another avenue of approach is bringing the system to 3 dimensions. Currently, our project is able to just do 360 degree coverage, but not elevation. The next year's team could add another antenna to our design and take our product to a 3 dimensional model.

The original purpose of the project, bringing the system to 360 degrees with foot accuracy, was achieved. Our simulations show that our product will be able to work in a variety of environments and settings. Future work could help to expand the utility of our project, however the product is readily available currently. We set out to track a drone precisely and were able to achieve it.



Figure 36: concept of final product[25]

Bibliography

- [1] Burzichelli, Corinne Dowling. 2016. "DELIVERY DRONES: WILL AMAZON AIR SEE THE NATIONAL AIRSPACE?." Rutgers Computer & Technology Law Journal 42, no. 1: 162-195. Applied Science & Technology Source, EBSCOhost (accessed March 17, 2018).
- [2] Amazon Technologies "US9718564B1 - Ground-Based Mobile Maintenance Facilities for Unmanned Aerial Vehicles." Google Patents, Google, 1 Aug. 2017, patents.google.com/patent/US9718564B1/en?q=9718564
- [3] "GPS: The Global Positioning System." GPS: The Global Positioning System, www.gps.gov/.
- [4] Dunbar, Brian. "Real-Time Tracking System Uses Ultra-Wideband RF Signals." NASA, NASA, 6 June 2013, www.nasa.gov/centers/johnson/techtransfer/technology/MSC-24184_UWB-Tracking.html.
- [5] Patel, Sanskruti1. 2017. "Integrating Machine Learning Techniques for Big Data Analytics." International Journal Of Advanced Research In Computer Science 8, no. 5: 2760-2763. Applied Science & Technology Source, EBSCOhost (accessed February 5, 2018).
- [6] 2017. "Global Positioning System (GPS)." Funk & Wagnalls New World Encyclopedia 1p. 1. Funk & Wagnalls New World Encyclopedia, EBSCOhost (accessed March 16, 2018).
- [7] Advani, Madhu, and Daniel S. Weile. 2017. "Position and orientation inference via on-board triangulation." Plos ONE 12, no. 6: 1-11. Academic Search Complete, EBSCOhost (accessed March 16, 2018).
- [8] Victor Bahl, Venkat Padmanabhan 2001 "RADAR." Microsoft Research, www.microsoft.com/en-us/research/project/radar/
- [9] Amazon Technologies. "Patent Application Publication." United States Patent and Trademark Office, 22 June 2017,
- [10]"Dedrone RF Sensors." Dedrone, www.dedrone.com/products/hardware/rf-sensors/overview.
- [11]Inc. "Commercial Drone Market to Hit \$17bn by 2024: Global Market Insights, Inc." GlobeNewswire News Room, "GlobeNewswire", 28 Feb. 2018, globenewswire.com/news-release/2018/02/28/1401040/0/en/Commercial-Drone-Market-to-hit-17bn-by-2024-Global-Market-Insights-Inc.html.

Picture References

Figure 1

[12]

<http://www.supplychaindigital.com/technology/llamasoft-and-zipline-working-develop-drones-deliver-crucial-medical-supplies>

Figure 2

[13]

<https://www.youtube.com/watch?v=dp4CyShpjl>

Figure 3

[14]

<https://www.officer.com/tactical/swat/robotic-equipment/product/12073955/uav-direct-tactical-search-rescue-drone-system>

Figure 4

[15]

<https://www.globalsecurity.org/military/library/budget/fy1997/dot-e/airforce/97navstar.html>

Figure 5

[16]

<https://security.stackexchange.com/questions/44326/can-wimax-track-its-users-geo-position>

Figure 6

[17]

<http://www.xmarto.com/helpcenter/?/article/1>

Figure 10

[18]

<https://www.suasnews.com/2016/06/sales-manager-san-francisco/>

Figure 15

[19]

<http://www.ti.com/product/ADC08D1520>

Figure 18

[20]

<https://thepihut.com/products/adafruit-mcp3008-8-channel-10-bit-adc-with-spi-interface>

Figure 20 and 21

[21]

<https://www.minicircuits.com/pdfs/ZX47-40+.pdf>

Figure 28

[22]

<https://www.takealot.com/raspberry-pi-3-model-b-1gb-project-board/PLID41466406>

Figure 34

[23]

<https://blog.stickytickets.com.au/the-ethics-of-the-ticketing-industry/>

Figure 35

[24]

<https://farragut.org/lower-school-students-show-compassion-during-december/>

Figure 36

[25]

<http://bel-india.com/Products.aspx?MId=13&LId=1&link=59>

Table 1

[26]

<http://www.insidegnss.com/node/3125>

Appendices:

Appendix A: Machine Learning Algorithms

```
#trainML.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

signals = pd.read_csv("signalData.csv")

X = signals[['A1', 'A2', 'A3']]
y = signals['X']

X_train, X_test, Y_train, Y_test = train_test_split(X, y)

from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor(n_neighbors=2)
knn.fit(X_train, Y_train)

sum = 0
numDiffs = 0
for i in range(1,1000):
    li = signals.values[i].tolist()[0:3]
    prediction = knn.predict([li])
    if(prediction != signals.values[i][5]):
        sum += abs(prediction-signals.values[i][5])
        numDiffs += 1

print(knn.score(X_test, Y_test))
print(sum/numDiffs)
```

```
-----
#testnn.py
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

def read_dataset():
    df = pd.read_csv("./signalData.csv")
    X = df[df.columns[0:3]].values
    y = df[df.columns[3:5]].values

    return (X, y)

X, Y = read_dataset();
```

```

X, Y = shuffle(X, Y, random_state=1)

train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size=0.20)

print(train_x.shape);
print(train_y.shape)
print(test_x.shape)

learning_rate = 0.1
training_epochs = 10
cost_history = np.empty(shape=[1], dtype=float)
n_dim = X.shape[1]
print("n_dim", n_dim)
n_class = 2
model_path = "./TensorFlow/NN"

n_hidden_1 = 60
n_hidden_2 = 60
n_hidden_3 = 60
n_hidden_4 = 60

x = tf.placeholder(tf.float32, [None, n_dim])
W = tf.Variable(tf.zeros([n_dim, n_class]))
b = tf.Variable(tf.zeros([n_class]))
y_ = tf.placeholder(tf.float32, [None, n_class])

def multilayer_perceptron(x, weights, biases):
    layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1']);
    layer_1 = tf.nn.sigmoid(layer_1)

    layer_2 = tf.add(tf.matmul(layer_1, weights['h2']), biases['b2']);
    layer_2 = tf.nn.sigmoid(layer_2)

    layer_3 = tf.add(tf.matmul(layer_2, weights['h3']), biases['b3']);
    layer_3 = tf.nn.sigmoid(layer_3)

    layer_4 = tf.add(tf.matmul(layer_3, weights['h4']), biases['b4']);
    layer_4 = tf.nn.relu(layer_4)

    out_layer = tf.matmul(layer_4, weights['out']) + biases['out']
    return out_layer

weights = {
    'h1':tf.Variable(tf.truncated_normal([n_dim, n_hidden_1])),
    'h2':tf.Variable(tf.truncated_normal([n_hidden_1, n_hidden_2])),
    'h3':tf.Variable(tf.truncated_normal([n_hidden_2, n_hidden_3])),
    'h4':tf.Variable(tf.truncated_normal([n_hidden_3, n_hidden_4])),
    'out':tf.Variable(tf.truncated_normal([n_hidden_4, n_class]))
}

biases = {
    'b1':tf.Variable(tf.truncated_normal([n_hidden_1])),
    'b2':tf.Variable(tf.truncated_normal([n_hidden_2])),
    'b3':tf.Variable(tf.truncated_normal([n_hidden_3])),

```

```

        'b4':tf.Variable(tf.truncated_normal([n_hidden_4])),
        'out':tf.Variable(tf.truncated_normal([n_class]))
    }

    init = tf.global_variables_initializer()

    saver = tf.train.Saver()

    y = multilayer_perceptron(x, weights, biases)
    print(y.shape)
    cost_function = tf.reduce_mean(tf.cast(tf.abs(tf.subtract(y,y_)), tf.float32))
    training_step =
    tf.train.GradientDescentOptimizer(learning_rate).minimize(cost_function)

    sess = tf.Session()

    File_writer = tf.summary.FileWriter("./TensorFlow/graph", sess.graph)

    sess.run(init)

    mse_history = []
    accuracy_history = []

    for epoch in range(training_epochs):
        sess.run(training_step, feed_dict={x:train_x, y_:train_y})
        cost = sess.run(cost_function, feed_dict={x:train_x, y_:train_y})
        cost_history = np.append(cost_history, cost)

        accuracy = tf.div(tf.reduce_mean(tf.cast(tf.abs(tf.subtract(y,y_)),
        tf.float32)), [120, 360])
        acc = sess.run(accuracy, feed_dict={x:train_x, y_:train_y});
        accuracy_history.append(acc)

        pred_y = sess.run(y, feed_dict={x:test_x})
        mse = tf.reduce_mean(tf.square(pred_y-test_y))
        mse_ = sess.run(mse)
        mse_history.append(mse_)

        print('epoch : ', epoch, ' - ', 'cost: ', cost, " - MSE: ", mse_, "-
Train error(%): ", acc)

    save_path = saver.save(sess, model_path)
    print("Model saved in file: %s" % save_path)
    sess.close()

```

Appendix B: Location Tracking App and GUI

```
<!-- index.html -->
{% load static %}
<link rel='stylesheet' type='text/css' href="{% static 'locations/style.css'
%}"/>
<script>
    var height = window.innerHeight;
    var midPoint = height / 2;
    var radius = 200;
    console.log(height);
    var currentLocation = {
        x: 0,
        y: 0
    }
    var ctxMap;
    var ctxLocation;
    var pointSize = 10;
    var ringRadius = 20;

    window.addEventListener("load", function(event) {
        var cvMap = document.getElementById("cvmap");
        cvMap.width = height;
        cvMap.height = height;
        ctxMap = cvMap.getContext("2d");
        var cvLocation = document.getElementById("cvlocation");
        cvLocation.width = height + 200;
        cvLocation.height = height;
        cvLocation.style.zIndex = "100";
        ctxLocation = cvLocation.getContext("2d");
        var socket = new WebSocket("ws://" + window.location.host + "/chat/");
        socket.onmessage = function(e) {
            currentLocation = JSON.parse(e.data);
            var pointRadius = Math.sqrt(Math.pow(currentLocation.x, 2) +
Math.pow(currentLocation.y, 2));
            if (pointRadius > 0.9 * radius) {
                radius = Math.max(radius * 1.2, pointRadius * 1.2);
                redrawMap();
            } else if (pointRadius < 0.1 * radius) {
                radius = Math.max(radius / 1.2, pointRadius * 1.2);
                redrawMap();
            }
            }

            drawCurLoc(ctxLocation, currentLocation.x, currentLocation.y);

        }

        socket.onopen = function() {
            console.log('opened');
        }
        // Call onopen directly if socket is already open
        if (socket.readyState == WebSocket.OPEN) socket.onopen();
    });
</script>
```

```

        redrawMap();

    });

    function drawCurLoc() {
        ctxLocation.clearRect(0, 0, height + 200, height);
        var pointX = currentLocation.x / radius * midPoint + midPoint;
        var pointY = -currentLocation.y / radius * midPoint + midPoint;
        ctxLocation.fillStyle = "#FF0000";
        ctxLocation.fillRect(pointX - pointSize / 2, pointY - pointSize / 2,
pointSize, pointSize);

        ctxLocation.strokeStyle = "#FF0000";
        ctxLocation.beginPath();
        ctxLocation.moveTo(midPoint, midPoint);
        ctxLocation.lineTo(pointX, pointY);
        ctxLocation.closePath();
        ctxLocation.stroke();

        ctxLocation.font = "20px Arial";
        var text = getPolarLocationStr(currentLocation)
        ctxLocation.fillStyle = "rgba(255,255,255,0.95)"
        ctxLocation.fillRect(pointX + 10, pointY - 15,
ctxLocation.measureText(text).width + 10, 28);
        ctxLocation.fillStyle = "#FF0000";
        ctxLocation.fillText(text, pointX + 15, pointY + 5)

    }

    function getPolarLocationStr() {
        var radius = Math.sqrt(Math.pow(currentLocation.x, 2) +
Math.pow(currentLocation.y, 2))
        var angle =
(Math.atan(currentLocation.y/currentLocation.x)*180/Math.PI+(currentLocation.x<
0?540:360))*360;
        return "(" + radius.toFixed(2) + " in, " + angle.toFixed(2) + "°)"
    }

    function redrawMap() {
        ctxMap.clearRect(0, 0, height, height);
        ctxMap.strokeStyle = "#000000";
        drawKey();
        ctxMap.beginPath();
        ctxMap.moveTo(midPoint, midPoint);
        ctxMap.lineTo(height, midPoint);
        ctxMap.stroke();
        for (var rad = 0; rad < midPoint; rad += ringRadius / radius *
midPoint) {
            drawCircle(ctxMap, rad);
        }
    }

    function drawKey() {
        ctxMap.beginPath();

```



```

        ctxMap.moveTo(20, 20);
        ctxMap.lineTo(20, 30);
        ctxMap.lineTo(20 + ringRadius / radius * midPoint, 30)
        ctxMap.lineTo(20 + ringRadius / radius * midPoint, 20)
        ctxMap.stroke();
        ctxMap.font = "15px Arial";
        ctxMap.fillText(ringRadius + " inches (between rings)", 20, 50);
    }

    function drawCircle(ctx, rad) {
        ctx.beginPath();
        ctx.arc(midPoint, midPoint, rad, 0, 2 * Math.PI);
        ctx.closePath();
        ctx.stroke();
    }
}
</script>
<canvas id="cvmap" class="cv"></canvas>
<canvas id="cvlocation" class="cv"></canvas>

```

```

-----
#consumers.py
from django.http import HttpResponse
from channels.handler import AsgiHandler
from channels import Group
from random import *
from threading import Timer
import json
import time
from locations.pointCollector import PointCollector

pointC = None
def sendData(lastPoint):
    newPoint = pointC.returnLocation(lastPoint)
    Group("chat").send({
        "text": json.dumps({
            'x': newPoint[0],
            'y': newPoint[1]
        })
    })
    t = Timer(0.5, sendData, args=[newPoint])
    t.start()

def startSendingData():
    global pointC
    if(not pointC):
        pointC = PointCollector()
        pointC.startCollecting()
        sendData((0,0))

def ws_add(message):
    print("added")
    # Accept the incoming connection
    message.reply_channel.send({"accept": True})
    # Add them to the chat group

```

```

Group("chat").add(message.reply_channel)
startSendingData()

# Connected to websocket.disconnect
def ws_disconnect(message):
    print("disconnected")
    Group("chat").discard(message.reply_channel)

def ws_message(message):
    # ASGI WebSocket packet-received and send-packet message types
    # both have a "text" key for their textual data.
    print(message.content['text'])
    Group("chat").send({
        "text": "[user] %s" % message.content['text'],
    })

```

```

-----
#pointCollector.py
from threading import Thread, Timer
from random import *
import math
from scipy.spatial import distance
from locations.readADC import ADCReader
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor

```

```

class Node:
    nextNode = None
    radius = 0
    angle = 0
    x = 0
    y = 0
    def __init__(self, radius, angle):
        self.radius = radius
        self.angle = angle
        radians = angle * math.pi / 180
        self.x = radius * math.cos(radians)
        self.y = radius * math.sin(radians)

    def setNext(self, nextNode):
        self.nextNode = nextNode

```

```

class PointCollector():

    def __init__(self):
        self.thread = Thread(target = self.collectPoints)

```

```

self.collecting = False
self.locations = None
self.adcReader = ADCReader()
self.adcReader.startCollecting()
self.locationModel = LocationModel()

def stopCollecting(self):
    self.adcReader.stopCollecting()
    self.collecting = False
    self.thread.join()

def startCollecting(self):
    self.collecting = True
    self.thread.start()

def returnLocation(self, lastLocation):
    nodeList = []
    cursor = self.locations
    for i in range(0,5):
        if(cursor != None):
            nodeList.append(cursor)
            cursor = cursor.nextNode

    sortedList = sorted(nodeList, key=lambda node: distance.euclidean((node.x,
node.y), lastLocation))
    xSum = 0
    ySum = 0
    numElements = min(len(sortedList), 3);
    for i in range(0, numElements):
        xSum += sortedList[i].x
        ySum += sortedList[i].y

    if(numElements == 0):
        return (0,0)
    else :
        return (xSum/numElements, ySum/numElements)

#internal api
def collectPoints(self):
    if(self.collecting):
        amplitudes = self.adcReader.returnSignals()
        print(amplitudes)
        print(self.locationModel.getRadius(amplitudes),
self.locationModel.getAngle(amplitudes))
        newNode = Node(self.locationModel.getRadius(amplitudes),
self.locationModel.getAngle(amplitudes))
        newNode.setNext(self.locations)
        self.locations = newNode
        self.trimList()
        t = Timer(0.1, self.collectPoints)
        t.start()

def trimList(self):
    cursor = self.locations

```

```

    for x in range(0,5):
        if(x == 5 and cursor != None):
            cursor.next = None
        elif (cursor != None):
            cursor = cursor.nextNode

class LocationModel():

    def __init__(self):
        signals = pd.read_csv("signalData.csv")

        X = signals[['A1', 'A2', 'A3']]

        y = signals['distance']
        self.knnDist = KNeighborsRegressor(n_neighbors=9)
        self.knnDist.fit(X, y)

        y = signals['angle']
        self.knnAngle = KNeighborsRegressor(n_neighbors=30)
        self.knnAngle.fit(X, y)

    def getRadius(self, amplitudes):
        return self.knnDist.predict([amplitudes])[0]

    def getAngle(self, amplitudes):
        return (540 - self.knnAngle.predict([amplitudes])[0])%360

-----
#readADC.py
import time
from datetime import datetime
from promise import Promise
from random import *
from threading import Thread

# Import SPI library (for hardware SPI) and MCP3008 library.
import Adafruit_MCP3008

CLK = 18
MISO = 23
MOSI = 24
CS = 25
mcp = Adafruit_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)
scale=3.3/1024

class ADCReader():

    def __init__(self):
        self.collecting = False
        self.signals = [0,0,0]
        self.thread = Thread(target = self.collectSignals)

```

```

        self.pins = [0, 3, 7]

    def stopCollecting(self):
        self.collecting = False
        self.thread.join()
        print('adcreader thread finished')

    def startCollecting(self):
        self.collecting = True
        self.thread.start()

    def returnSignals(self):
        return self.signals

#internal api
    def collectSignals(self):
        while (self.collecting):
            tStart = time.time() * 100000
            maximums = [0,0,0]
            while (int(round(time.time()) * 100000) - tStart < 100000):
                for x in range(0,3):
                    value = round(mcp.read_adc(self.pins[x])*scale,3) #change to read
from adc
                    if (value > maximums[x]):
                        maximums[x] = value
                    time.sleep(0.01);
                self.signals = maximums

```

Appendix C: Data Collection Script

```
#startCollecting.py
# import required libs
import time
import RPi.GPIO as GPIO
from threading import Thread
from locations.readADC import ADCReader
import csv
import math

#adc pins - black:0, white: 3, gray:7

adcReader = ADCReader()
adcReader.startCollecting()

with open('signalData.csv', 'w') as csvfile:
    dataWriter = csv.writer(csvfile, delimiter=',')

    # GPIO.cleanup() #cleaning up in case GPIOs have been preactivated

    # Use BCM GPIO references
    # instead of physical pin numbers
    # GPIO.setmode(GPIO.BCM)

    # be sure you are setting pins accordingly
    # GPIO10,GPI09,GPI011,GPI25
    distancePins = [22,27,17,5]
    anglePins = [6,13,19,26]
    waitTimeDistance = 0.002
    waitTimeRotation = 0.04

    # Set all pins as output
    for pin in distancePins:
        GPIO.setup(pin,GPIO.OUT)
        GPIO.output(pin, False)
    for pin in anglePins:
        GPIO.setup(pin,GPIO.OUT)
        GPIO.output(pin, False)

    #wait some time to start
    time.sleep(2)

    #Full torque
    sequence = [None] * 4
    sequence[0] = [0,0,1,1]
    sequence[1] = [1,0,0,1]
    sequence[2] = [1,1,0,0]
    sequence[3] = [0,1,1,0]

    revSequence = [None] * 4
    revSequence[0] = [0,0,1,1]
    revSequence[1] = [0,1,1,0]
```

```

revSequence[2] = [1,1,0,0]
revSequence[3] = [1,0,0,1]

distanceScalar = 3 #inches per change

maxResolution = 2048
numberOfRotations = 102
angleScalar = 360/numberOfRotations #degrees per change

#distance change once all angles have been measured
#15 feet in 3 inch increments -> 60 increments
for d in range(1, 32): #30 inches
    # move receiver forward 3 inches
    # did 800*36 cyces in 21 5/8 inches -> 3995 cycles in 3 inches
    for i in range(0, 3995): #3995
        for pin in range(0, 4):
            xpin = distancePins[pin]
            GPIO.output(xpin, sequence[i%4][pin])
            time.sleep(waitTimeDistance)

#rotate receiver
for a in range(1, numberOfRotations+1):
    if (d%2 == 1) :
        for i in range(0, math.floor(maxResolution/numberOfRotations)):
            for pin in range(0, 4):
                xpin = anglePins[pin]
                GPIO.output(xpin, sequence[i%4][pin])
                time.sleep(waitTimeRotation)
    else :
        for i in range(0, math.floor(maxResolution/numberOfRotations)):
            for pin in range(0, 4):
                xpin = anglePins[pin]
                GPIO.output(xpin, revSequence[i%4][pin])
                time.sleep(waitTimeRotation)

#get data
angle = round(a*angleScalar,3)
if (d%2 == 0):
    angle = round(360 - a*angleScalar)
radius = 8+d*distanceScalar

for i in range(0, 1):
    time.sleep(0.5)
    signals = adcReader.returnSignals()
    # print("Signals: " + str(signals) + "          Radius:"+str(radius)+"
Angle:" + str(angle))
    dataWriter.writerow(signals + [radius, angle])
    csvfile.flush()

```


Appendix D: Table of Components and Cost

Type of component	Component designation	Amount purchased	Price per component	Total price
Attenuator	VAT-10+	3	\$13.95	\$41.85
Amplifier	ZX60-V62+	3	\$49.95	\$149.85
Oscillator	ZX95-928CA-S+	3	\$51.95	\$155.85
Power Detector	ZX47-40-S+	3	\$89.95	\$269.85
PCB Board		6	\$1.63	\$9.80
SMA Connectors		10	\$2.10	\$21.00
SMA to BNC		2	\$9.50	\$19
Mixer	SYM-63LH	10	\$16.44	\$164.37
SMA cables	SCA49086-06	10	\$15.33	\$153.25
Legos		1	\$14.13	\$14.13
USB cables		3	\$6.99	\$20.97
Butt Connector (100pc)		1	\$9.29	\$9.29
Stepper Motor Driver	TB6612	2	\$7.57	\$15.14
Stepper Motor	Nema 17	1	\$13.99	\$13.99
Breadboard	ALLUS BB-009	1	\$7.88	\$7.88
Balsa wood strip		1	\$16.46	\$16.46
Stepper Motor	28BYJ-48 ULN2003	1	\$12.99	\$12.99
Timing Belt	BIQU GT2	1	\$12.98	\$12.98
Timing Belt pulley		1	\$9.89	\$9.89