Drone Localization and Jamming Assembly

Nicholas Cauley
Ryan Crowley
Mara Pranter

Advisor: Professor Kaveh Pahlavan

December 10th, 2020

Abstract

With the cost of drones decreasing and their presence in the world increasing, there needs to be a way to control their presence in restricted airspace. To solve this problem, our team designed an assembly to both locate an unwanted drone or drone pilot and jam the communication between drone and the drone controller. First the location was discovered using several SDRs to record signal strength at different locations. This would find either the drone or drone controller when other signals were not present on the same bandwidth. The SDR signal data was then processed to get the signal's received signal strength (RSS) These results were then mathematically modeled with the path loss model, which turned the RSS values into a distance in meters. Then the recursive least squares algorithm transformed the distances from each SDR into a position in the space that corresponds to the drone or drone controller. Finally, the Cramer Rao Lower Bound was used to estimate the amount of error possible in the position estimation. The jamming was achieved using a microcontroller to broadcast white Gaussian noise at the same frequency as the drone communication to render that communication useless. The location determining system was found to be within a meter of error on the xy plane, however, it had more error on the z axis as the SDRs were not at different enough z positions to sense the z position as accurately. The jamming system was designed and built using analog RF components, fed by voltages from the microcontroller. The effectiveness of the jamming system could not be fully tested as strict COVID-19 protocols limited access to diagnostic equipment.

Acknowledgements

This project has three members; Nick Cauley, Ryan Crowley, and Mara Pranter. We would first like to thank our advisor, Professor Kaveh Pahlavan, for his guidance to create a project our team is proud of as well as advice that will aid us long beyond the scope of this project. We would also like to thank WPI’s Electronics Technician, William Appleyard, for all of his help in the construction of our project and the acquisition of components.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ACRONYMS

CRLB	Cramer Rao Lower Bound
DLVA	Detector Logarithmic Video Amplifier
FAA	Federal Aviation Administration
RF	Radio Frequency
RLS	Recursive Least Squares
RSS	Received Signal Strength
SDR	Software Defined Radio
UAV	Unmanned Aerial Vehicle
VCO	Voltage Controlled Oscillator
WPI	Worcester Polytechnic Institute

1. Introduction

Drones, also known as unmanned aerial vehicles (UAVs), are becoming more and more accessible as the market for them grows. In 2019, The Federal Aviation Administration (FAA) estimated that there were over 1.25 million drones flying in America, and they predict this number only will increase.[1] This market growth has led to drones in general becoming cheaper to buy, as well as the expected capabilities of them to rise. This combination of lower cost, higher performance and availability is good for the consumer, but poses a massive risk as an increasing number of pilots, out of either negligence or maliciousness, are flying in restricted airspaces around the world. This is a massive safety concern, as it could lead to drones interfering with aircraft, potentially causing deadly crashes.[2] In 2017, a flight near Jean Lesage Airport in Canada collided with a drone.[3] While tragedy was avoided in this incident, the potential for a disaster still looms. Due to the immense risk, airports likely shut down if a drone is flying nearby, possibly causing millions in damages for the airport and its travelers. In 2018, Gatwick Airport in the United Kingdom was shut down for a period of three days. It is estimated that this caused over $1.8 million in damages to the airport, not including the damages of passengers whose flights were delayed or canceled.[4] There is a great need to create a system to prevent all of this damage from occurring.

1.1. Project Description

All of this damage can be easily caused with any consumer available drone. Our project is a system that can be implemented in airports, among other areas where drones should not be, to prevent the flying of drones in an area and locate the position of the pilot. Our project is an assembly of systems to locate a drone or its controller, and then jam the signals, preventing the drone from being operated.In the past, a WPI student team created and tested an Adversary UAV Localization system.[5] Their work was used as a starting point for our project. The testing done in the previous project used a cellphone as a substitute for the drone signal. We improved their testing methodology by using a drone controller, providing a more realistic scenario. The biggest improvement on the past project is our addition of a jamming system, allowing the user of the assembly to prevent the drone from flying if it is located in restricted airspace. Our project's localization system finds the location of the drone or its controller by reading the signal strength with several receivers. Once the location of the drone is established, if it is determined that it should be jammed, the jamming system will generate a signal that overwhelms the drone and its controller's communication, causing the drone to cease flight. A block diagram and general overview diagram can be found in Section 3, System Architecture in Figures 3.1 and 3.2.

1.2. Report Outline

This report is broken up into several main sections to aid the reader in understanding the various components and parts of our project. Section 2, Theoretical Background discusses the knowledge and theoretical information required for the reader to understand the detailed workings of the system. Section 3, System Architecture lays out the specific configurations of hardware and software that are used to receive, process, and jam drone signals.
Section 4, Results and Discussion explains the processes used to test the system, and then analyzes the results of the tests. Section 5, Conclusions and Future Recommendations goes over the analyzed results, discussing what about our system worked and where it can be improved. Section 5 also gives recommendations for future advancements of this project.

2. Theoretical Background

In this section, the theoretical background needed to create and understand this project is discussed. The first subsection discusses the theory behind localization of signals, specifically how they are received, processed, and the various equations, algorithms, and models that go along with it. The second subsection will provide background on the processes of jamming signals, from both a hardware and software perspective.

2.1. Localization

In order to best locate either the drone or the drone pilot there are four steps; finding the received signal strength (RSS), the path loss model, recursive least squares (RLS) algorithm, and error analysis. The received signal strength is put into the path loss model which calculates the distance between an access point (SDR) and the object (drone/drone pilot). Once there are calculated distances from each access point to the object, the RLS algorithm is used to position where in a 3D space the object is. Finally, the amount of error possible is estimated with the Cramer Rao Lower Bound (CRLB).

2.1.1. Signal Processing

Signal processing gives the RSS to begin the process of mathematically determining the location of the drone or drone pilot. In order to process the signals, one must first read the signal with a software-defined radio (SDR), The SDR itself can not process the signals; it depends on software from a separate device to process signals. The different levels of software that can process signals for SDRs are host, embedded, and programmable logic programs. For the purpose of the localization assembly, GNU Radio, an open source software that runs on the host level was selected due to its high level of documentation in this and similar use cases. GNU Radio allows the user to process signals by making a flowgraph of different signal processing blocks. This flowgraph then generates a Python script that can be run to process live signals from the SDR. Since GNU Radio is a host level software, it must be run on a computer connected to the SDR over an ethernet cord. An important part of the system is having several SDRs in different positions, all reading signals that get processed into RSS values by the host computer. To connect the several SDRs to one computer, a network switch can be used to join the SDRs together before connecting them with the computer.

2.1.2 Localization Setup

The physical setup of the localization system is heavily based on the system created by the Adversary UAV Localization project's team.[5] Their configuration can be described as several SDRs placed in different locations, all connected to a host computer. This setup allows the SDRs to get different RSS values from the same signal, allowing for a localization algorithm to locate the signal source. Our configuration is similar, pictured later in Section 4.1, Figure 4.1. The key part in both configurations is different locations of the SDRs, which allows for the aforementioned different RSS values. The more varied their positions, the more accurate our localization can be.

2.1.3. Path Loss Model

The path loss model allows for the RSS values found by the SDRs to be converted into a distance in meters. The equation is as follows:

$$P_r = P_0 - 10\alpha log_{10}(\frac{r}{r_0}) + X(\sigma) \tag{2.1}$$

Where P_r is the RSS between the access point and object, P_0 is the RSS at a distance r_0, α is the distance-power gradient, r is the distance between the access point and object, r_0 is the reference distance, and X(σ) represents the shadow fading. When the path loss is graphed these variables are more apparent.

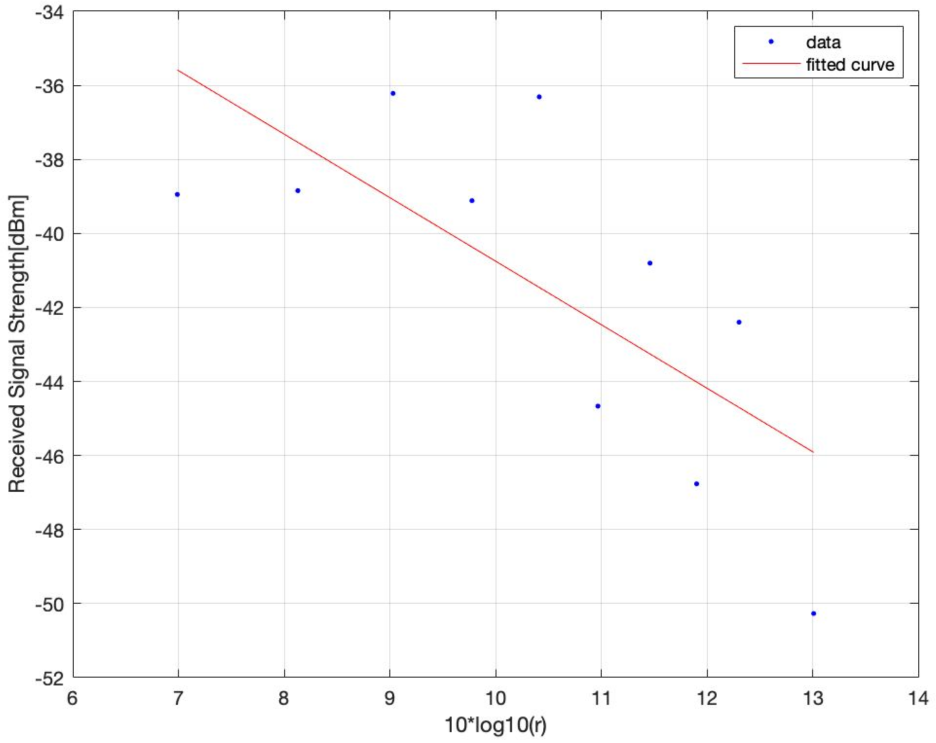


Figure 2.1 Path Loss Model

In the graph above, P_0 is the Y intercept, α is the slope of the line, X(σ) is the variance of the points. The points represent the distance (r) and the RSS (P_R). Using this equation and graphing the α and σ are able to be easily calculated then later used in CRLB for error estimation. In order to calculate the distance (r) quickly, it is assumed that X(σ) is negligible and α is 2. The P_0 at r=1 is first calculated, this then allows for the distance to be calculated using:

$$r = 10^{(\frac{P_0 - P_r}{20})} \tag{2.2}$$

This equation let the team quickly calculate distances from the access points to the object to put into the RLS algorithm.

2.1.4. RLS Algorithm

The RLS algorithm takes each distance from every access point to the object and the position of each access point to determine the position of the object. This is done by creating a circle around the access point with a radius of the distance from the access point to the object. Where all of the circles from each access point overlay is where the object is. This would work perfectly if all of the distances from the access point to the object are completely accurate, unfortunately in the real world as well as this application these distances are not entirely accurate. This means there needs to be an estimation of the position, this is done with the recursive least squares (RLS) algorithm. This is a method by which to minimize error by guessing a best fit and seeing how great the error squared is. Then the position where this error is the lowest is where the object is assumed to be. This can be demonstrated in two dimensions in the figure below.

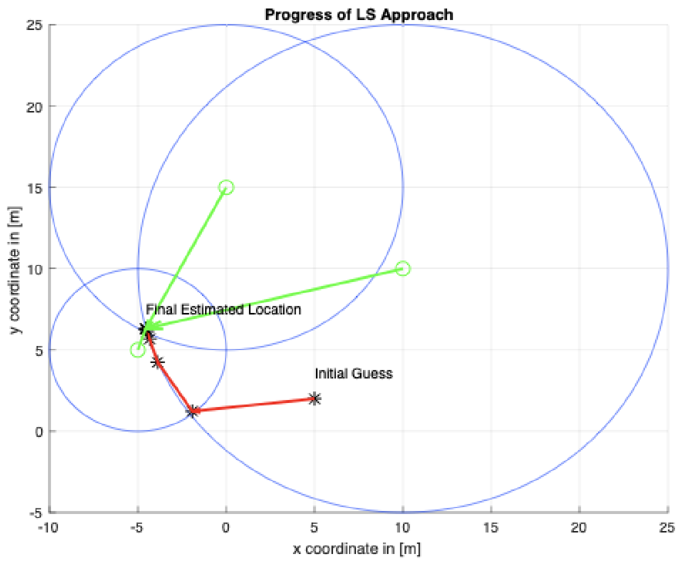


Figure 2.2 RLS Algorithm in a 2D Space

In this team's application the algorithm is moved into a 3D space, this means the circles are spheres and there needs to be four access points in order to estimate the object's z position.

2.1.5. CRLB Error Analysis in 2D

To analyze the potential error of the localization system's configuration, the Cramer Rao Lower Bound (CRLB) was calculated. CRLB is the theoretical minimum variance of a system. In the case of a localization system, it gives the standard deviation of localization error at a point in space. The lower the CRLB value is, the less likely it is for the system to have high error at that point. Calculating the CRLB for an entire space of points is important to get an estimate on the error of the system configuration.

The equation for the CRLB in this localization system for 2D is derived from equation 2.1, the path loss model. First the distance for each point from the SDR is calculated. This is repeated for each SDR in the system. The equation used is:

$$r = \sqrt{(x_{point} - x_{SDR})^2 + (y_{point} - y_{SDR})^2} \qquad (2.3)^6$$

The distance power gradient, α, is a constant calculated in the path loss model shown in section 2.1.2. It is multiplied by another distance based calculation that uses the distance, r, calculated in equation 2.3. This entire calculation is repeated for each dimension, twice in the case of a 2D space, for each SDR. The entire equation is:

$$H[1] = -10 * \alpha * (x_{point} - x_{SDR}) / r^2 \tag{2.4}$$

$$H[2] = -10 * \alpha * (y_{point} - y_{SDR}) / r^2 \tag{2.5}$$

These two arrays are then combined into a single matrix, H. The covariance matrix is then created using H and σ, the standard deviation of shadow fading found in the path loss model. The equation for this is:

$$Cov = \sigma^2 * (H' * H)^{-1} \tag{2.6}$$

The covariance matrix now contains the covariance of each point's location error, σ^2_x and σ^2_y. To find the standard deviation of the location error for a point, the CRLB, the following equation is used:

$$\sigma_{point} = \sqrt{\sigma^2_x + \sigma^2_y} \tag{2.7}$$[5]

2.1.6. CRLB Error Analysis in 3D

While the previous set of equations finds the CRLB in a 2D space, the localization system being implemented needs to work in 3D space. To accomplish this, another dimension was added to the equations. The 3D version of equation 2.3 is:

$$r = \sqrt{(x_{point} - x_{SDR})^2 + (y_{point} - y_{SDR})^2 + (z_{point} - z_{SDR})^2} \tag{2.8}$$

Similarly, the 3D versions of the H matrix equations are as follows:

$$H[1] = -10 * \alpha * (x_{point} - x_{SDR}) / r^2 \tag{2.9}$$

$$H[2] = -10 * \alpha * (y_{point} - y_{SDR}) / r^2 \tag{2.10}$$

$$H[3] = -10 * \alpha * (z_{point} - z_{SDR}) / r^2 \tag{2.11}$$

The covariance matrix equation, 2.6, still works in 3D space with no modifications. The result does slightly change however. It still contains the covariances of each point's location error, but has an extra dimension represented as σ^2_z. Equation 2.7 changes slightly in 3D space, as it accounts for the new dimension in the covariance matrix, as shown below:

$$\sigma_{point} = \sqrt{\sigma^2_x + \sigma^2_y + \sigma^2_z} \tag{2.12}$$

The result of these 3D equations is the standard deviation of location error in 3D space. To find the CRLB for every point in 3D space, a MATLAB script using the 3D CRLB equations was used to automate the process.

2.2. Jamming

2.2.1 Frequency Considerations

As one of the major objectives of the project is to jam drones within an area for safety reasons, it is important that the project identifies which drone frequencies necessary to jam. Due to the high potential for other RF signals coming in and out of the drone jammer's range, it is important that specific RF frequencies be defined which we plan to jam. Otherwise, we could potentially interfere with other RF signals, which poses both a safety and legality risk. Table 2.1, shown below, has a wide variety of popular drone brands and the frequency (or frequencies) used for communication between the drone and its controller.

Brand	Frequency
DJI Phantom	2.4 / 5.8 GHz
Futaba	2.4 GHz
Spektrum	2.4 GHz
JR	2.4 GHz
Hitec	2.4 GHz
Graupner	2.4 GHz
Yuneec	2.4 GHz
Parrot AR2	2.4 GHz
Immersion	433 MHz

Table 2.1: Frequencies used by Popular Drone Brands[7]

One key takeaway from Table 1 is that most common drone brands communicate with a frequency of 2.4 GHz, with 5.8 GHz and 433 MHz being less common frequencies used. This would imply that jamming the 2.4 GHz frequency would jam the majority of commercial drones. However, the more advanced drones (which tend to be a greater security threat) also send a video to the controller so that the user can see the surrounding area from the drone's perspective. In some cases, it may be ideal to not only jam the signals from the controller to the drone, but also the video signals from the drone to the controller so the user doesn't gain sensitive information even after the drone is jammed. Table 2.2, shown below, shows which frequencies common drone brands use to send video signals back to the controller.

Brand	Frequency
DJI	2.4 GHz
Immersion	2.4 GHz

Yuneec	5.8 GHz
Connex	5.8 GHz
Boscam	5.8 GHz

Table 2.2: Frequencies used for Video by Common Drone Brands[7]

One major outcome from Table 2.2 is that the video link is often done through the 5.8 GHz band or 2.4 GHz band. Therefore, based on the results from these two tables, as well as some further research completed on drone brands and their frequencies, this project has determined that jamming the 2.4 GHz band will be sufficient for this application. A simple bandpass filter for the 2.4-2.5 GHz band will allow for the jamming assembly to only process signals within this band. This will be further detailed in Chapter 3, System Architecture.

2.2.2 Jamming Scenario

In order to understand the hardware and software that is necessary for a drone jamming assembly, it is important to first design and analyze a block diagram which shows each step in the process from the reception of a signal to the transmission of a high power signal meant to jam the drone's communication with its controller. As identified in Section 2.2.1 the common drone frequencies which will be taken into account in the drone jamming assembly is 2.4 GHz. Shown below in Figure 2.1 is a top-level block diagram of how the RF drone jamming assembly will be designed to operate.

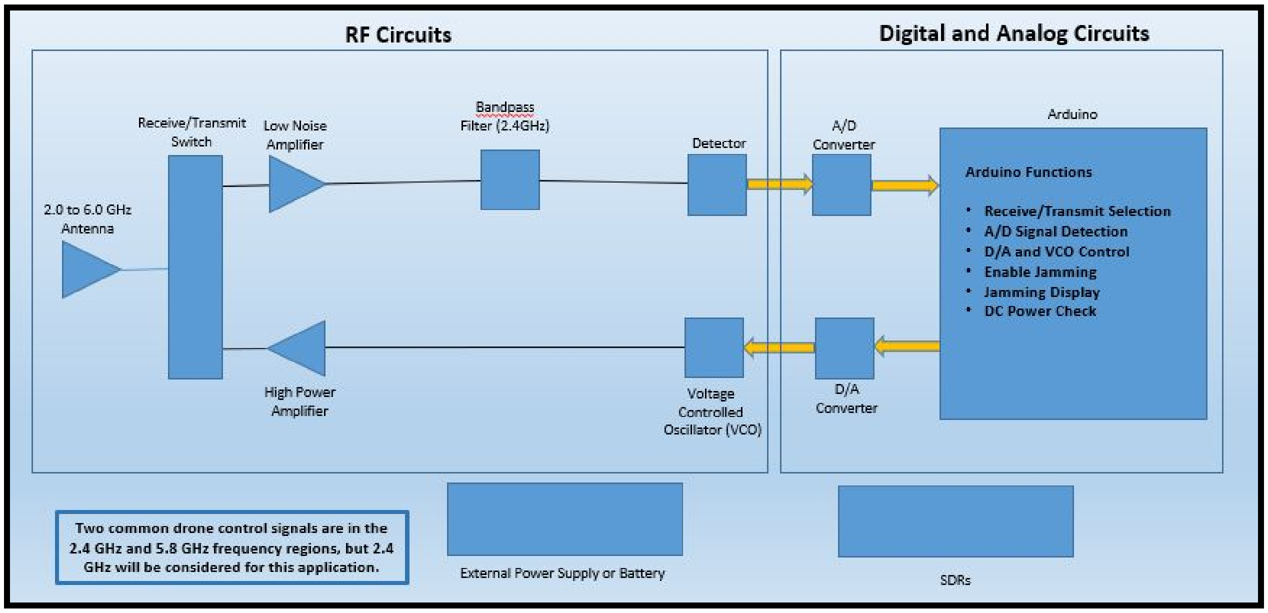


Figure 2.3: Drone Jamming Block Diagram

The first part of the drone jamming assembly, the 2.0 to 6.0 GHz antenna, is used to both receive and transmit RF signals. The reason for the 2.0 to 6.0 GHz range is because this range covers all received signals to be analyzed.It is important that the antenna be able to perform both signal reception and transmission because when it is determined that a received signal is within either of the frequency ranges that needs to be jammed, a high power signal of the same frequency will be emitted to confuse the drone. In order for this to be accomplished, there needs to be a switch which transitions between the receive and transmit operation of the antenna. This switch will be controlled by the microcontroller.

When the microcontroller is used in the receive mode, all signals received through the antenna will be sent to a Low Noise Amplifier. The purpose of this amplifier in the block diagram is to amplify any low-power signals while maintaining a negligent change to its signal-to-noise ratio. This electrical component amplifies the power of the signal while adding minimal amounts of noise to the system. This allows for received signals that are low in power to be more easily processed and analyzed at later components of the RF jamming assembly in Figure 2.1.

A Detector Logarithmic Video Amplifier (DLVA) will be the next electrical component that takes in the RF signal. A DLVA converts the energy of the RF signal into a Direct Current (DC) Voltage (the detector portion), and then amplifies the signal through an amplifier with a logarithmic transfer function (the logarithmic video amplifier portion). By relating the RF power of each bandpass filter to a specific DC Voltage, the microcontroller can then determine if the signals are significant enough to be an actual drone signal as opposed to noise. This part in the process is crucial, as it links the RF signals to information that the microcontroller can process.

The electrical component which will receive this outputted Voltage is a Voltage Controlled Oscillator (VCO). A VCO takes a specific DC Voltage input and produces an RF signal output that is determined by the Voltage of the input. The purpose of this electrical component is to allow for the microcontroller to send out a 2.4 GHz signal that will eventually be transmitted through the antenna. This ensures that the signal sent out matches the frequency of the signal it wishes to jam.

Lastly, a High Power Amplifier will amplify the power of the RF signal outputted by the VCO. This electrical component ensures that the signal is of a high enough power to adequately jam the drone's signals. If the power of the signal outputted by the jamming system is not significant enough, then the drone will still be able to communicate with its host controller without significant interruption. If the RF transmit/receive switch is turned to transmit mode, this high power signal will then be sent out to the surrounding environment through the antenna. This process will be followed whenever the jamming assembly is powered on.

2.2.3 Software Considerations

The barrage jamming technique was selected to be the method by which to jam the drone communication because it is able to fill an entire band of communication with noise making communication impossible. The barrage jamming technique is particularly useful as many drones will change frequencies as communication drops and some have complex linking procedures.[8] The one possible problem with barrage jamming is that as the signal is spread over a band it loses strength. This is able to be countered by making sure the power is greater from the jammer than the controller. This can be done by methods as simple as having the jammer closer to the drone than the controller. The equation that describes barrage jamming output is as follows:

$$w[n] = \sqrt{\frac{P}{2}x[n]} + j\sqrt{\frac{P}{2}}y[n] \tag{2.13}$$

Where P is the effective radiated power in watts and x[n] and y[n] are uncorrelated sequences of zero-mean Gaussian random variables with unit variance. The real portion of this equation can be generated as graphed below, to output from the microcontroller to jam communications.

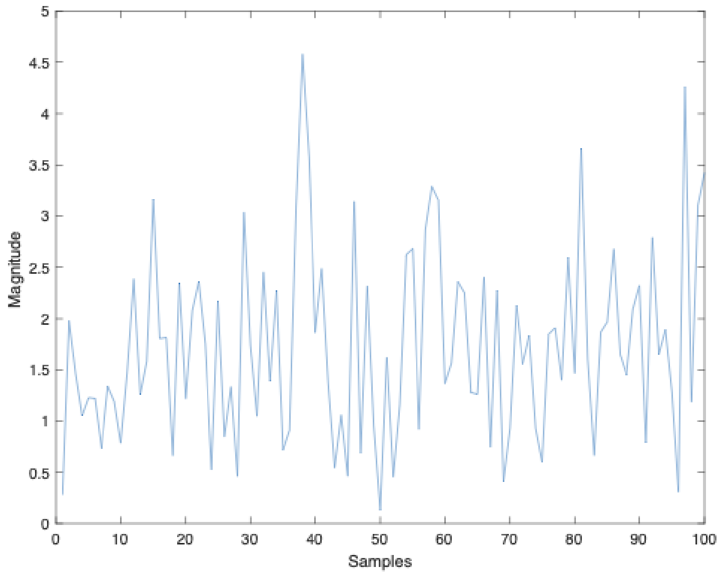


Figure 2.4: Barrage Jamming Noise

3. System Architecture

The overall system is broken into two main parts. First there is the location system which is responsible for finding either the drone or drone pilot. This is done with software defined radios (SDRs) which return the received signal strength which can then be mathematically transformed into a position in a 3D space. The second part of the system is the jamming device which uses barrage jamming software on a microcontroller and RF hardware to disrupt the signal between the drone controller and the drone. These systems are demonstrated visually below.

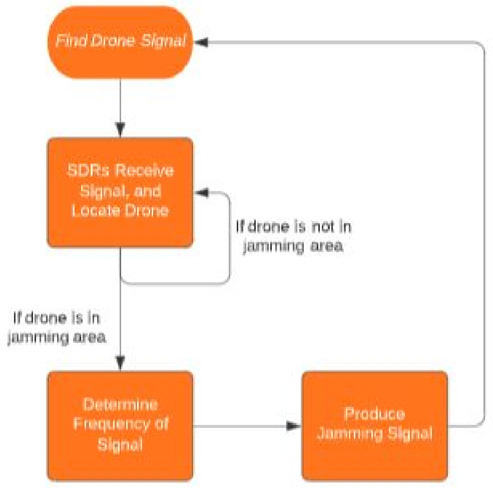

Find Drone Signal
SDRs Receive Signal, and Locate Drone
If drone is not in jamming area
If drone is in jamming area
Determine Frequency of Signal
Produce Jamming Signal

Figure 3.1 Assembly Block Diagram

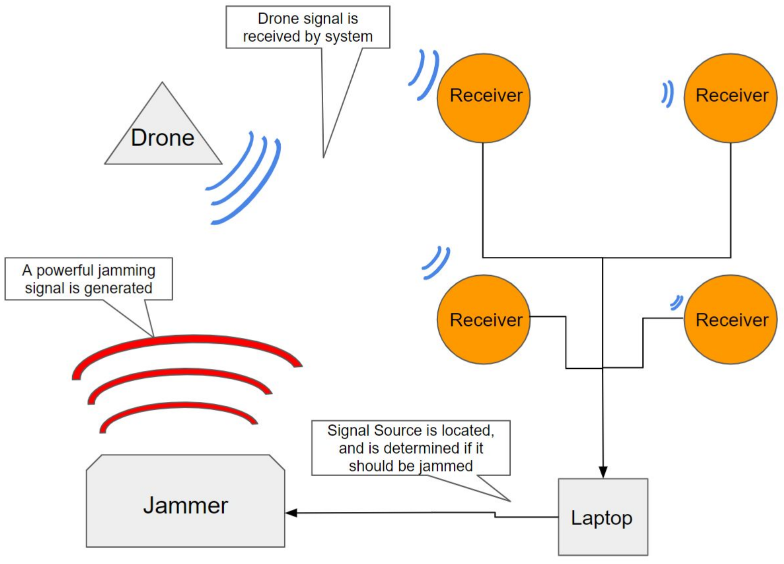

Drone signal is received by system
Drone
Receiver
Receiver
A powerful jamming signal is generated
Receiver
Receiver
Signal Source is located, and is determined if it should be jammed
Jammer
Laptop

Figure 3.2 General Systems Overview

3.1. Hardware Modules

The categories of hardware used in the assembly can be grouped into three sections: localization, the microcontroller, and jamming. The localization hardware group contains the components used in receiving and processing the signals, and locating their source. The microcontroller links the localization and jamming systems together, feeding the jamming system appropriate voltages to match the signals frequency. The jamming group of components includes the RF hardware that makes up the jamming signal generating circuit.

3.1.1. Localization Components

The localization portion of the system uses various hardware to read and process the hardware. Figure 2.1 below shows an overview of the components and connections of the hardware used in the localization system.

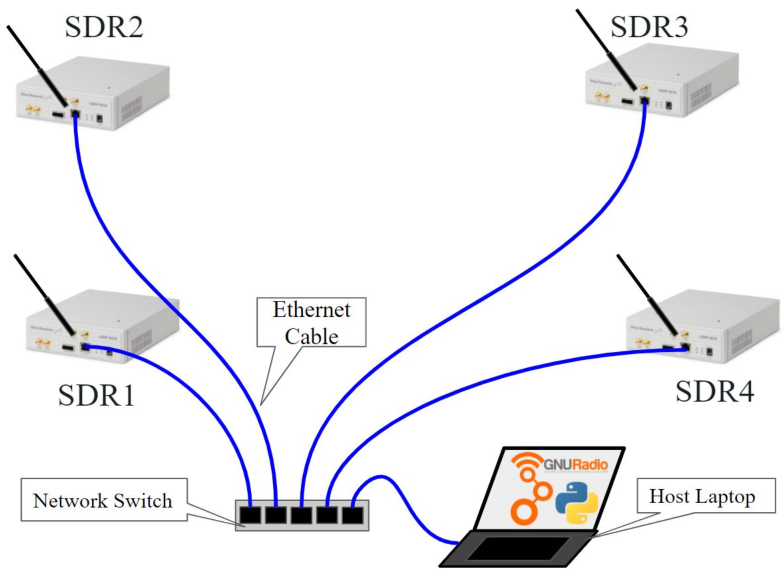


Figure 3.3 Localization Hardware Overview Diagram (power cords not shown)

The system is made up of 4 radios connected by a network switch by ethernet cables. A laptop that is running signal processing software is also connected to the switch, Not pictured in the diagram are the power strip and extension cords used to get power to the radios and switch.

3.1.2. Microcontroller

A microcontroller, allows this project to best accomplish the digital processing and creation of signals. This is because microcontrollers are fast and can easily be integrated with other circuitry. The microcontroller will be responsible for the following:

- ❖ Switching the antenna between receiving and transmitting
- ❖ Taking voltages from the RF circuits and converting it into digital data
- ❖ Filter noise
- ❖ Create a jamming signal and send the signal to the transmission portion of the RF circuit

The details of how these tasks are tackled in the coding of the microcontroller is explored later in the Section 3.3 of this paper. The microcontroller being used is the Arduino MKR 1010. Below is a diagram of the board’s main components.

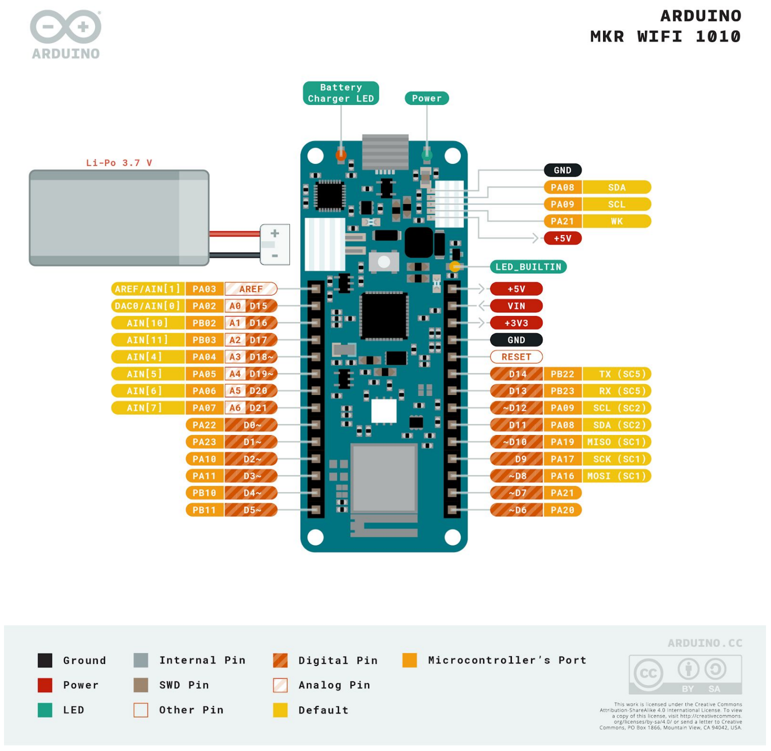


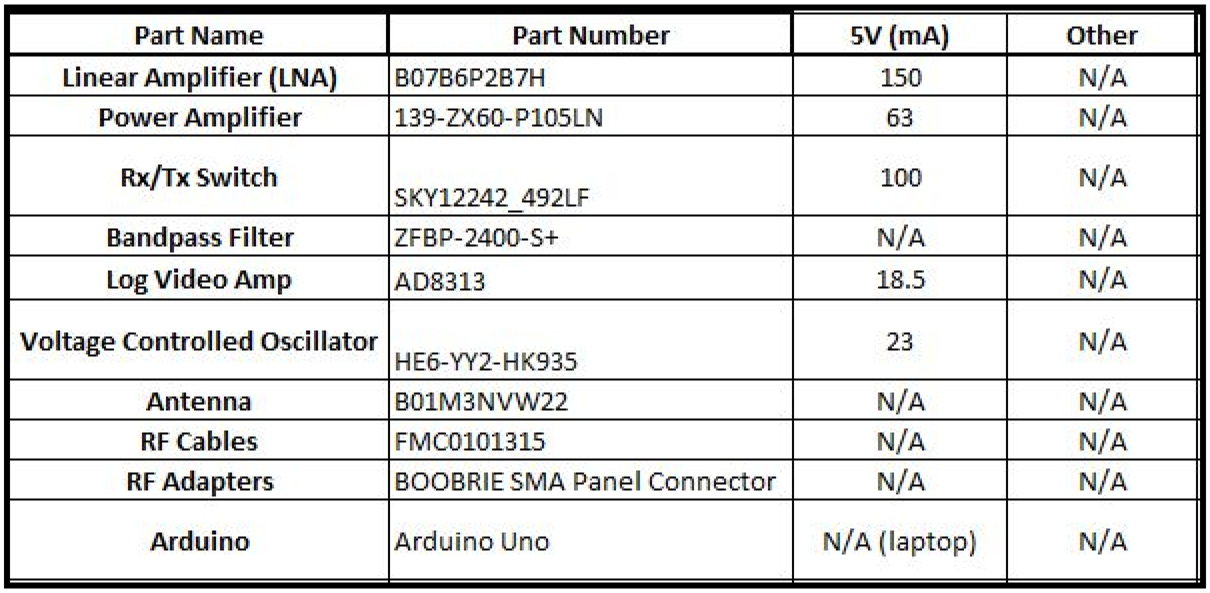
Figure 3.4 Main Components Arduino MKR 1010[9]

The analog signal from the RF drone detection circuitry will come in through pin A0. The outcoming barrage jamming signal will come out of DAC0.

The microcontroller in this specific application has many necessary functions in order to properly determine the significance of all input signals, as well as how to react in all cases. At the input of the microcontroller is an Analog to Digital (A/D) converter, which takes the output of the DLVA and relates Voltage levels to specific binary representations. This allows for the microcontroller to perform the following functions: receive and transmit switch selection, filter switch selection, A/D signal detection, Digital to Analog (D/A) control, enabling jamming, power to the display, overall system battery percentage, and many others. If it has been determined that a signal is significant enough in power level to be jammed, this information will be received by the D/A converter, which takes a digital input from the microcontroller and produces an output Voltage.

3.1.3. Jamming Components

All of the components discussed from Section 2.2.2 need to be carefully ordered based on our project's required specifications. One of the important concerns for the jammer assembly is the Voltage and current that each active component draws. This has an impact on which external power supplies are necessary to power the entire system for an adequate amount of time. Table 3.1 below shows the Voltage and current levels required for operation by all components.

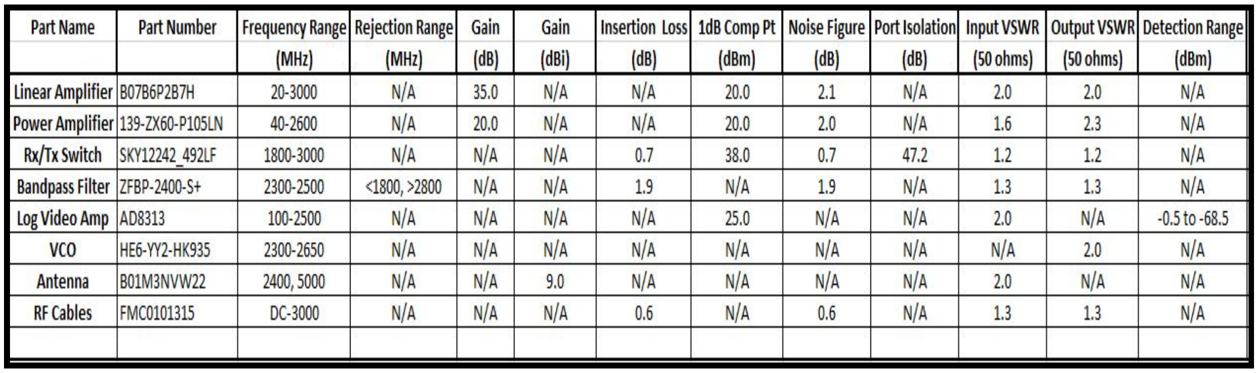


Part Name	Part Number	5V (mA)	Other
Linear Amplifier (LNA)	B07B6P2B7H	150	N/A
Power Amplifier	139-ZX60-P105LN	63	N/A
Rx/Tx Switch	SKY12242_492LF	100	N/A
Bandpass Filter	ZFBP-2400-S+	N/A	N/A
Log Video Amp	AD8313	18.5	N/A
Voltage Controlled Oscillator	HE6-YY2-HK935	23	N/A
Antenna	B01M3NVW22	N/A	N/A
RF Cables	FMC0101315	N/A	N/A
RF Adapters	BOOBRIE SMA Panel Connector	N/A	N/A
Arduino	Arduino Uno	N/A (laptop)	N/A

Table 3.1: Component Voltage and Current Requirements

All ordered components can be powered by a 5V DC supply, which gives us the availability to power the entire circuit on one 5V power source. The total current required by all components (excluding the Arduino) is 354.5 mA, which can be powered by a USB cord connected to a laptop, which supplies 5V at 500 mA. The final conclusion from Table 3.1 is that all RF components can be powered by a single external laptop, with one USB cord powering and controlling the Arduino, and another powering all of the active RF components.

Another consideration with respect to the ordered RF components is the RF specifications of each part. Table 3.1 highlights each of the main specifications of each component.



Part Name	Part Number	Frequency Range	Rejection Range	Gain	Gain	Insertion Loss	1dB Comp Pt	Noise Figure	Port Isolation	Input VSWR	Output VSWR	Detection Range
		(MHz)	(MHz)	(dB)	(dBi)	(dB)	(dBm)	(dB)	(dB)	(50 ohms)	(50 ohms)	(dBm)
Linear Amplifier	B07B6P2B7H	20-3000	N/A	35.0	N/A	N/A	20.0	2.1	N/A	2.0	2.0	N/A
Power Amplifier	139-ZX60-P105LN	40-2600	N/A	20.0	N/A	N/A	20.0	2.0	N/A	1.6	2.3	N/A
Rx/Tx Switch	SKY12242_492LF	1800-3000	N/A	N/A	N/A	0.7	38.0	0.7	47.2	1.2	1.2	N/A
Bandpass Filter	ZFBP-2400-S+	2300-2500	<1800, >2800	N/A	N/A	1.9	N/A	1.9	N/A	1.3	1.3	N/A
Log Video Amp	AD8313	100-2500	N/A	N/A	N/A	N/A	25.0	N/A	N/A	2.0	N/A	-0.5 to -68.5
VCO	HE6-YY2-HK935	2300-2650	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	2.0	N/A
Antenna	B01M3NVW22	2400, 5000	N/A	N/A	9.0	N/A	N/A	N/A	N/A	2.0	N/A	N/A
RF Cables	FMC0101315	DC-3000	N/A	N/A	N/A	0.6	N/A	0.6	N/A	1.3	1.3	N/A

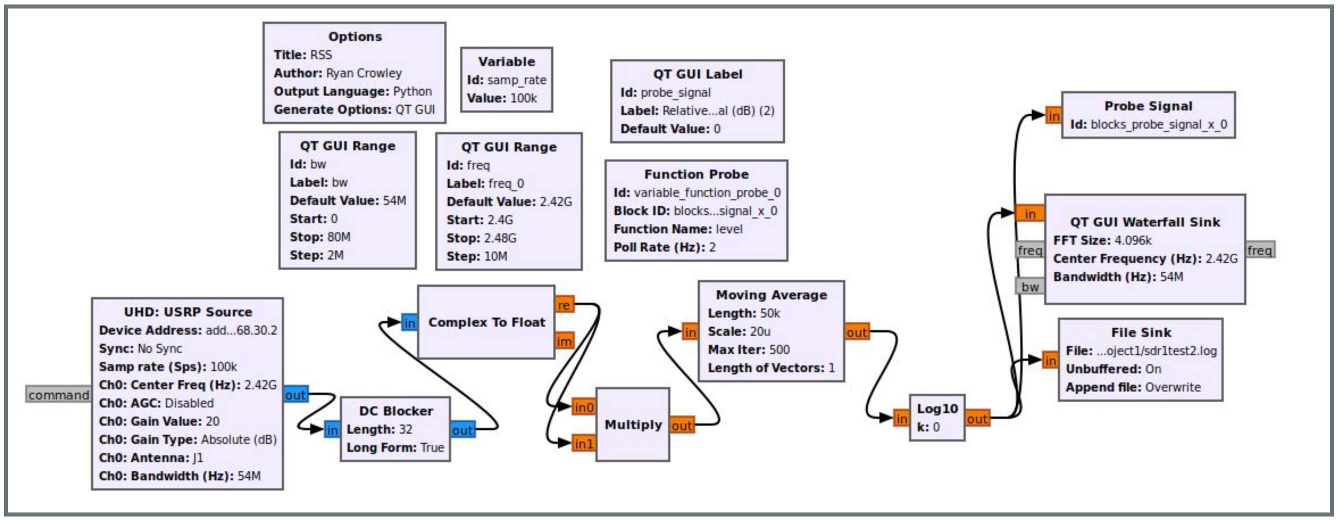
Table 3.2: Component Specifications

The one common specification of all RF components in the jamming assembly is the frequency range of each. As shown in Table 3.2 above, all components meet the desired jamming range of 2400 MHz to 2500 MHz. This means that all components throughout the assembly will be able to properly operate within the desired band.

3.2. Software Modules

3.2.1. Receiving and Processing

As mentioned in section 2.1.1, to receive and process signals using SDRs a software is needed. The software used in this system is GNU Radio, ran on a host computer connected to the radios over ethernet. The GNU Radio flowgraph shown below in Figure 3.3 uses the SDR as a receiver tuned to a frequency range of about 2.4 - 2.45 GHz. It then takes the incoming signals, squares their magnitude, and uses that value in a logarithmic function to find the RSS. The flowgraph then writes the RSS to a log file, resulting in millions of data points in just a few seconds. This flowgraph was then repeated for the three remaining SDRs, each with their own log file. After generating and running the script from the flowgraph, a separate Python script was created to average the millions of RSS values for each SDR.



Options
Title: RSS
Author: Ryan Crowley
Output Language: Python
Generate Options: QT GUI

Variable
Id: samp_rate
Value: 100k

QT GUI Label
Id: probe_signal
Label: Relative...al (dB) (2)
Default Value: 0

Probe Signal
Id: blocks_probe_signal_x_0

QT GUI Range
Id: bw
Label: bw
Default Value: 54M
Start: 0
Stop: 80M
Step: 2M

QT GUI Range
Id: freq
Label: freq_0
Default Value: 2.42G
Start: 2.4G
Stop: 2.48G
Step: 10M

Function Probe
Id: variable_function_probe_0
Block ID: blocks...signal_x_0
Function Name: level
Poll Rate (Hz): 2

QT GUI Waterfall Sink
FFT Size: 4.096k
Center Frequency (Hz): 2.42G
Bandwidth (Hz): 54M

UHD: USRP Source
Device Address: add...68.30.2
Sync: No Sync
Samp rate (Sps): 100k
Ch0: Center Freq (Hz): 2.42G
Ch0: AGC: Disabled
Ch0: Gain Value: 20
Ch0: Gain Type: Absolute (dB)
Ch0: Antenna: J1
Ch0: Bandwidth (Hz): 54M

Complex To Float

DC Blocker
Length: 32
Long Form: True

Multiply

Moving Average
Length: 50k
Scale: 20u
Max Iter: 500
Length of Vectors: 1

Log10
k: 0

File Sink
File: ...oject1/sdr1test2.log
Unbuffered: On
Append file: Overwrite

Figure 3.5 GNU Radio Flowgraph to find RSS of Incoming Signals

These scripts were used on the host laptop in the configuration shown in Figure 3.2. This connects the software to the localization hardware, completing the localization system.

3.2.2. Jamming

The software has various roles to fulfill in the jamming section of this project, these tasks are listed in the 3.1.1. microcontroller section of this paper. First switching the antenna between receiving and transmitting in order to both read the signal to the drone then properly jam it. The readings should then be processed into a digital format to work with. The data is then cleaned of any noise to isolate the important communication signal. Finally a pseudo-random signal, the barrage jamming noise, is created to be broadcasted and jam the communication to the drone.

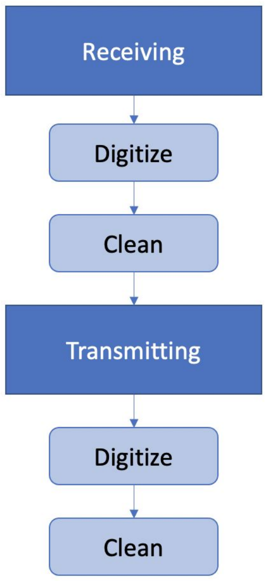


Figure 3.6 Software Flow Diagram

The microcontroller will begin on the receiving setting in order to first find the drone signals to jam. Then the data will be cleaned in order to filter out the noise and only record what is important signal data. This can be done by testing and finding boundaries at which actual drone signals work at and removing signals that are not within those bounds. The transmission will then create a jamming signal that is random and at the same frequency to disrupt the signal to the drone. The microcontroller will use barrage jamming in order to render either the 2.4 GHz communications useless. Finally this signal is broadcasted continuously sending the data back into the RF circuits for amplification.

The programming of the Arduino was done in MATLAB with the compatibility with Arduino and Simulink. This decision was made because there is a library in MATLAB, the Phased Array System Toolbox, which allows for barrage jamming noise to be calculated in one line of code. There will first be a switch case with the condition of finding a proper signal to change between receiving and transmitting functions. Once a signal is verified, the case switches to the transmission, where the barrage jamming signal will be created using the previous equation and then is transmitted back through the RF circuitry.

4. Results and Discussion

4.1. Localization Experiment

To test the localization portion of the system, 4 SDRs were placed in a 10 by 10 meter square at varying elevations. In order to test signals originating from outside the 10 by 10 bounds, the total testing area was set at 14 by 14 meters. Figure 4.1 below shows the configuration of the test set up from a top down view.

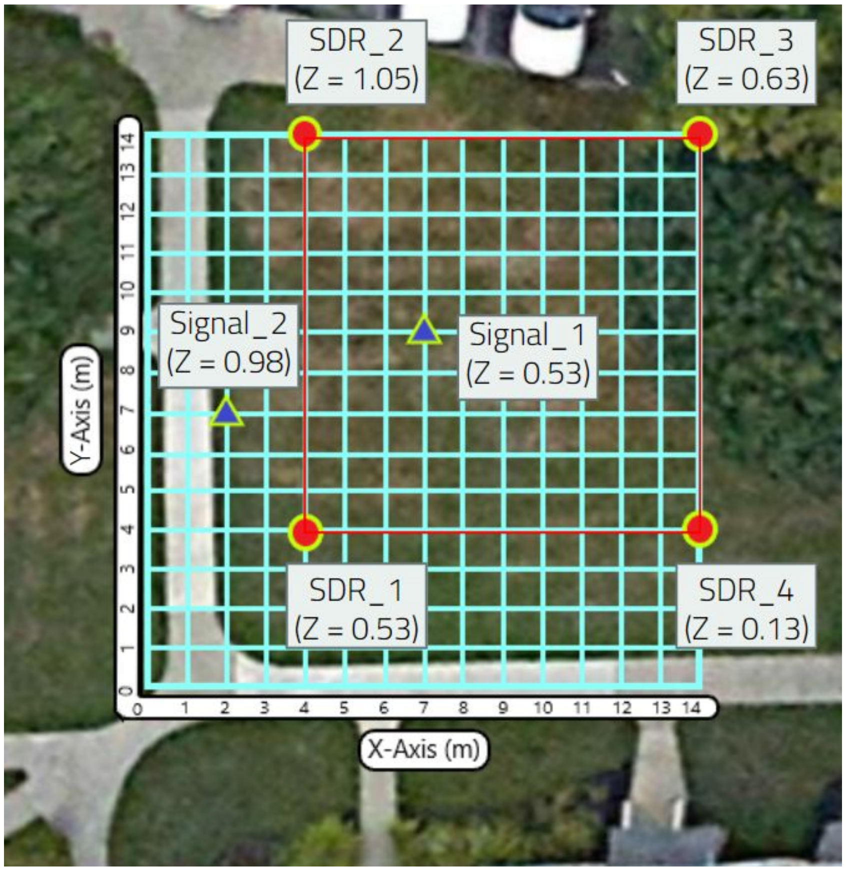


Figure 4.1 Testing Configuration for Localization, 4 SDRs, 2 Signal Positions

After setting up the testing configuration, the first step was to measure P_0, the signal strength of the controller 1 meter away individually for each SDR using the GNU Radio generated script from section 3.2.1. This step is used to get a reference strength to compare the RSS values to when the localization test begins. Once each SDR's P_0 was recorded, the controller was placed inside the 10 by 10 bounds. With the controller at this position, labeled Signal_1 in Figure 4.1, The GNU Radio script was run for approximately 5 seconds. Once over, the data for each SDR was averaged and used in the RLS algorithm. The controller was then moved to its position outside the bounds, labeled Signal_2 in Figure 4.1. The script was run again for approximately 5 seconds and the data was averaged and used in the RLS algorithm.

4.1.1. Mathematical Modelling

The first step in finding the location of the controller is determining the distance between the access points and the controller. This was done with the Equation 2.2 version of the path loss equation. This distance was calculated quickly with a Python script as well as the actual distance between the object and the controller using trigonometry. The following calculated distances and errors are demonstrated in the tables below.

Controller (3,5,0.53)	Calculated Distance (m)	Error (m)
SDR 1 (0, 0, 0.53)	5.86	-0.03
SDR 2 (0, 10, 1.05)	6.84	-0.99
SDR 3 (10, 10, 0.63)	7.65	0.95
SDR 4 (10,0, 0.13)	9.11	-0.49

Table 4.1 Controller at Position (3, 5, 0.53) Distance from SDRs

Controller (-2,3,0.98)	Calculated Distance (m)	Error (m)
SDR 1 (0, 0, 0.53)	3.81	-0.17
SDR 2 (0, 10, 1.05)	6.45	0.83
SDR 3 (10, 10, 0.63)	13.53	0.37
SDR 4 (10,0, 0.13)	12.19	0.2

Table 4.2 Controller at Position(-2,3,0.98) Distance from SDRs

Next using these calculated distances they are plugged into a Matlab script that calculates the position using the RLS algorithm. These results are visually shown below in the following figures. The controller at position (3, 5, 0.53) was calculated to be at (3.57, 5.20, 2.44). The controller at position (-2, 3, 0.98) was calculated to be at (-1.85,3.41,1.52).

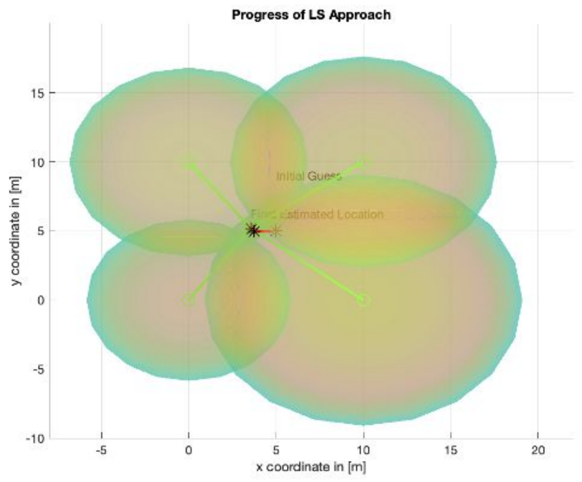


Figure 4.2 Controller at Position (3, 5, 0.53) RLS Algorithm

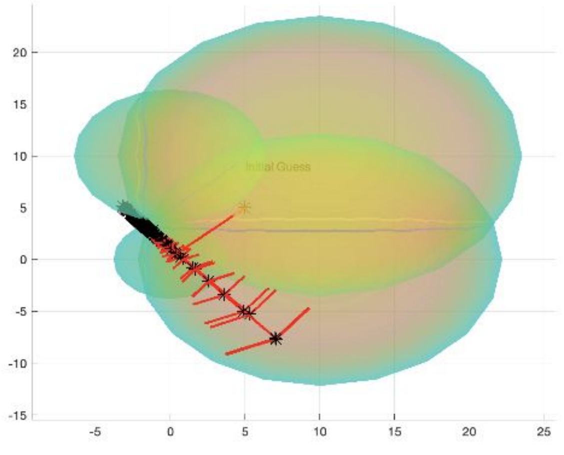
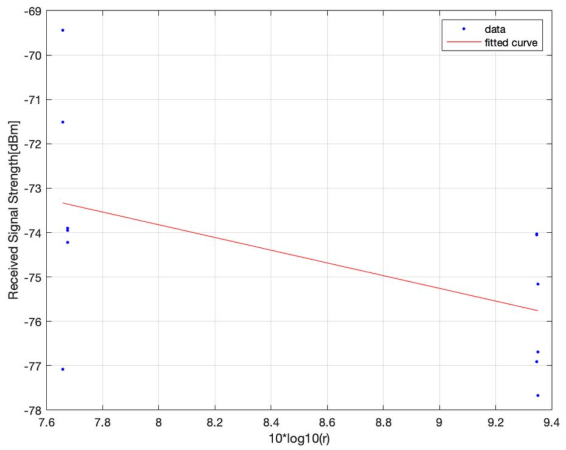


Figure 4.3 Controller at Position(-2,3,0.98) RLS Algorithm

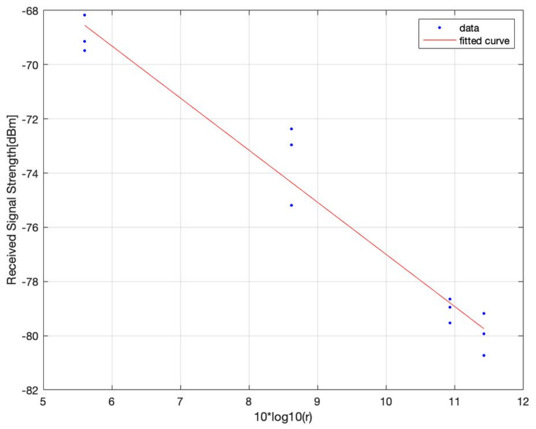
4.1.2. Error Estimation

Another MATLAB script was able to take the calculated distances from the path loss model and the RSS values and graph them giving a line of best fit with the values of P_0, α, and X(σ). These values are then used in the CRLB to calculate how much error this precise system will produce.



```
The Estimated Path Loss Model is:
Pr=-62.3642-14.3249*log(r)
Mean value of shadow fading is:
  -37.5362
Standard Deviation of shadow fading is:
    2.1047
```

Figure 4.4 Controller at Position (3, 5, 0.53) Path Loss Model



```
The Estimated Path Loss Model is:
Pr=-57.7782-19.2179*log(r)
Mean value of shadow fading is:
  -37.0628
Standard Deviation of shadow fading is:
    0.9637
```

Figure 4.5 Controller at Position(-2,3,0.98) Path Loss Model

Two top-down contour maps of the CRLB data were simulated of this configuration. Figure 4.6 below is the contour map at the height of the signal inside the bounds using its σ and α values. At the exact position of the signal, the standard deviation of location error is 1.705 meters.

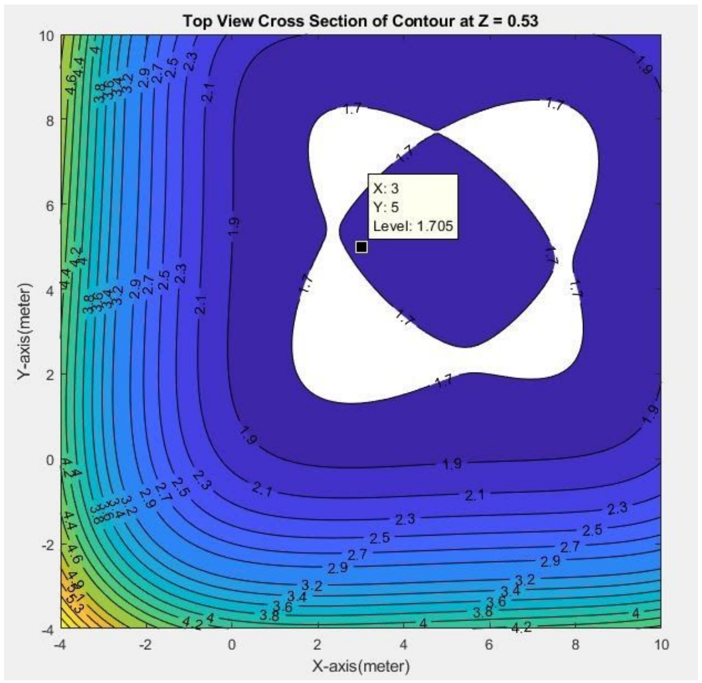


Figure 4.6 Top-Down Contour Map Using σ and α Values of the Signal Inside the Bounds

Figure 4.7 below is the contour map at the height of the signal outside the bounds at its elevation. This map was generated with the σ and α values of its respective signal. At the signal's exact position, the standard deviation of the error is 2.13 meters. Overall, both maps show that the estimated error of our system is low, especially within the 10 by 10 meter square of the SDRs. This means that our system seems to be accurate at locating the position of the drone.

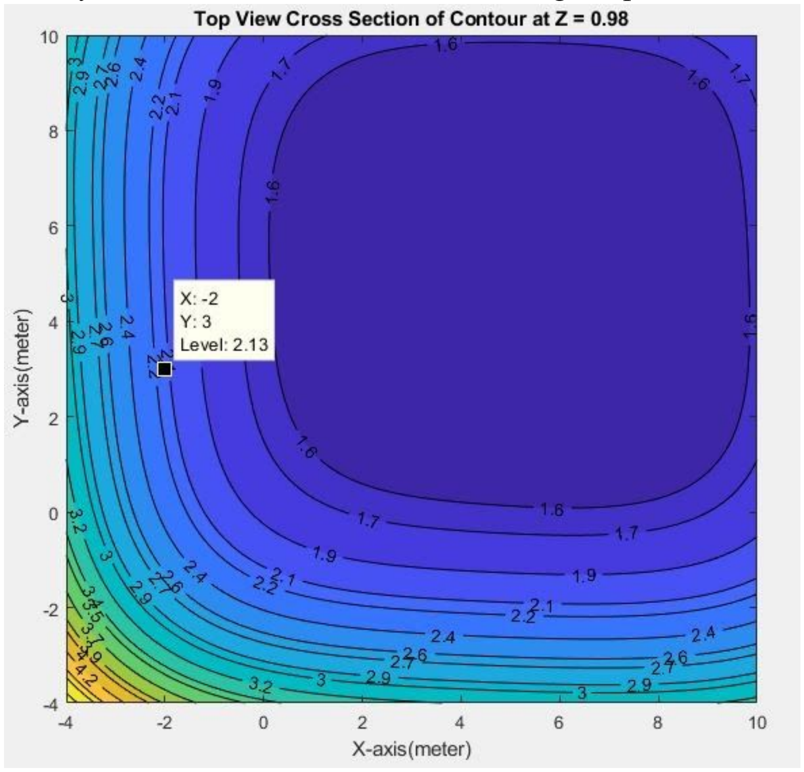


Figure 4.7 Top-Down Contour Map Using σ and α Values of the Signal Outside the Bounds

4.2. Jamming Analysis

4.2.1. Final Assembly Design

After ordering all components and determining which inputs and outputs each component requires for the jamming assembly to operate properly, a final assembly design needs to be created as an outline for how the entire assembly needs to be controlled through wiring. This design was created and modified in Visio, and is shown in Figure 4.8 below.

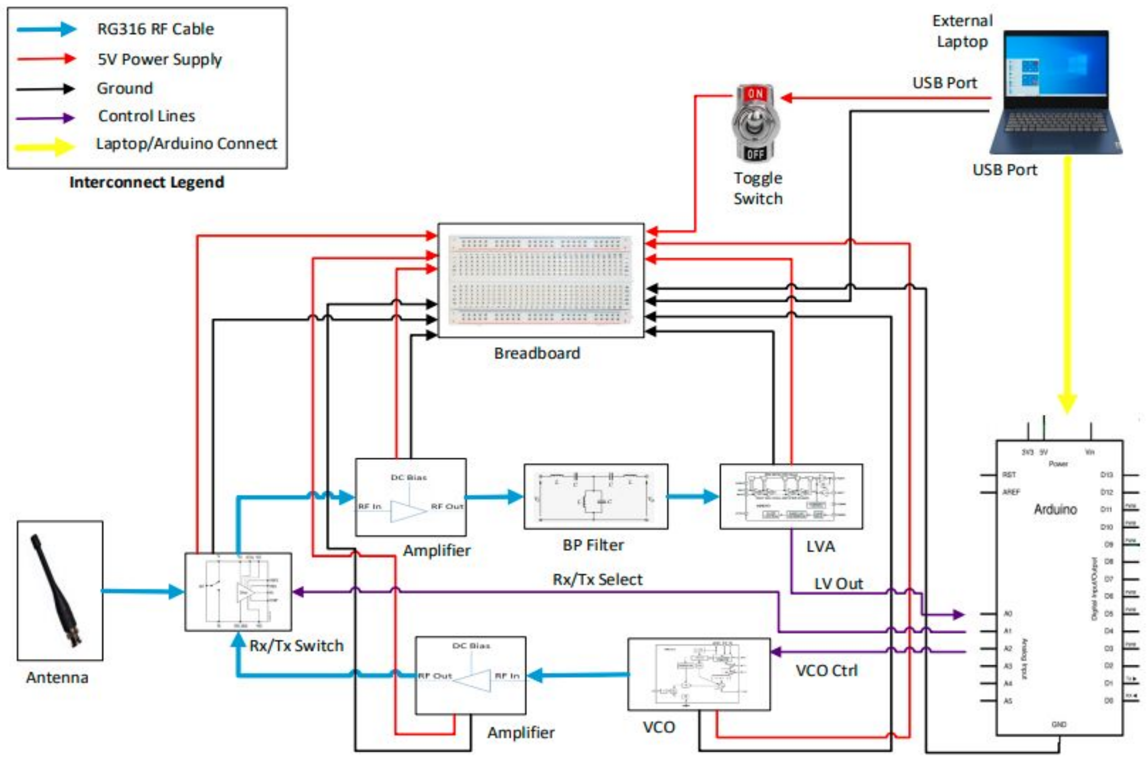


Figure 4.8 Final Jammer Assembly Design

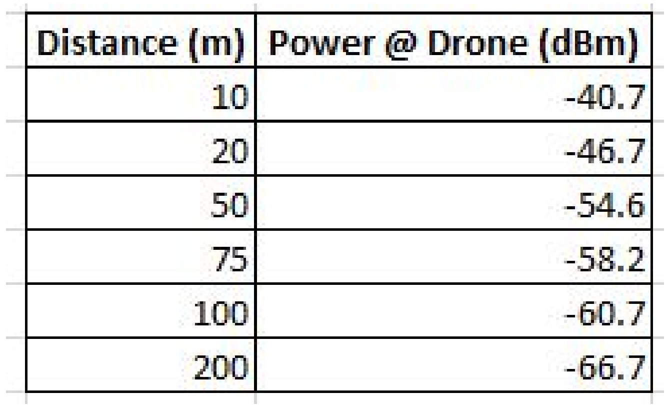
All connections for the jammer assembly are accomplished through RG316 RF cable, a 5V power supply from the laptop, a common ground, control lines to and from the Arduino, and a laptop/Arduino connect (USB cord). As highlighted in section 3.1.2, all components are powered by a single external laptop. This saves money and wasted power, as we do not need to order rechargeable or disposable batteries to power the RF components. Another important aspect of this jammer design is the switch, which is used as both a power saving and safety aspect. When the laptop is plugged in, the switch in the off mode will not allow any power to be drawn from the laptop. Not only does this limit power usage to when the user intends to use the assembly, but it also allows the user to plug in the laptop in any environment without potentially unintentionally jamming signals.

4.2.2. Power Considerations

The power level of a signal determines which signal the receiving party (in our case, a drone) will be able to process. In theory, any jamming to signal (J/S) ratio greater than 0 dB will jam the drone. However, a J/S ratio of 6dB or greater will jam the drone with extremely high probability, and this is the ratio our jammer aims to meet. The power that our test drone, the Tinyhawk II, uses for communication was measured in a lab environment to be 19.4 dBm. The output power of the jammer assembly was determined from the component specifications in Table 3.2 to be 27.1 dBm. Tables 4.3 and 4.4 below show the controller power at the drone over distance and the jammer power at the drone over distance, respectively. The free space path loss (FSPL) equation used in determining the received power at specific distances is:

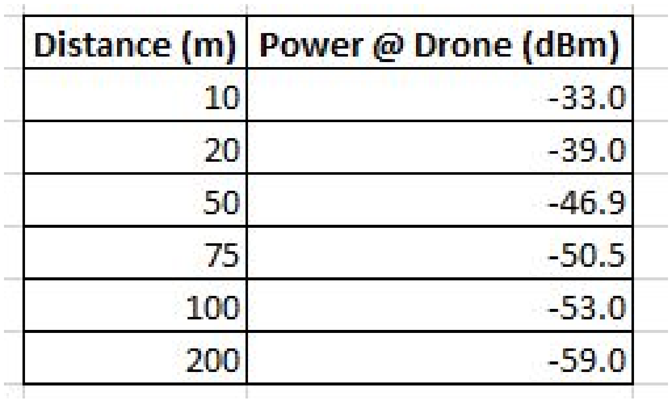
$$FSPL\ (dB)\ =\ 20log(d) + 20log(f) + 20log(4pi/c) \tag{4.1}$$[7]

Where d is the distance in meters between the transmitted signal and where is is received, f is the frequency of the transmitted signal in Hertz (2,400,000,000 Hz), and c is the speed of light (3*10^8 m/s).



Distance (m)	Power @ Drone (dBm)
10	-40.7
20	-46.7
50	-54.6
75	-58.2
100	-60.7
200	-66.7

Table 4.3: Controller Power at Drone vs Distance



Distance (m)	Power @ Drone (dBm)
10	-33.0
20	-39.0
50	-46.9
75	-50.5
100	-53.0
200	-59.0

Table 4.4: Jammer Power at Drone vs Distance

The major conclusion drawn from Tables 4.3 and 4.4 is that when the distances between the drone and controller and the drone and jammer are the same, the power of the jamming signal at the drone is 7.7 dB higher than the controller signal power. This is greater than 6 dB, which is the aim of our jamming assembly. Therefore, the drone jamming assembly is theoretically effective with high probability when the jammer and controller are equidistant from the drone.

5. Conclusions and Future Recommendations

In this section, the conclusions drawn from the results of testing the system and analyzing the data are discussed. The three sections talk about the localization part of the assembly, the jamming part of the assembly, and the assembly as a whole. Also, future recommendations for building upon this project are given.

5.1. Positioning Portion of Assembly

In the positioning portion of this project, the most important thing is to make sure the final location estimated is as close to actual as possible. The RLS algorithm calculated the controller at position (3, 5, 0.53) to be at (3.57, 5.20, 2.44). The controller at position (-2, 3, 0.98) was calculated by RLS to be at (-1.85,3.41,1.52). Both of these calculations were within a meter of error on the xy plane, however, on the z axis the error was greater. This is due to the lack of variance of the heights of the SDRs. On the xy plane the SDRs were all 10 meters apart, however, on the z axis there was only a variance of one meter. This led to the ability to determine the z position as far worse than the x or y position. This could be fixed in future applications by increasing the difference in heights of the SDRs. Future projects could also find a way to determine if the signal is coming from either the controller or the drone itself to know which is being located.

5.2. Jamming Portion of Assembly

In the jamming portion of this project, the most important thing is to make sure the drone can be disabled and jammed. Although the power considerations, discussed in Section 4.2.2 indicates that the jammer will disable the drone when the distance between the drone and jammer is equidistant to the drone and the controller distance. However, due to WPI team project COVID restrictions, we have not been able to test this theory with the designed jammer assembly. Future projects could design an automated antenna system that focuses the jamming signal in the direction of the drone. This better ties the localization section of this project to the jamming section, while also ensuring that the jamming signal is focused in the desired direction. Another consideration for future projects could be to find ways to increase the power level of the output jamming signal. This would increase the overall jamming range of the assembly, which would be beneficial in an application within a larger area of interest.

5.3. Complete Assembly

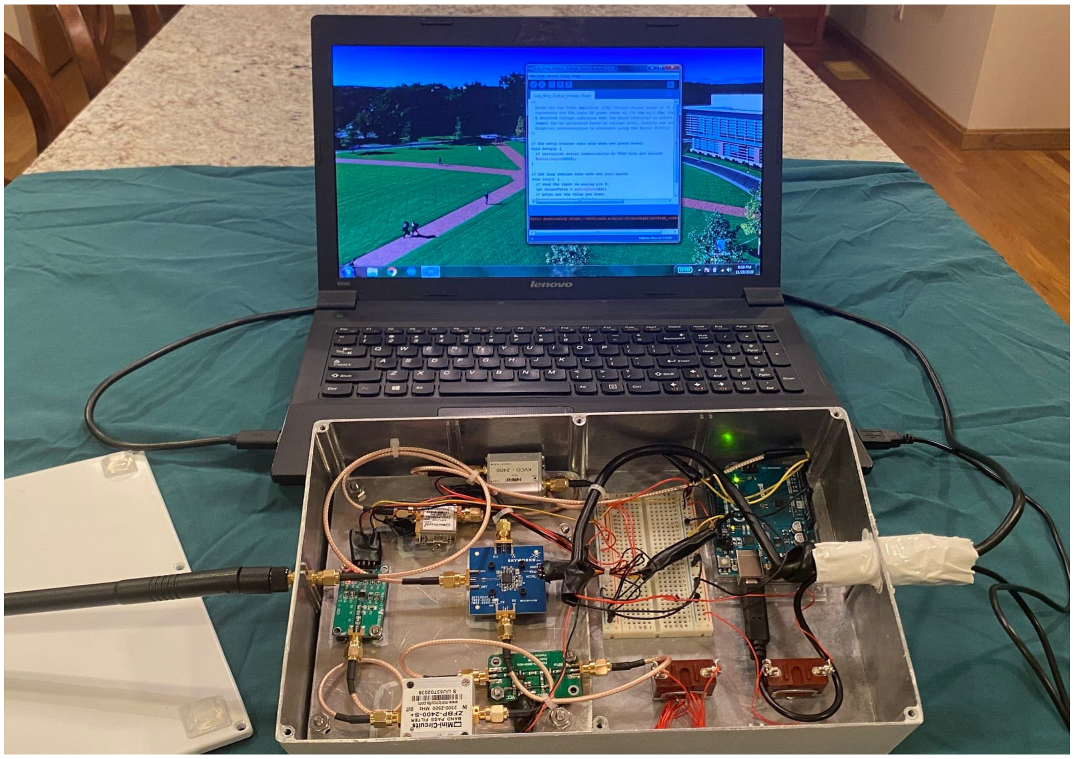


Figure 5.1 Final Jammer Assembly Opened

The Drone Jammer functionality as shown above in Figure 5.1 is composed of two primary functions, receive and jam. In theory, the controller frequency would be detected by the Drone Jammer, and if desired, the Drone Jammer would transmit a duplicated frequency in an effort to jam the drone. The transmission would need to be of higher power as seen by the drone, than the controller transmission. The idea behind this jamming technique is to provide a higher power level (greater than 6 dB) than the controller to "overpower" the drone and further confuse the drone by introducing Gaussian white noise.

The receive function is used to detect that a drone and controller is in the local area. The omni-directional antenna (B01M3NVW22) is used to pick the controller radio signals out of the air and to translate it to a Voltage for detection. This omni-directional antenna has 9dBi of gain. The received signal is steered to the detection path via the Rx/Tx switch. The Rx/Tx switch (SKY12242_492LF) is specified to have 0.7 dB of loss. The RF signal then gets amplified by a low noise amplifier (B07B6P2B7H). This amplifier has a low noise figure of 2.1 dB and a high gain of 35dB. The signal is then passed through a RF bandpass filter (ZFBP-2400-S+) that passes signals from 2300 MHz to 2500 MHz and rejects RF signals that are out of this range. The controller itself has a specified frequency range of 2400 MHz to 2483.5 MHz. This device filters out non-desirable signals in the environment. The signal is then directed to the log video amplifier (AD8313) which translates RF input signals from -68.5 dBm to -0.5 dBm to an output Voltage of 0V to 5V that is linearly proportional to the RF input power range. The LVA output Voltage is fed to the Arduino that detects that a Voltage is present and in summary a drone controller is detected in the environment.

If desired, the jam function could then be deployed. The Arduino would output a fixed Voltage to tune the Voltage controlled oscillator (HE6-YY2-HK935) to the specific controller frequency (would have to be predetermined by measurement in the lab). The VCO outputs a Voltage range of 2300 MHz to 2650 MHz based on an input tune Voltage of 0V to 4.5V. The VCO output power is 6.0 dBm. The RF signal would then pass through a RF amplifier (139-ZX60-P105LN) with 20 dB of gain and 20.0 dBm of output power. The Rx/Tx switch would then steer the RF jam signal to the same antenna (bidirectional and now used for jamming functionality). The signal would then be sent back into the environment via the antenna and in theory would be received by the drone and would disable its functionality.

The RF component building blocks were procured from various sources with performance, size, low 5V DC current, and standard SMA connectors being key decision factors. An Arduino Uno was selected as the controller for the Drone Jammer. A simple breadboard was used for component control and DC power supply interconnects. Two manual switches were incorporated, one to power on/off the box and the other as a spare. The RF component interconnects were made with short RG316 low loss RF cables with standard SMA RF connectors. The DC control and power supply lines were implemented with standard 22 gauge wire. Various wire colors were used to minimize errors; red for 5V, black for grounds, and yellow for control/data lines. The Drone Jammer has two connections to a laptop. One connection is to a USB port that provides 5V up to 500 mA to power the box (all components combined less than 350 mA) and the other connection is a separate USB port that communicates and controls the Arduino. A standard bud box was used to house the RF Jammer.

Figure 5.2 Final Jammer Assembly Closed

Shown above in Figure 5.2 is the appearance of the Jammer Assembly when it is closed. The bud box was painted white and stickers were added for aesthetic. The top switch controls whether or not the power from the laptop is powering all of the active components. The bottom switch was added for potential future functionality if more features were to be added in the future, but for now is not connected to any other components.

The COVID environment greatly limited the ability to debug and test the Drone Jammer. The WPI Engineering Laboratories and associated test equipment were not available for a majority of the term. The test equipment was limited to a simple multimeter in the home environment. The multimeter was initially used to "ohm out" the voltages, grounds, and control lines for potential shorts. There were no shorts or miss wires found in the fully assembled box, shown in Figure 5.1. Ideally, an RF generator source set at approximately 2400 MHz to 2500 MHz with varying power levels would be used to test the detection function. The log video amplifier output would be monitored and plotted by the Arduino as a function of frequency and RF power level. Without test equipment, the detection testing was limited to turning on the drone controller and confirming that the log video out voltage as monitored and displayed by the Arduino increased and dropped as the drone controller was turned on and off. The actual modulation of the controller output signal and its exact frequency could have been measured in the lab with a spectrum analyzer but with no test equipment the detection testing was again limited to measuring an increase in input power level as detected by the log video amplifier.

Jamming testing was limited to stepping the voltage controlled oscillator across its frequency range using a tunable voltage out of the Arduino and transmitting the signal out of the Drone Jammer box. An active drone was flown near the box to see if it negatively impacted the drone's flight. Output frequencies were varied and the drone distance to the jammer was varied, but there were no negative impacts to the drone. It was noted that the home Wi-Fi was consistently disabled during the testing, but again the drone flight was not impacted. A spectrum analyzer measuring the output frequency and amplitude would have aided the testing and debugging of the box. An oscilloscope would have also aided in the debugging of the Drone Jammer, as we could have analyzed the Voltage levels at each step of the assembly over a time domain to discern any undesired responses. In conclusion, the assembly of the Drone Jammer was completed as designed, but due to COVID measures taken by WPI, it became increasingly more difficult to debug any issues within the complex system without adequate testing equipment available.

BIBLIOGRAPHY

1. FAA, “FAA AEROSPACE FORECAST,” FAA, 2019. [Online]. Available: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2019-39_FAA_Aerospace_Forecast.pdf.

2. M. Petrova, “Watch what happens when a small drone hits a plane at 238 miles per hour,” *CNBC*, Oct. 17, 2018. https://www.cnbc.com/2018/10/17/drone-hits-plane-wing-shears-off-video.html.

3. “Drone that struck plane near Quebec City airport was breaking the rules | CBC News,” *CBCnews*, 15-Oct-2017. [Online]. Available: https://www.cbc.ca/news/canada/montreal/garneau-airport-drone-quebec-1.4355792.

4. S. Shackle, “The mystery of the Gatwick drone,” 01-Dec-2020. [Online]. Available: https://www.theguardian.com/uk-news/2020/dec/01/the-mystery-of-the-gatwick-drone.

5. A. Hassan, I. Gelman, J. Loftus, and B. Mahan, “Adversary UAV Localization With Software Defined Radio,” *Digital WPI*, Apr-2019. [Online]. Available: https://digitalcommons.wpi.edu/mqp-all/7115/.]

6. K. Pahlavan, “Chapter 3: RSS Positioning Systems,” in *Indoor geolocation science and technology: at the emergence of smart world and IoT*, River Publishers, 2019, pp. 53–62.

7. Phuc Nguyen, Hoang Truong, Mahesh Ravindranathan, Anh Nguyen, Richard Han and Tam Vu, "Drone Presence Detection by Identifying Physical Signatures in the Drone's RF Communication", *MobiSys '17 Proceedings of the 15th Annual International Conference on Mobile Systems Applications and Services*, pp. 211-224, 2017.

8. “Barrage Jammer - MATLAB & Simulink,” *www.mathworks.com*. https://www.mathworks.com/help/phased/ug/barrage-jammer.html (accessed Dec. 10, 2020).

9. “Arduino MKR WiFi 1010,” *Arduino.cc*, 2019. https://store.arduino.cc/usa/mkr-wifi-1010.

APPENDIX

Contributions by Sections

Nicholas Cauley	Ryan Crowley	Mara Pranter
2. Theoretical Background	1. Introduction	2.1 Localization
2.2.1 Frequency Considerations	1.1 Project Description	2.1.3 Path Loss Model
2.2.2 Jamming Scenario	1.2 Report Outline	2.1.4 RLS Algorithm
3.1.2 RF Components	2. Theoretical Background	2.2.3 Software Considerations
4.2.1 Final Assembly Design	2.1.1 Signal Processing	3.1.1 Microcontroller
4.2.2 Power Considerations	2.1.2 Localization Setup	3.3 Software Modules
5. Conclusions and Future Recommendations	2.1.5 CRLB Error Analysis in 2D	4.1.1 Mathematical Modelling
5.2 Jamming Portion of Assembly	2.1.6 CRLB Error Analysis in 3D	4.1.2 Error Estimation
5.3 Complete Assembly	3.2.1 Receiving and Processing	5.1 Positioning Portion of Assembly
	4.1 Localization Experiment	
	4.1.2 Error Estimation	
	5. Conclusions and Future Recommendations	

3D RLS MATLAB Script

```
clc;clear;close all;

% This Matlab code solve Example 7.2 in textbook
known_references = [0, 0, 0.53;0, 10, 1.05;10, 10, 0.63;10, 0, 0.13];
initial_guess = [5, 5, 1];
distances =
[3.8062735760120607,6.446638734864327,13.531107155919049,12.194574932461649];

figure(1);
hold on
grid on

% Draw Spheres
[X, Y, Z] = sphere;

sphere1 = surf(X * distances(1) + known_references(1,1), Y * distances(1) +
known_references(1,2), Z * distances(1) + known_references(1,3));
sphere2 = surf(X * distances(2) + known_references(2,1), Y * distances(2) +
known_references(2,2), Z * distances(2) + known_references(2,3));
sphere3 = surf(X * distances(3) + known_references(3,1), Y * distances(3) +
known_references(3,2), Z * distances(3) + known_references(3,3));
sphere4 = surf(X * distances(4) + known_references(4,1), Y * distances(4) +
known_references(4,2), Z * distances(4) + known_references(4,3));

set(sphere1, 'FaceAlpha', 0.5);
set(sphere2, 'FaceAlpha', 0.5);
set(sphere3, 'FaceAlpha', 0.5);
set(sphere4, 'FaceAlpha', 0.5);

shading interp;

i=1;
temp_location(i,:) = initial_guess;
temp_error = 0;

for j = 1 : size(known_references,1)
```

```
   temp_error = temp_error + abs((known_references(j,1) - temp_location(i,1))^2 +
(known_references(j,2) - temp_location(i,2))^2 + (known_references(j,3) - temp_location(i,3))^2
- distances(j)^2);
end

estimated_error = temp_error;

plot3(temp_location(i,1),temp_location(i,2),temp_location(i,3),'k*','MarkerSize', 10); %plot
text(temp_location(i,1), temp_location(i,2)*(1 + 0.8), 'Initial Guess');
disp(['The initial location estimation is:',
num2str([temp_location(i,1),temp_location(i,2),temp_location(i,3)])]);

% new matrix = [ ];
while norm(estimated_error) > 1e-2 %iterative process for LS algorithm
   for j=1 : size(known_references,1)  %Jacobian has been calculated in advance
       jacobian_matrix(j,:) = -2*(known_references(j,:) - temp_location(i,:)); %partial derivative is
i.e. -2(x 1-x)
       f(j) = (known_references(j,1) - temp_location(i,1))^2 + (known_references(j,2) -
temp_location(i,2))^2 + (known_references(j,3) - temp_location(i,3))^2 - distances(j)^2;
   end

   estimated_error = -inv(jacobian_matrix' * jacobian_matrix) * (jacobian_matrix') * f'; %update
the U and E
   temp_location(i+1,:) = temp_location(i,:) + estimated_error';

   plot3(temp_location(i+1,1),temp_location(i+1,2),temp_location(i+1,3),'k*','MarkerSize',10);
%plot
   dp = temp_location(i+1,:)-temp_location(i,:);

quiver3(temp_location(i,1),temp_location(i,2),temp_location(i,3),dp(1),dp(2),dp(3),0,'Color','r','
LineWidth',2);
   %text(temp location(i+1,1), temp location(i+1,2)*(1 + 0.005), num2str(i));

   i = i + 1;
   lx=num2str(temp_location(i,1));
   ly=num2str(temp_location(i,2));
   lz=num2str(temp_location(i,3));
   err=sqrt(estimated_error(1)^2+estimated_error(2)^2+estimated_error(3)^2);
```

```
  disp(['The ',num2str(i-1), 'th estimated location is:','[',lx,',',ly,',',lz,']',' with an error of ',
num2str(err)]);
end

plot3(known_references(:,1),known_references(:,2),known_references(:,3),'go','MarkerSize',10);

for i=1:length(known_references)
  dp = temp_location(end,:) - known_references(i,:);

quiver3(known_references(i,1),known_references(i,2),known_references(i,3),dp(1),dp(2),dp(3),0
,'Color','g','LineWidth',2);
end

text(temp_location(end,1), temp_location(end,2)*(1 + 0.2),'Final Estimated Location');
%axis([1.1*min(temp location(:,1)),1.1*max(temp location(:,1)), 0.9*min(temp location(:,2)),
1.1*max(temp location(:,2))]);
title('Progress of LS Approach')
xlabel('x coordinate in [m]');
ylabel('y coordinate in [m]');
zlabel('z coordinate in [m]');
```

Path Loss Model MATLAB Script

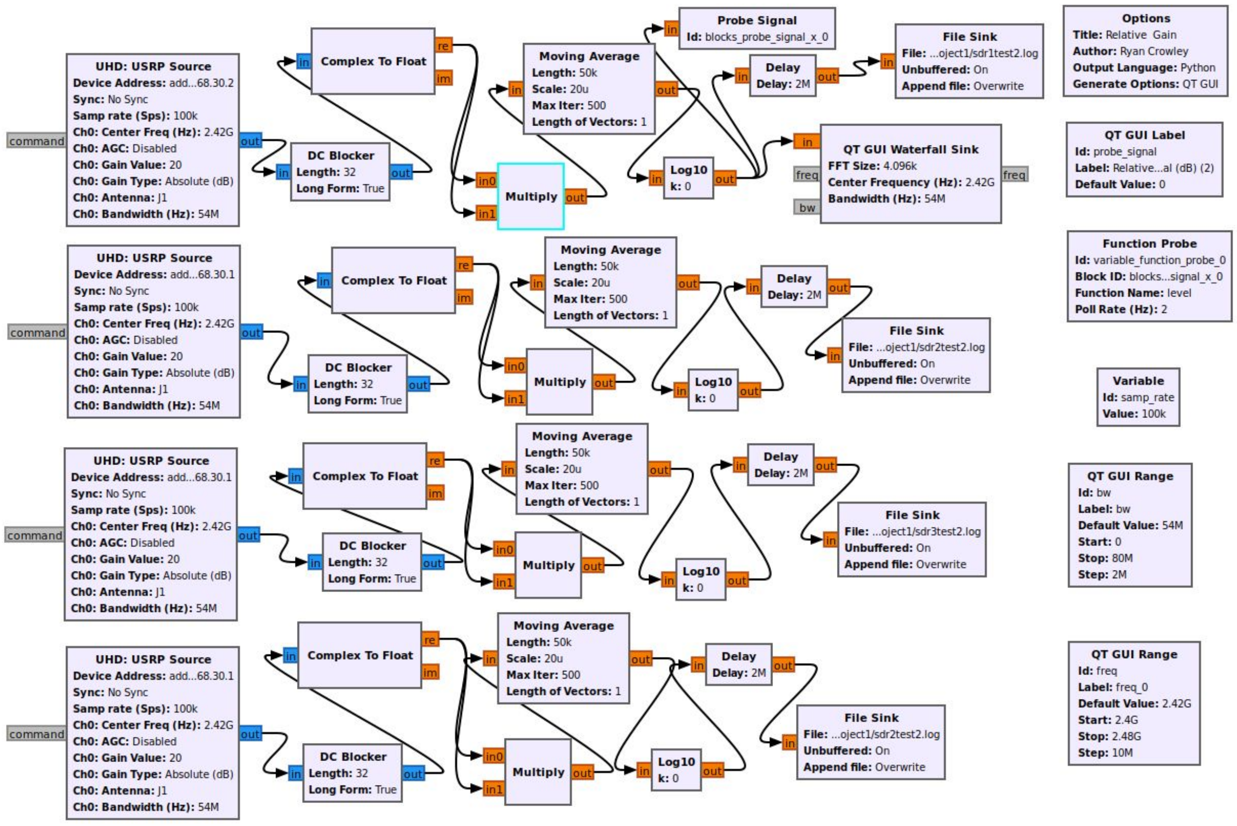
```
clc;close all;
% Enter your distance in meter here and rss
r=[5.0;6.5;8.0;9.5;11.0;12.5;14.0;15.5;17.0;20.0];
Pr=[-38.9459;-38.8443;-36.2157;-39.1163;-36.3094;-
44.6655;-40.8041;-46.7637;-42.3983;-50.2684];
% Plot your Lp vs. Distance in dB
r_dB=10*log10(r); % Careful! Change your distance in 10log(r) when plotting
% Linear Fitting
F1=fit(r_dB,Pr,'poly1');
%Plot your data with fitting
plot(F1,r_dB,Pr);
grid on
xlabel('10*log10(r)');
ylabel('Received Signal Strength[dBm]');
val=coeffvalues(F1);
disp('The Estimated Path Loss Model is:');
model1=['Pr=',num2str(val(2)),num2str(val(1)*10),'*log(r)'];
disp(model1);
disp('Mean value of shadow fading is:');
disp(mean(Pr+20+2*r_dB));
disp('Standard Deviation of shadow fading is:');
disp(std(Pr+40+2*r_dB));
```

3D CRLB MATLAB Script

```
%%
% The simulation is based on Kobayashi's paper 'Signal Strength Based Indoor Geolocation'
% The result can be validated by Fig. 3 in the paper
% Calculation of CRLB is based on Equation (3.3d),(3.4e) and (3.4d)
%%
close all;clear all;clc;warning off;
%% Initialization
% Locations of Access Points
% results
APx(1)=0;APy(1)=0;APz(1)=0.53;
APx(2)=10;APy(2)=0;APz(2)=1.05;
APx(3)=0;APy(3)=10;APz(3)=0.63;
APx(4)=10;APy(4)=10;APz(4)=0.13;
%APx(5)=0;APy(5)=0;APz(5)=-7;
%APx(6)=-5;APy(6)=5;APz(6)=5;
SD=.98; % Standard Deviation of Shadow Fading
NUM=4; % Number of Access Points
% Locations of Receivers
pace=.1;
mx=-4:pace:10;
my=-4:pace:10;
mz=-4:pace:10;
%nxyz=length(mx);
for zi=1:length(mx)
   for yi=1:length(mx)
      for xi=1:length(mx)
         for i1=1:NUM
            alpha=2;
            r(i1,xi,yi,zi)=sqrt((mx(xi)-APx(i1))^2+(my(yi)-APy(i1))^2+(mz(zi)-APz(i1))^2); %
Distance Between Transmitter and Receiver
            H1(i1,xi,yi,zi)=-10*alpha/log(10)*(mx(xi)-APx(i1))/(r(i1,xi,yi,zi))^2; % First Column
of H Matrix
            H2(i1,xi,yi,zi)=-10*alpha/log(10)*(mx(yi)-APy(i1))/(r(i1,xi,yi,zi))^2; % Second
Column of H Matrix
            H3(i1,xi,yi,zi)=-10*alpha/log(10)*(mx(zi)-APz(i1))/(r(i1,xi,yi,zi))^2; % Third Column
of H Matrix
         end
         H(:,:,xi,yi,zi)=[H1(:,xi,yi,zi),H2(:,xi,yi,zi),H3(:,xi,yi,zi)];
         dddd = (H(:,:,xi,yi,zi))'*H(:,:,xi,yi,zi);
         Covv(:,:,xi,yi,zi)=SD^2*dddd^(-1); % Covariance Matrix of Error Estimate
         SDr(xi,yi,zi)=sqrt(Covv(1,1,xi,yi,zi)+Covv(2,2,xi,yi,zi)+Covv(3,3,xi,yi,zi)); % Standard
Deviation of Location Error
      end
   end
end
```

```
figure(1)
[C h]=contourf(mx,my,SDr(:,:,1),20);
h.LevelList=round(h.LevelList,1);  %rounds levels to first decimal place
clabel(C,h);
xlabel('X-axis(meter)');
ylabel('Y-axis(meter)');
title('Top View Cross Section of Contour at Z = 0')
```

GNU Radio Flowgraph



UHD: USRP Source
Device Address: add...68.30.2
Sync: No Sync
Samp rate (Sps): 100k
Ch0: Center Freq (Hz): 2.42G
Ch0: AGC: Disabled
Ch0: Gain Value: 20
Ch0: Gain Type: Absolute (dB)
Ch0: Antenna: J1
Ch0: Bandwidth (Hz): 54M

Complex To Float

DC Blocker
Length: 32
Long Form: True

Multiply

Moving Average
Length: 50k
Scale: 20u
Max Iter: 500
Length of Vectors: 1

Probe Signal
Id: blocks_probe_signal_x_0

Log10
k: 0

Delay
Delay: 2M

File Sink
File: ...oject1/sdr1test2.log
Unbuffered: On
Append file: Overwrite

QT GUI Waterfall Sink
FFT Size: 4.096k
Center Frequency (Hz): 2.42G
Bandwidth (Hz): 54M

UHD: USRP Source
Device Address: add...68.30.1
Sync: No Sync
Samp rate (Sps): 100k
Ch0: Center Freq (Hz): 2.42G
Ch0: AGC: Disabled
Ch0: Gain Value: 20
Ch0: Gain Type: Absolute (dB)
Ch0: Antenna: J1
Ch0: Bandwidth (Hz): 54M

Complex To Float

DC Blocker
Length: 32
Long Form: True

Multiply

Moving Average
Length: 50k
Scale: 20u
Max Iter: 500
Length of Vectors: 1

Log10
k: 0

Delay
Delay: 2M

File Sink
File: ...oject1/sdr2test2.log
Unbuffered: On
Append file: Overwrite

UHD: USRP Source
Device Address: add...68.30.1
Sync: No Sync
Samp rate (Sps): 100k
Ch0: Center Freq (Hz): 2.42G
Ch0: AGC: Disabled
Ch0: Gain Value: 20
Ch0: Gain Type: Absolute (dB)
Ch0: Antenna: J1
Ch0: Bandwidth (Hz): 54M

Complex To Float

DC Blocker
Length: 32
Long Form: True

Multiply

Moving Average
Length: 50k
Scale: 20u
Max Iter: 500
Length of Vectors: 1

Log10
k: 0

Delay
Delay: 2M

File Sink
File: ...oject1/sdr3test2.log
Unbuffered: On
Append file: Overwrite

UHD: USRP Source
Device Address: add...68.30.1
Sync: No Sync
Samp rate (Sps): 100k
Ch0: Center Freq (Hz): 2.42G
Ch0: AGC: Disabled
Ch0: Gain Value: 20
Ch0: Gain Type: Absolute (dB)
Ch0: Antenna: J1
Ch0: Bandwidth (Hz): 54M

Complex To Float

DC Blocker
Length: 32
Long Form: True

Multiply

Moving Average
Length: 50k
Scale: 20u
Max Iter: 500
Length of Vectors: 1

Log10
k: 0

Delay
Delay: 2M

File Sink
File: ...oject1/sdr2test2.log
Unbuffered: On
Append file: Overwrite

Options
Title: Relative Gain
Author: Ryan Crowley
Output Language: Python
Generate Options: QT GUI

QT GUI Label
Id: probe_signal
Label: Relative...al (dB) (2)
Default Value: 0

Function Probe
Id: variable_function_probe_0
Block ID: blocks...signal_x_0
Function Name: level
Poll Rate (Hz): 2

Variable
Id: samp_rate
Value: 100k

QT GUI Range
Id: bw
Label: bw
Default Value: 54M
Start: 0
Stop: 80M
Step: 2M

QT GUI Range
Id: freq
Label: freq_0
Default Value: 2.42G
Start: 2.4G
Stop: 2.48G
Step: 10M