Major Qualifying Projects (All Years)   Major Qualifying Projects

April 2019

Internet of Things Spectrum Monitoring and Localization

Katherine M. Smith
Worcester Polytechnic Institute

Lauren Rachel Getz
Worcester Polytechnic Institute

William Redmond Schwartz
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Internet of Things Spectrum Monitoring and Localization

A Major Qualifying Project Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the
Degree of Bachelor of Science
By:

_____

Lauren Getz

_____

William Schwartz

_____

Katherine Smith

Project Advisors:

_____

Professor Stephen J. Bitar, Electrical & Computer Engineering- Advisor-of-Record

_____

Professor Michael J. Ciaraldi, Computer Science -Co-Advisor

_____

Professor Alexander M. Wyglinski, Electrical & Computer Engineering- Co-Advisor

Abstract

A system which localizes indoor sub-1GHz transmission signals provides a low cost and effective alternative to traditional signal sensing and localization. The system uses a pre-existing WiFi network, PlutoSDRs, and a web server to visualize the estimated location of the transmitted signals. The system achieved 2.5-meter accuracy under ideal conditions satisfying the goal of 3-meter accuracy set forth by the team.

Acknowledgements

We would like to thank our advisors, Stephen Bitar, Michael Ciaraldi, and Alex Wyglinski for

their mentorship throughout the duration of this project.

Executive Summary

Implementing localization with SDR devices is more cost effective and more accurate than other spectrum sensing products currently on the market. With the development of Internet of Things (IoT), many smart devices are increasingly using wireless communication, making localization of these devices an important security measure. Using the PlutoSDR, signals were effectively localized within 3-meter accuracy on the 900MHz band.

Problem Statement

IoT devices can pose a threat to a secure network. Since there are a lot of known IoT vulnerabilities [1], many malicious actors will target them in order to gain access to an otherwise secure network. Due to this, it is important to know when and where IoT devices reside on a network. It is important to know when these devices are broadcasting information and where they are broadcasting this information from to determine if traffic being generated is malicious or benign.

Current State of the Art

Most competing solutions exist at 2.4 GHz and are used for navigation for devices to know their own position in situations where GPS does not work, such as big cities. Currently top companies in the technology market do not have consumer products available for indoor localization, although Apple and Google have bought into the industry and are expected to release applications in the near future [2].

There are very few devices which can both localize signals and sense the signals. Typically, devices operate as spectrum analyzers which can determine which band a signal is coming from.

One of these such devices is the 4.4Ghz Sound Hound device which retails for nearly $1000 and does not possess localization capabilities [3].

Proposed Approach

Create an easily deployable system to localize sub-1 GHz transmission. The goal for this project is to create a system that can localize a transmitter continuously broadcasting a single tone within 3 meters of accuracy in real-time.

The system is uses the PlutoSDR [4] which is an open source software defined radio created by Analog Devices. It was developed to serve as an active learning module for students. This platform was chosen because it is extremely capable for its price point. Similar functionality typically costs at least $300 and cannot easily be programmed to run as a standalone system [5].

Using a minimum of 3 PlutoSDR modules configured as receivers in different known locations to estimate the location of a single tone transmitter. To do this, the received signal strength (RSS) collected by each of the plutos. Then, use the onboard processing power of the PlutoSDRs to sample IQ values, compute a 2048-point FFT, and capture the bin containing the maximum power measurement in real time.

These power measurements are then be sent to a central server over WiFi. With the known location of the receivers, we can create a 90% confidence interval using Maximum Likelihood Estimation (MLE) of the distance between the transmitter and the receiver. By collecting these estimates from deployed PlutoSDRs we can use all of the 1-dimensional ranging estimates from the MLE 90% confidence intervals and find the overlap of the regions of confidence to reasonably estimate the location of the transmitter. These estimates are stored on the central server and displayed through a hosted web-app in real time. All the measurements and estimates are also stored to allow for further analysis.

Results

This project successfully implemented the Maximum Likelihood Estimation localization theorem using 3 PlutoSDRs and were able to create location estimates in real time. These estimates were plotted on a 2D x-y plane and updated live.
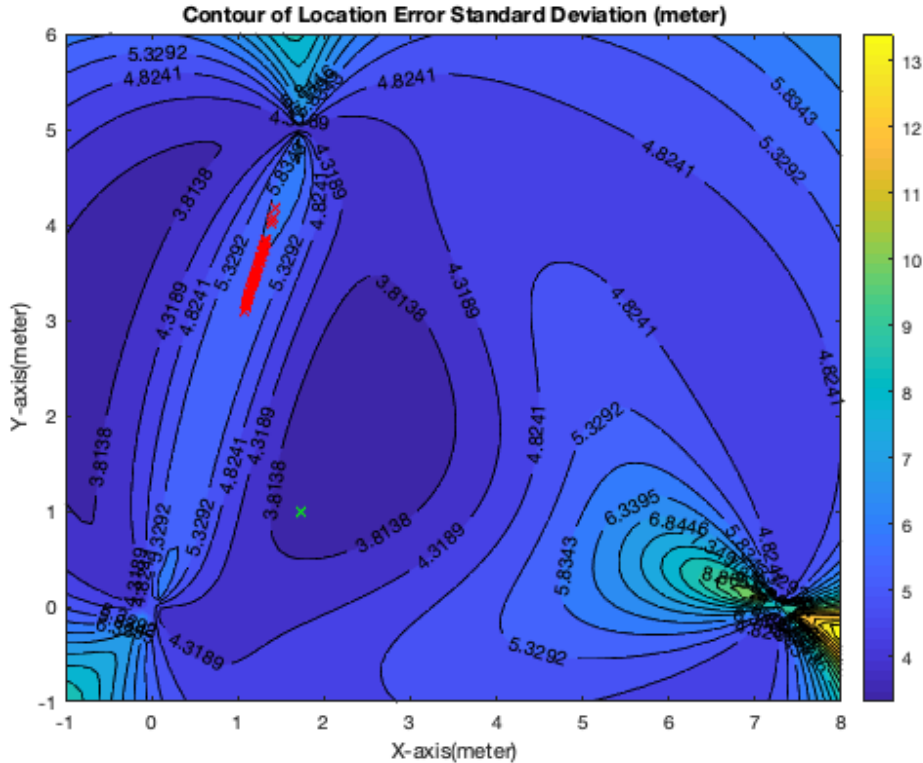

Figure 1: Standard Deviation Contours and Preliminary Results

The tests yielded results with a maximum error of 2.5 meters. Figure 1 shows the standard deviation contours plotted beneath the signal location estimates. The green X shows the actual location of the transmitting signal and the red Xs show 244 estimated locations of the signal produced by the system. The green X is within the contour representing 3.8 meters of standard deviation, showing that all 244 estimates are within the expected standard deviation. With more SDR devices collecting data, the accuracy will increase.

Table of Contents

List of Figures

List of Terms and Abbreviations

6LoWPAN: Pv6 over Low-Power Wireless Personal Area Networks

ADC: Analog-to-Digital Converter

DAC: Digital-to-Analog Converter

DFT: Discrete Fourier Transform

FCC: Federal Communications Commission

FFT: Fast Fourier Transform

FFTA: Fast Fourier Transform Algorithm

FPGA: Field Programmable Gate Array

IEEE: Institute of Electrical and Electronics Engineers

IFFT: Inverse Fast Fourier Transform

IoT: Internet of Things

ISM: Industrial, Scientific, and Medical

LAN: Local Area Network

LOS: Line of Sight

LSB: Least Significant Bit

ML: Maximum Likelihood

MLE: Maximum Likelihood Estimation

MSB: Most Significant Bit

PCB: Printed Circuit Board

RF: Radio Frequency

SDR: Software Defined Radio

SNR: Signal to Noise Ratio

TDoA: Time Difference of Arrival

ToA: Time of Arrival

1. Introduction

1.1. Motivation

In recent years, IoT has evolved considerably as technology becomes more ingrained in everyday life. IoT devices are used to control and monitor things such as thermostats, lights, vehicles, and more. Globally, IoT devices make up a large share of the technology market, with the largest growth in the areas of smart cities and self-driving cars [6].
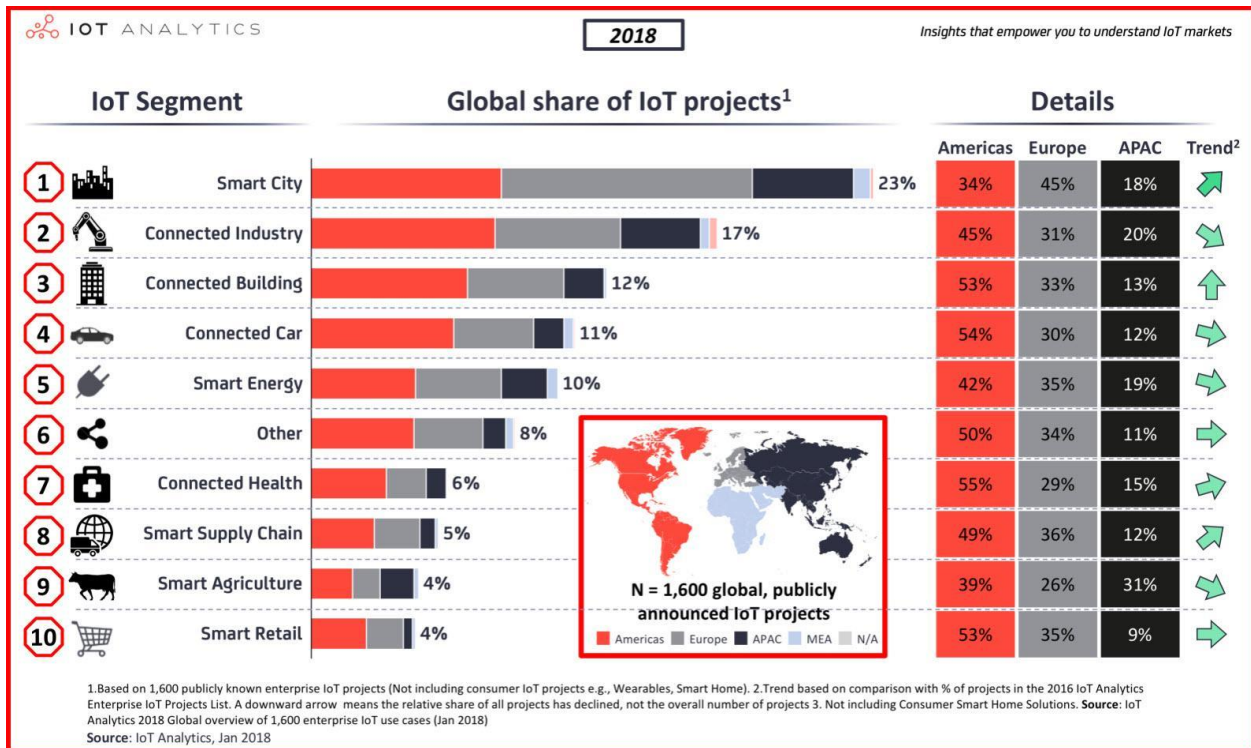
Figure 2: Understanding the IoT Market [6]

The IoT device market is expected to grow significantly over the next few years, specifically with independent consumers. The graph below shows a summary of the expected growth in each of the top markets.

IoT Market Revenue by Application in North America 2017-2022

Size of the Internet of Things (IoT) Market by Application in North America from 2017 to 2022 (in billions of U.S. dollars)
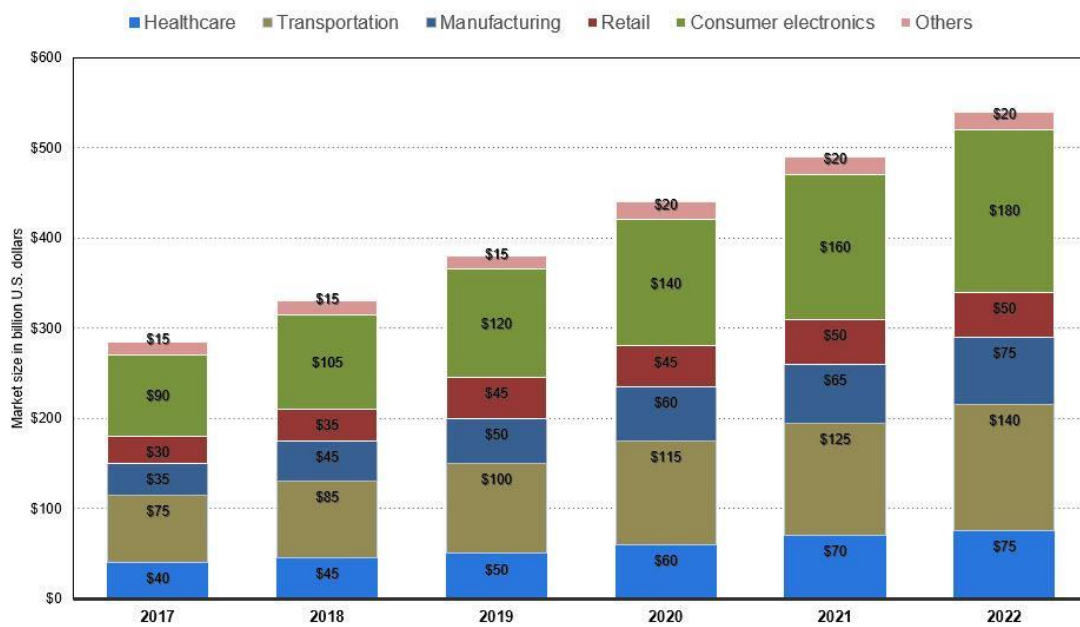
Figure 3: Size of IoT Market by Application [7]

IoT devices create new opportunities for the use of networked technology, however they also present security and congestion threats [1] [8]. Limited frequency bands are available for use by IoT devices as determined by the FCC, which can cause channel congestion [8]. Interference poses a problem for applications such as GPS radio astronomy, where the arriving signals are extremely weak [9]. Smart transmitting devices can also present security issues. Many IoT devices do not encrypt their communication [10] leaving them as easy targets for hackers to gain access to your network. IoT devices can also unintentionally emit signals outside of its intended frequency and interfere with other devices. Systems to monitor and localize sub-1 GHz transmissions do not currently exist, but are an excellent way to enforce policy, and ensure security.

1.2.   Current State of the Art

Most competing solutions exist at 2.4 GHz and are used for navigation for devices to know their own position in situations where GPS does not work, such as big cities [11] [12]. Currently top companies in the technology market do not have consumer products available for indoor localization, although Apple and Google have bought into the industry and are expected to release applications in the near future [2].

There are very few devices which can both localize signals and sense the signals. Typically, devices operate as spectrum analyzers which can determine which band a signal is coming from. One of these such devices is the 4.4Ghz Sound Hound device which retails for nearly $1000 and does not possess localization capabilities [3].

2. Background Research and Development

In order to develop a system which could effectively integrate indoor localization with signal sensing capabilities, a background in signal theory and wireless communication is needed. Also discussed is the basics of localization development.

2.1. RF Spectrum

The Radio Frequency (RF) spectrum is a portion of the electromagnetic spectrum that encompasses radio waves which are used for wireless communication.
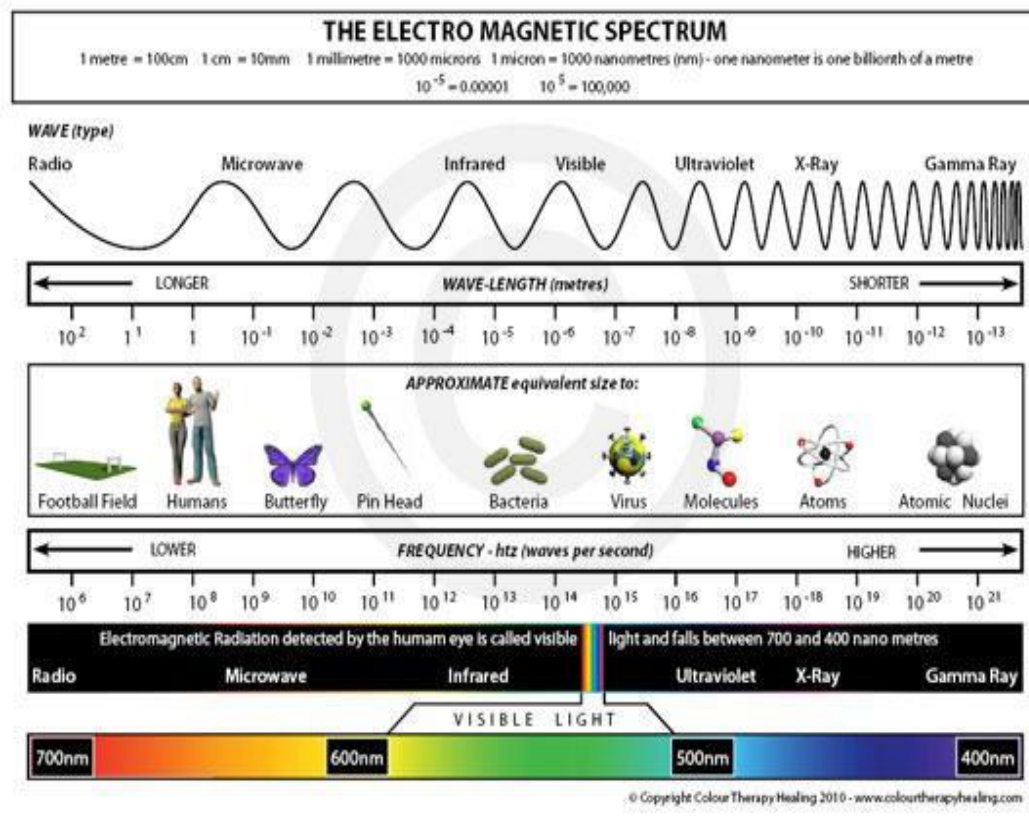
Figure 4: Diagram of RF Spectrum [13]

4

The RF spectrum is comprised of the lower frequencies of the electromagnetic spectrum from approximately 3 KHz to 300 GHz [14]. These waves are sinusoidal waves that travel at the speed of light. The speed of light is constant, so waves with lower frequencies have much larger wavelengths, while those with higher frequencies have much smaller wavelengths.

Radio waves are generated by applying an alternating current to an antenna which generates a proportional alternating electromagnetic field. When this electromagnetic field passes over the receiving antenna, a proportional but significantly smaller alternating current is induced in the antenna. The propagation characteristics of radio waves through any medium are governed by the Maxwell equations [15].

2.2. Propagation Characteristics

In free space, all electromagnetic waves conform to the inverse square law. The inverse square law states that the power density (p) of an electromagnetic wave is proportional to the inverse square of the distance (d) from the source, shown in Equation 1 below.

$$p \sim \frac{1}{d^2}$$

Equation 1

This means that as distance from the source of the transmission increases the power of the signal will exponentially decrease as the distance increases. Radio waves are also affected by reflections, refraction, diffraction, absorption, polarization, and scattering, which change the manner in which they propagate [16].

Various propagation theorems are used in radio transmission systems. Line of Sight (LOS) is when the radio waves travel in a straight line from the transmission source to the receiving

antenna. LOS is the only propagation method possible for microwave frequencies and above because those waves are not capable of passing around obstacles such as mountains. However, the low end of the microwave band is capable of traveling through building walls. When using a LOS transmission model, it is important to pay attention to the Fresnel zone, which defined as an area that is covered by an imaginary ellipse with the transmitting and receiving antenna as the foci of the ellipse [17].
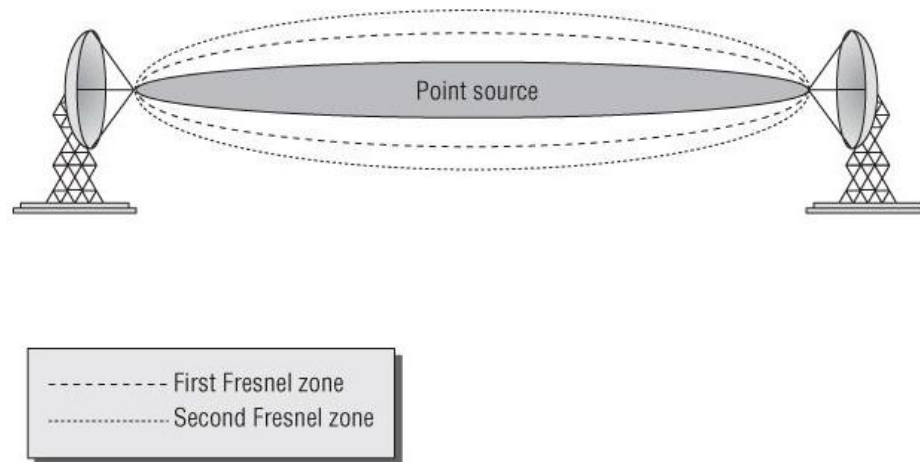
Figure 5: Fresnel Zone [18]

Any obstruction within the ellipse can degrade the signal. In LOS communication some amount of radiation does not propagate directly to the receiving antenna. This stray radiation can then reflect off of objects and radiate to the receiver with a phase delay, which may lead to destructive interference.

At lower frequencies, due to diffraction, radio waves can bend around obstacles and travel beyond the horizon, no longer requiring LOS propagation. Ground waves between 30 KHz and 3000 KHz can follow the contour of Earth. As a wave's frequency increases, absorption by molecular resonances causes significant power loss in radio propagation, particularly from water

(H_2O) and oxygen (O_2). This fault in higher frequencies makes lower frequencies a very valuable resource.

2.3.   Frequency Utilization

As discussed previously, the RF spectrum is a finite portion of the electromagnetic spectrum. As such, the utilization of the frequencies within the spectrum is limited because each device that uses wireless technology requires an unobstructed transmission on the frequency. The utilization is particularly limited in the lower frequency bands due to their higher quality propagation characteristics [19]. The uses of the RF spectrum include mobile phones, television broadcasting, and space research. In order for the RF spectrum to be used effectively, the U.S. government allocates different bands of frequencies within the RF spectrum for different uses. All governments have spectrum allocations practices; however, this project focuses only on the U.S. standards. Occasionally, additional uses of frequency bands are authorized, but only if there is significant need and national interest [20]. The most recent version of this allocation is from 2016, and is shown below in Figure 7.
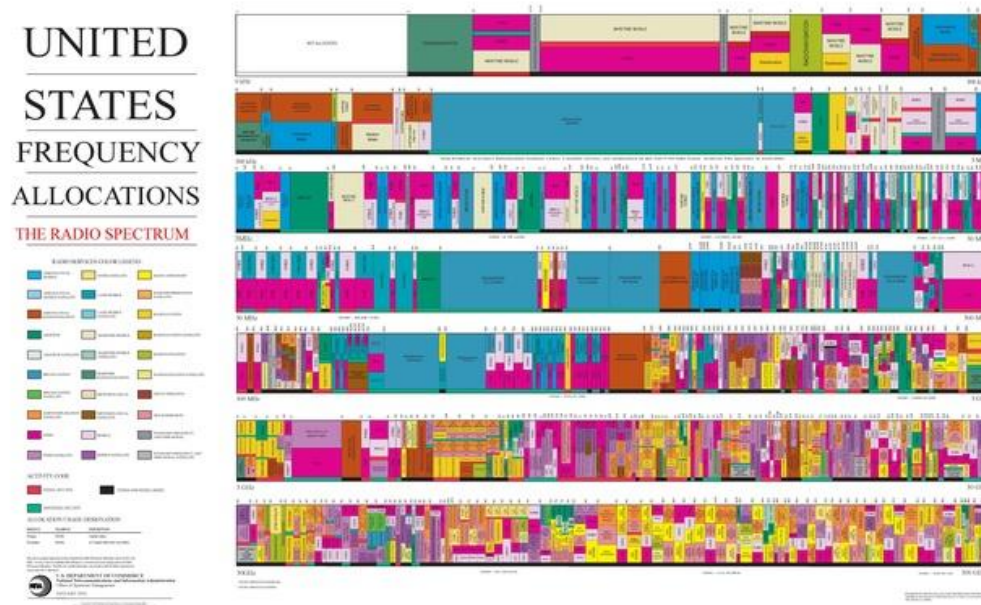
Figure 6: USA Frequency Allocations

The types of frequency bands allocated can effectively be split into three categories: unlicensed bands, licensed bands, and forbidden bands [21]. Unlicensed bands can be used by anyone in compliance with the rules of the band, licensed bands can only be used by the owner of the license, and forbidden bands cannot be used as they are reserved for use by groups such as the military [21].

2.4.   Time and Frequency Domains

In 1882, Joseph Fourier showed that time varying functions could be written as an infinite sum of harmonics, which is known as the Fourier series. His discovery shows that the time and frequency domains are alternate ways of representing the same signal and the Fourier transform is the relationship between the two domains [22].

The Fourier transform has four different forms, Fourier transforms, Fourier series, discrete Fourier transform (DFT), and Fast Fourier transform (FFT). There is also the inverse FFT (IFFT)

which converts the frequency domain information to the time domain. The equation for the Fourier transform shown in the following equation.

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$

Where x(t) is the time domain signal, t is the time variable in seconds, and ω is the frequency variable in radians per second. The DFT treats the time domain and frequency domain signals as if they were periodic, and the sampled pattern is just one period of a signal that is repeated indefinitely. One problem with the DFT is the time complexity of the algorithm is $O(n^2)$, which in the past made it difficult to compute the Fourier transform in real time. This changed with the optimization of the FFT.

The FFT was popularized in 1966 in a publication by Cooley and Tukey [24]. The FFT runs in $O(nlogn)$, as this implementation of the discrete Fourier transform takes advantage of a divide and conquer approach that recursively breaks down the DFT into 2 components of size N/2 at each recursive step, along with O(n) multiplications of complex roots. The output of the FFT of the FFT is a series of M/2 points in the frequency domain, where M is the number of time-domain samples (typically a power of 2). The total frequency range covered by the FFT is DC to Fs/2, where fs is the sampling rate. The resolution, which is also known as the spacing between the points of the FFT, is fs/M.

The theoretical noise floor of the FFT is the theoretical Signal to Noise Ratio (SNR) plus the process gain of the FFT which is $10log_{10}(\frac{M}{2})$, so the more samples M, the lower the noise floor. The benefits trail off exponentially, as we are taking the log. Figure 8 below is a diagram illustrating the FFT noise floor:
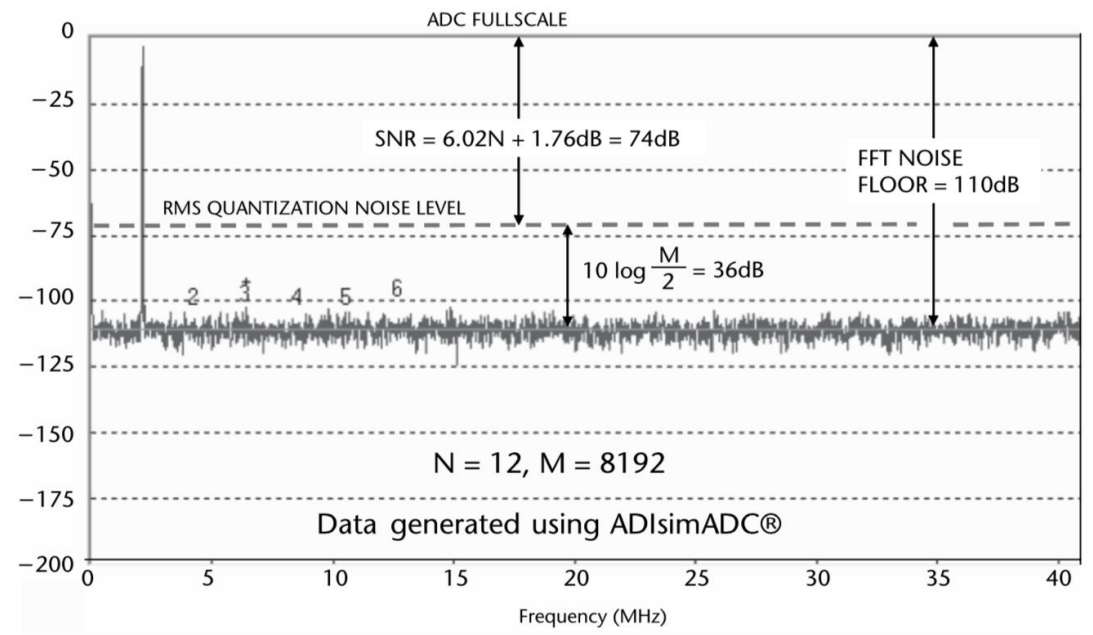
Figure 7: FFT Noise Floor

Continuous-time analog signals must be sampled at a discrete rate with an analog to digital converter (ADC) in order to be logically processed [25]. An example would be a program which computes the FFT or any other application where a digital representation is required. There are a few things to keep in mind to ensure an accurate representation of the analog waveform.

Analog to digital converters only have a finite number of bits, a 12-bit ADC can only represent analog signals at 2^12 different levels. All values between each level will be rounded to the least significant bit (LSB). The maximum error of an ideal ADC is $\pm \frac{1}{2}$LSB. because any signal value that ½ above the LSB will be rounded down, and any value below ½ LSB will be rounded up. Ideally, more bits mean more precision and less quantization error, however that could come at the expense of power, cost, and/or performance.

This is one of the most important parts of sampling, if not the single most important. A signal must be sampled at minimum the Nyquist rate [26], which is double the maximum frequency

of the signal. This is because in order to accurately represent the signal, it is required to have at least two data points within the smallest period, or that frequency will be lost in the sampled signal. Uniformly sampling a signal creates images of the sampled input signal that are periodic with the sampling frequency fs. For this reason, it is important to filter the desired band before sampling, as any frequency component that is outside the Nyquist bandwidth will be aliased back into the first Nyquist zone.

2.5.　IoT Devices

An Internet of Things (IoT) device can be defined as anything connected to the internet. However, in recent years, the idea of the IoT has evolved considerably as increasingly more everyday devices are created to be networkable, where a physical object also has a virtual existence. These objects include things such as thermostats, lights, vehicles, and more. It's estimated that by 2020 there will be 15 billion IoT devices, not including phones, PCs, or tablets [27]. The idea of IoT devices in relation to privacy has become a larger discussion in recent years with many voicing the concern that technology is functioning ahead of the policy. Brian Solis of Altimeter Group who has conducted studies on IoT privacy implications has stated, "We are looking at a future in which companies will indulge in digital Darwinism, using IoT, AI [Artificial Intelligence] and machine learning to rapidly evolve in a way we've never seen before," [28] The Institute of Electrical and Electronics Engineers (IEEE) has developed certain privacy policies in relation to IoT devices, however these documents are still evolving with new developments [29] Various standards and protocols related to IoT are discussed in this report.

IoT devices are wireless technologies; they operate within the RF spectrum. These devices are only capable of utilizing specific frequency bands that are available to them or allocated to them. In this section we discuss a few key IoT frequency bands of interest.

433 MHz is an unlicensed band which typically spans from 433.05 MHz to 434.79 MHz and is shared by amateur radio, low-powered applications, and the radiolocation service. The radiolocation service is the only primary   service in this band [30]. For IoT devices using this band, there are certain limitations due to FCC (Federal Communications Commission) regulations. Regulation 10CFR47 Part 15.231 specifically regulates operation at the 433 MHz band. Polling transmissions are allowed; however, they cannot exceed two seconds of transmission time per hour, with a maximum rate of polling for 300 microseconds every 10 seconds. Additionally, the transmit power is limited to approximately 10,000 microvolts per meter at 3 meters [31]. Due to these limitations, some IoT devices use this band however many are unable to. One widely used application of 433 MHz band is garage door openers.

Also known as the ISM (Industrial, Scientific, and Medical) band at 915 MHz, which spans from 902 MHz to 928 MHz, this has become popular for establishing wireless connections with short-range wireless devices [32]. This band is also unlicensed which makes it popular for IoT applications. IoT devices operating in this band fall under the regulation FCC Part 15, and an overview of it states that, "Firstly, the device may not cause harmful interference, and secondly the device must accept any interference received, including interference that may cause undesired operation. Hence, there is no guaranteed quality of service when operating a Part 15 device" [33]. There are also additional FCC regulations for this band depending on the type of IoT device used.

2.4 GHz is an additional unlicensed ISM band, which differs in its benefits and drawbacks. A 915 MHz system will more easily broadcast through a multi floor house; 2.4 GHz systems will ideally have a longer range. [34] The 2.4–2.4835 GHz band is regulated by FCC sections 15.247 and 15.249. The 915 MHz band and the 2.4 GHz band have similar regulations, with the same power limitations for each. An advantage of the 2.4 GHz band is that it is unlicensed worldwide,

so a device may operate at that band regardless of its locations, and it has more channels available than the 915 MHz band due to having a larger bandwidth. Disadvantages for this band include "increased cost and current consumption of the active components, reduced propagation distance for the same power, and increased band congestion due to such systems as Bluetooth and wireless internet" [35].

The 5 GHz frequency band is a super high frequency band used for things such as radars, mobile phones, and commercial wireless LAN (Local Area Networks) [36]. The FCC has released a "5G FAST Plan" which is said to be a "comprehensive strategy to Facilitate America's Superiority in 5G Technology". The main components of this plan are to push more spectrum into the marketplace, update existing infrastructure policy, and to modernize outdated regulations. [37] For IoT devices this band offers potentially higher speeds.

2.6.   Protocols

The IEEE standard 802.15.4, "Standard for Low-Rate Wireless Networks", defines the protocol for devices using low-data-rate, low-power, and low-complexity short-range (10 meters or less) radio frequency transmissions in a wireless personal area network. Devices in this network can use either a 64-bit IEEE address or a 16-bit address assigned during association. Wireless links under this standard are limited to operating in unlicensed ISM frequency bands, which includes 2.4 GHz and 915 MHz [38].

The pv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) standard, defined in RFC 6282 by the Internet Engineering Task Force, specifies the operation of IPv6 over the IEEE 802.15.4 standard [39]. It is based on the utilization of low powered IP-driven nodes and a large mesh network [39] This makes 6LoWPAN a great option for IoT due to the optimization

of the IPv6 packets which can be carried efficiently within small link layer frames, such as those defined by IEEE 802.15.4.

There are a few lightweight layer protocols that are fairly well known such as the Zigbee protocol and Z-wave, but these are all built on top of 6LoWPAN. [40]

2.7. Software-Defined Radio

A Software-Defined Radio (SDR) is a radio in which some or all of the components are implemented in software as opposed to with hardware [41]. While the functionality of SDRs, transmitting and receiving data, is similar to their more traditional analog radio counterparts, they perform multiple tasks digitally and simultaneously, using programmable devices such as FPGAs. The advantages to this include that they are relatively inexpensive and are highly flexible due to their configurability.
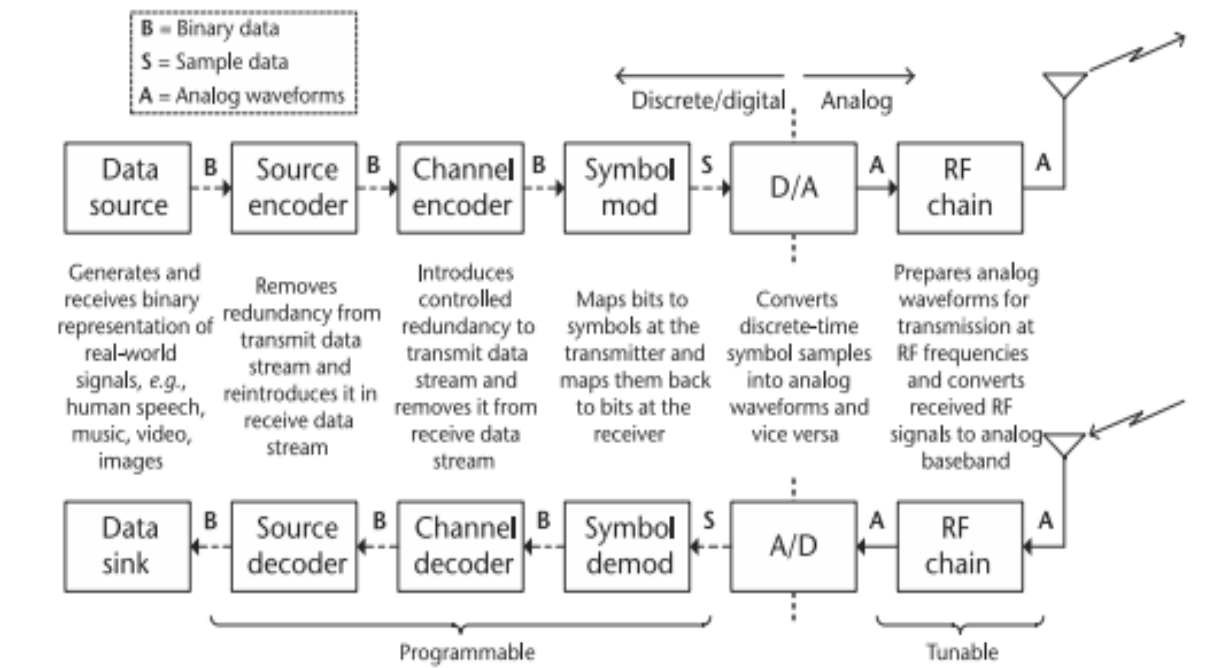
Figure 8: Data Types and Processing Reference[11]

For receiving data, an SDR must intercept an analog signal being transmitted, convert it from an analog signal to a digital signal, determine its characteristics such as signal amplitude, carrier frequency, or phase from other noise or distortion, and return the original signal back in a binary representation. Conversely, to transmit a signal an SDR must digitize it to use the binary representation if the signal is not already digital and convert that representation to an electromagnetic sinusoidal waveform with the proper amplitude, carrier frequency, and phase characteristics [42].

2.8.   RTL-SDR

The RTL-SDR is a low-cost, ~$25, USB dongle that is commonly used by SDR hobbyists as well as professional engineers. Originally, the RTL-SDR was created when Antti Palosaari, Eric Fry and Osmocom discovered that television tuners with the RTL2832U chip could be used to acquire the raw I/Q data on the chipset. Since its conception the community-developed software used for it to access the radio spectrum has been shared at no financial cost. [43]

Table 1 below shows the frequency ranges for each chipset. It should be noted that for each chipset the reception range varies slightly.

Table 1: Chips and Frequency Ranges

Chip	Frequency Range
R820T and R820T2	24 MHz to 1766 MHz
E4000	52 MHz to 2200 MHz, gap between 1100 MHz to 1250 MHz
FC 0013	22 MHz to 949 MHz

FC 0012	22 MHz to 1100 MHz

Additionally, while the maximum sample rate of the device is 3.2 mega-samples per second, that is an unstable rate for it to operate at and it may drop samples, so 2.4 mega-samples per second is the highest recommended sampling rate. [44]

2.9.   ADALM-PLUTO

The ADALM-PLUTO is an SDR device created by Analog Devices to be used as a low cost education tool for students, hobbyists, and industry engineers alike. It can be used to explore different electrical engineering fundamentals related to SDR and RF signal processing. [45]

The PlutoSDR is based on the Analog Devices chip AD9363 which has the main attributes of an RF 2×2 transceiver with integrated 12-bit DACs and ADCs. It is tunable up to 20MHz and has 325 MHz to 3.8 GHz of bandwidth [45].

In addition to the AD9363, the PlutoSDR also has the Xilinx Zynq 7010 for an onboard FPGA to handle processing of FFT's and other necessary calculations. This small board was chosen due to size and low number of pins for easy integration into the PCB, however Analog Devices has stated this could be hacked and replaced with a larger (and most likely more expensive) FPGA depending on the end user's needs. [45]

The following image shows typical utilization of the onboard FPGA:

Resource	Utilization	Available	Utilization %
LUT	6422	17600	36.49
LUTRAM	1066	6000	17.77
FF	13162	35200	37.39
BRAM	2	60	3.33
DSP	70	80	87.50
IO	51	54	94.44
BUFG	3	32	9.38
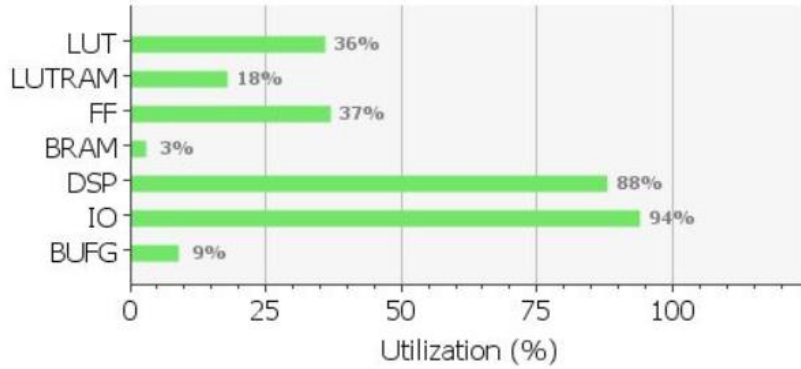
Figure 9: PlutoSDR Data Usage

It is possible to expand the availability on the board based on the use desired. For example, if the PlutoSDR is used only as a receiver, much of the DSP functions, which are by default used for transmission, can be cleared to allocate additional space for onboard processing of received signals.

The PlutoSDR is able to interface with MATLAB, Simulink, GNU Radio or custom C, C++, C#, or Python environment on a host (x86) Windows, Linux or Mac or embedded Linux platform (Raspberry Pi, Beaglebone, 96boards.org, over USB. [4] On the onboard FPGA itself it is programmable in Linux, although reprogramming the onboard device is slightly more complicated than interfacing with the above-mentioned programs via USB. Depending on the application, different software configurations can be adjusted.

2.10. GNU Radio and MATLAB

GNU Radio is a tool that provides signal processing blocks to interface with SDRs. The use of the GNU Radio development toolkit is entirely free as well as open-source. By performing the signal processing for an SDR, GNU Radio is able to receive and transmit data with the SDR hardware. It's an extremely powerful tool for hobbyists as well as professionals, as it has available "channel codes, synchronization elements, equalizers, demodulators, vocoders, decoders, and many other types of blocks which are typically found in signal processing systems" allowing for the creation of real time SDR systems. [46]

An example of a GNU Radio flowchart implementing its signal processing blocks with the PlutoSDR is shown below in Figure 11.
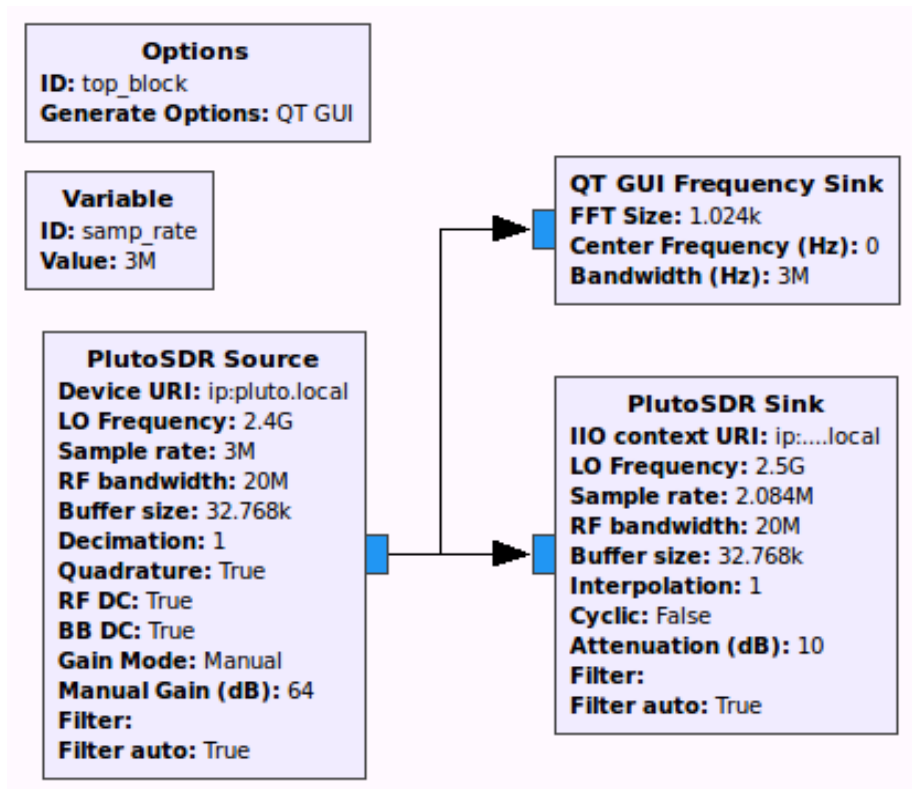

Figure 10: GNURadio Block Diagram

2.10.1. MATLAB

MATLAB is an excellent tool used to interface with SDR devices. MathWorks, the maker of MATLAB, has developed an entire communications toolbox dedicated to different software and simulation support. MathWorks has developed a set of test programs and support packages for both PlutoSDR and the RTL-SDR devices. The Communications Toolbox Support Package for Analog Devices PlutoSDR Radio (PlutoSDR) enables users to implement MATLAB and Simulink to prototype, verify, and test practical wireless systems. [47] These capabilities are also available for use with the RTL-SDR, for example, it can be used to receive and process wireless signals such as FM radio, airplane surveillance signals (ADS-B), and signals from smart meters (water or energy metering devices). [48]

The user-friendly examples make MATLAB a good starting point or method to verify results, however it cannot run without an operating PC. Many examples using both the RTL-SDR or PlutoSDR implement the devices' receiving abilities as spectrum analyzers. This is shown here using the PlutoSDR in Figure 11.
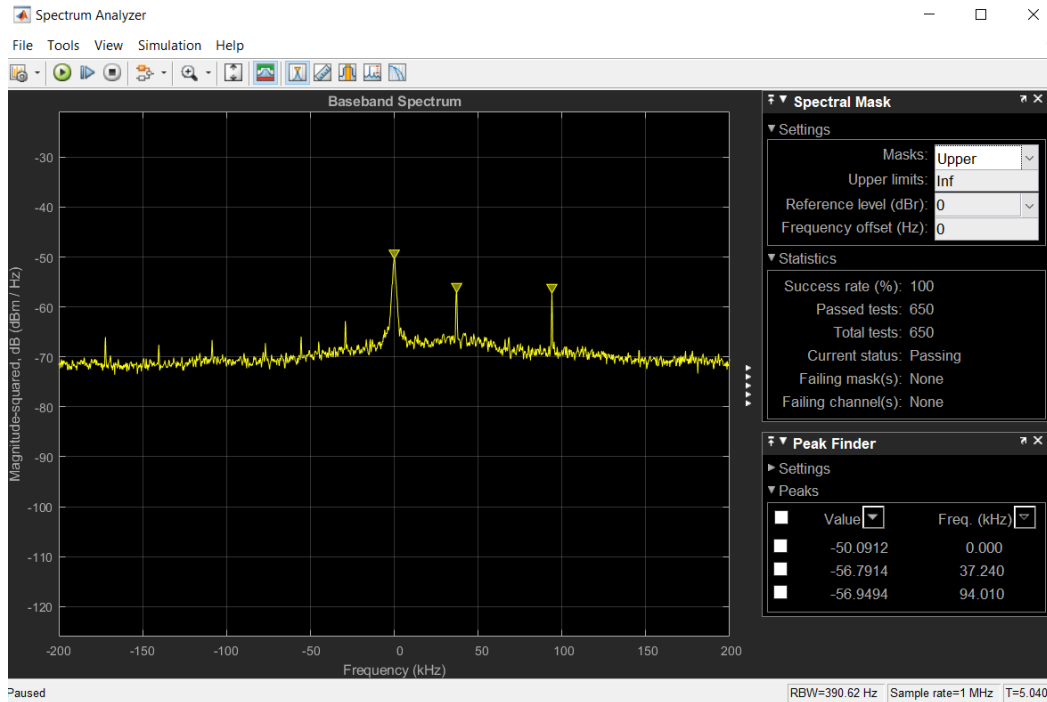
Figure 11: Spectrum Sensing Using SDRAngel and PlutoSDR

2.11. Localization

There are two localization methods that are used in wireless localization, the first is time of arrival (TOA). TOA is widely used in outdoor rural locations, and is the system used by the United States Global Positioning System (GPS). The second method is to use received signal strength (RSS). RSS has become more popular for dense urban environments where TOA suffers from multipath [12].

Time of arrival is used when the transmission time is known. At the receiver when it receives the signal it subtracts the time the signal was sent. This delta value becomes the time of flight and is multiplied by the speed of an electromagnetic wave in free space, c = 2.99E10 meters per second in order to determine the distance traveled. Given c, the speed of an electromagnetic wave, t_1 for the timestamp of when the signal was sent, t_2 for the time the transmission was

received, and X_i, Y_i for the known coordinates of the receiver, the coordinates of the transmitter can be determined using equation 4:

$$c(t_2 - t_1) = \sqrt{(X_i - x)^2 + (Y_i - y)^2}$$

Equation 3 [12]

By solving this system of equations using three or more receivers it is possible to determine the origins of the transmitter. However, there are two problems which arise. First the transmission may not have a timestamp. Second this system is highly dependent on the stability and precision of timing. We are measuring the speed of light which travels about ⅓ of a meter every nanosecond. So every nanosecond of inaccuracy translates to ⅓ of a meter in error.

Time Difference of Arrival takes care of one issue with ToA as the original transmission no longer requires a timestamp. Instead this system uses a set of receivers, and measures the time it takes for each of the receivers to receive the signal [12]. The time measurement at each of the receiving sites is then subtracted from the first received signal. Provided an unknown source location (x, y), the known receiver locations are (X_n, Y_n). All distances between the receivers are known. The distance between the sources and the nth and $(n+1)^{th}$ receiver can be modeled as:

$$d_{n+1,n} = c(T_{n+1} - T_n) = |d_{n+1} - D_n| = \sqrt{(x - X_{n+1})^2 + (y - Y_{n+1})^2} - \sqrt{(x - X_n)^2 + (y - Y_n)^2}$$

Equation 4 [12]

This technique suffers the same crucial synchronization requirements as TOA.

Received signal strength (RSS) uses the measured signal power at each of the receivers. The RSS path-loss model can be described as:

$$P_i = P_0 - 10\alpha log_{10}(d)$$

Equation 5 [12]

Where Pi is the averaged signal strength in dB received at the ith sensor, P_s is the transmitted signal strength in dB, α is the path-loss factor, and d_i is the distance between the source and the ith sensor.

2.11.1. Weighted Centroid

The basic weighted centroid algorithm always estimates the location of the target as the middle of all references. Basic properties of the algorithm are simplicity and stability. In addition, centroid algorithm works for any number of reference points and it does not rely on the accuracy of measured RSS [12].

In a WPS weighted centroid algorithm, the RSS readings from *M-Aps,* with estimated locations, (xi , yi); i = 1, 2 . . . , L, the locations of the device, (x, y), is estimated by:

$$\hat{x} = \sum_{i=i}^{M} p_i^x * x_i$$

Equation 6 [12]

$$\hat{y} = \sum_{i=i}^{M} p_i^y * y_i$$

Equation 7 [12]

Where p_k^x and p_k^y are the *weights* of averaging along the X and Y axis for localization. These are the probabilities of having the device in location of a specific AP. This probability is a function of RSS reading from each measurement [12].

2.11.2. Maximum Likelihood Estimate (MLE)

We use the path loss model $P_r = P_0 - 10 * \alpha * log_{10}(r) + X(\sigma)$ to define a rim shaped confidence region for RSS based ranging. For a given confidence γ (eg 90%), we first calculate the fade margin, F_σ, for variance of shadow fading, σ, and the required confidence from:

$$F_\sigma = \sqrt{2\sigma} * erfc^{-1}(1 - \gamma)$$

Equation 8 [12]

Then, we use the fade margin and the measured RSS, P_M, to calculate the radius of two circle defining the inner and outer boundaries from:

$$r_1 = 10^{\frac{P_0 - F_\sigma - P_M}{10\alpha}}$$

Equation 9 [12]

$$r_1 = 10^{\frac{P_0 + F_\sigma - P_M}{10\alpha}}$$

Equation 10 [12]

The area between the two circle designates a region with the desired certainty. This single receiver MLE method shown above is represented visually in the figure below:
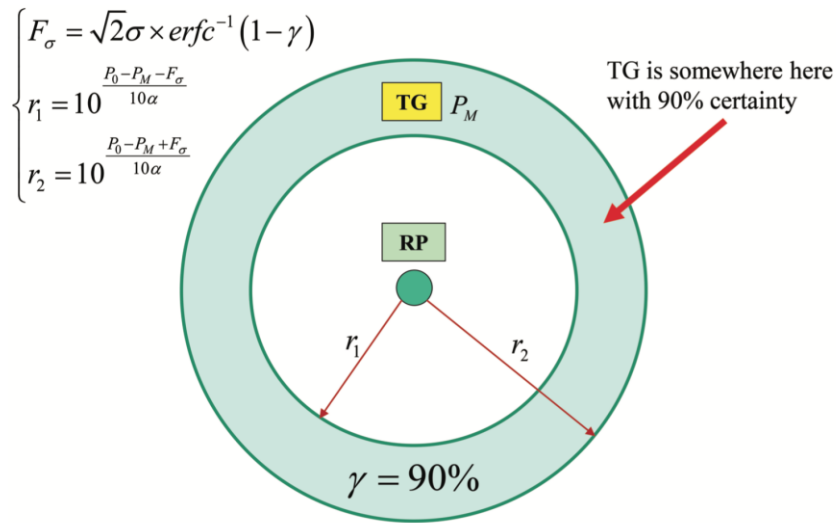
$$\begin{cases} F_\sigma = \sqrt{2}\sigma \times erfc^{-1}\left(1-\gamma\right) \\ r_1 = 10^{\frac{P_0 - P_M - F_\sigma}{10\alpha}} \\ r_2 = 10^{\frac{P_0 - P_M + F_\sigma}{10\alpha}} \end{cases}$$

TG is somewhere here with 90% certainty

TG P_M

RP

r_1 r_2

$$\gamma = 90\%$$

Figure 12: MLE Visual Description [12]

By creating the confidence regions around the know location of the receivers we get overlapping regions. As seen in the following figure:
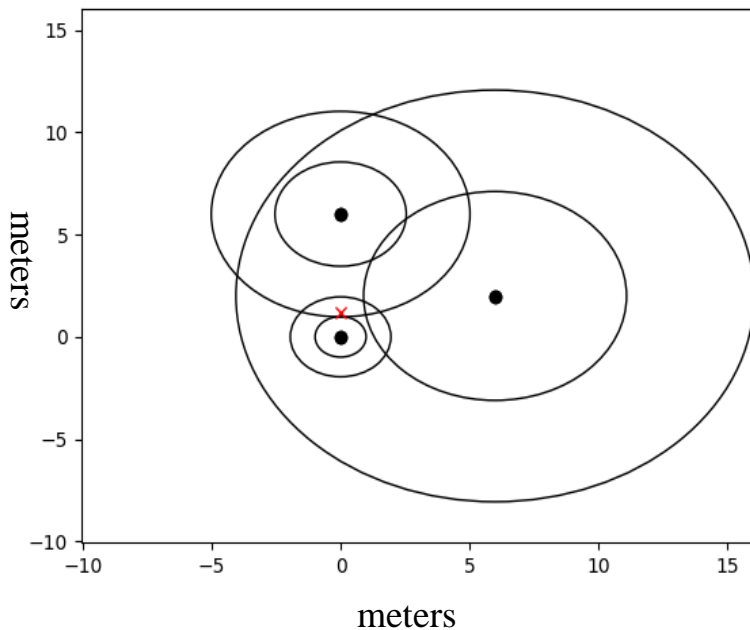

Figure 13: MLE 2-Dimensional Diagram

The area where all 3 MLE intervals overlap becomes the estimate of the area that we would estimate the receiver is located. The red X is the location of the actual receiver in this example. It is clearly seen that the red X is almost directly in the center of our overlapping estimate region.

The way that a single point estimate is created is by finding the intersection points of all the circles, then going through each intersection point and determining if that point is with in each of the MLE regions. The resulting points that lie within all the receivers MLE regions (in the example above the 3 intersections closest to the red x) are then averaged to get a final single point estimate.

3.   Proposed Approach

3.1.   Problem Definition

Industrial wireless applications often share communication channels with other wireless technologies and communication protocols. This coexistence produces interferences and transmissions which require appropriate mechanisms to manage retransmissions. Nevertheless, these mechanisms increase the network latency and overhead due to retransmissions. The loss of data and the measures to handle them produce an undesirable drop in the reliability and hinder the overall robustness and timeliness of the network. Interference avoidance mechanisms, such as frequency hopping techniques, reduce the need for retransmissions due to interferences but they are often tailored to specific scenarios and are not easily adapted to other use cases. On the other hand, the total absence of interference avoidance mechanisms introduces a security risk because the communication channel may be intentionally attacked and interfered with to hinder or totally block it.

The IEEE standard 802.15.4 defines the protocol for devices using low-data-rate, low-power, and low-complexity short-range radio frequency transmissions in a wireless personal area network. Further, the 6LoWPAN standard specifies operation of IPv6 over the IEEE 802.15.4 standard.

3.2.   Solution

Using low cost SDR devices to characterize and localize IoT band electrospace usage below 1 GHz and store signal energies, specifically looking at the 908 MHz band. This will enable an open source, low cost solution to emerging IoT signal sensing and localizing needs.

3.3.   System Structure

This system will consist of four PlutoSDR devices, which are each receiving and measuring signals. Each PlutoSDR is powered by a wall power adapter, and is connected to a USB WiFi adaptor via a USB OTG cable. A 2.4 GHz WiFi router will provide a wireless network connecting the PlutoSDR devices to a host machine.
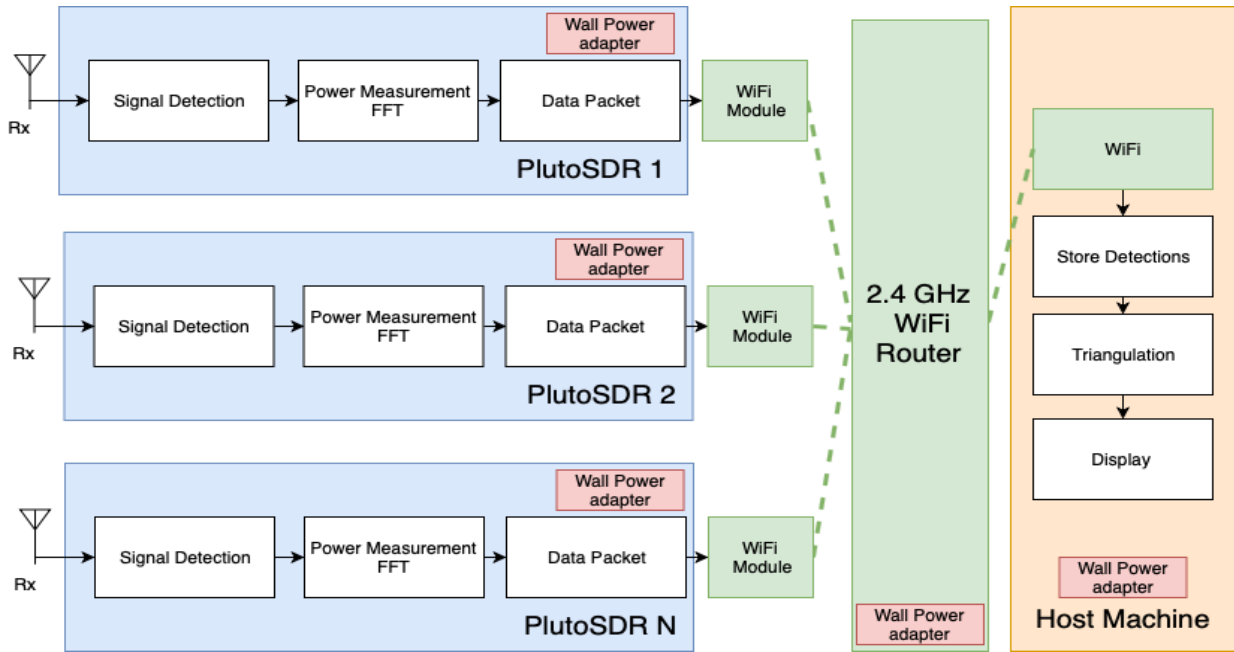
3.3.1. Hardware

Figure 14: System Block Diagram

The PlutoSDR will serve as receiving device, scanning frequency bands, measuring received signals, and communicating with a host machine. We plan to use a total of four PlutoSDRs in our final design, however the design could be modified to operate with any number of PlutoSDR devices.

The PlutoSDR is a portable, self-contained RF learning module which costs $99 for students and $149 for other buyers. Specifications that are pertinent to our uses of the device are listed in the Appendix.

From a hardware system standpoint, the PlutoSDR will be powered via a wall power adaptor from the power-only USB port. We chose wall power as opposed to battery power so that the device will be known to consistently have sufficient power, given normal circumstances, without needing to rely on a variable battery. The device will additionally be connected to a USB WiFi adaptor via a USB OTG cable.

As can be seen from these specifications, the PlutoSDR transceiver will fulfill our goal of characterizing IoT band electro space usage below 1 GHz. The USB 2.0 capabilities will allow the device to be wirelessly networked via a WiFi dongle. The onboard FPGA is accessible and programmable, allowing for us to alter it for our purposes.

For the USB WiFi adaptor that will be connected to the PlutoSDR via a USB OTG cable, we will be using the TP-Link TL-WN725N N150, which costs $9.99. This WiFi adaptor gives devices connected to it wireless networking capabilities. Specifications that are pertinent to our uses of the device can be found in the appendix.

Since this adaptor complies with IEEE 802.11n standards, the module will be supported by the PlutoSDR. This module has also been tested with the PlutoSDR by previous developers. The USB OTG adapter cable which will connect the TP-Link TL-WN725N N150 to the PlutoSDR, we will be using a standard micro-USB plug type B to USB female type A adapter by SODIAL(R). This adapter has a 15cm cable length and costs $1.71. The USB to wall power adaptor which will assist in powering the PlutoSDR, we will be using a standard USB to wall power adapter, which costs $6.99.

For our project, we will not be specifying an exact host machine or 2.4GHz WiFi router to be used. It is assumed in this configuration that our device will simply be connected to an existing system. There are not specific requirements which vary significantly considering typical systems which need to be considered for our device to be operable.

3.3.2.   Cost

The cost per unit of each device will be less than $150. Compared to other SDR devices on the market, this is very good for our proposed solution. It is assumed a server or laptop configuration already exists to have the data transmitted to over WiFi.

Device	Cost	Total Cost per Unit
PlutoSDR	$99	
WiFi dongle	$9.99	$117.69
Wall power adapter	$6.99	
USB OTG Cable	$1.71	

4. Implementation

Our implementation consists of 4 separate components. The PlutoSDR embedded software, the Python server, the web client, and the GNU Radio transmitter script. The following sections will describe how each of these components work together.

4.1. PlutoSDR Embedded Software

Instructions for installing all of the libraries and dependencies used to compile software to run directly on the PlutoSDR itself can be found on gnuradion.org [49]. This presentation gives detailed instructions on how to cross compile software to run on the embedded ARM A9 processor.

Below are descriptions of notable libraries used in the Embedded code, that would not typically be used in other applications:

Iio is a library written by Analog Devices to interface directly with hardware. Analog Device's has extensive documentation on how iio works [50]. Ad9361 is a libiio wrapper specifically written for the AD9361 ultra-wide band RF chip designed by analog devices. This chip as the exact same interface at the AD9393 chip which is installed on the PlutoSDR. This library wrapper makes it easier to interface with the hardware, and adds some buffer utilities that are commonly used in SDR applications. [50] FFTW is short for Fastest Fourier Transform in the West. This library is portable and supports many different architectures. On platforms that support it, it uses the neon instruction set. This library is standard for computing an FFT as fast as possible on any piece of given hardware [51].

The PlutoSDR embedded code is fairly straightforward in its objectives. The software needs to sample signals in real time, and perform an FFT on a large number of samples in order to get an accurate frequency representation. It will also requires being able to identify the power estimate of the signal, and send the signal data to the server.

- **Initialization:** The Initialization phase is a function call to 'initialization', which sets the sampling rate, the center frequency, and the bandwidth of the filters. It also processes all the command line arguments . The most used command line argument is setting the device identification, which is used by the host software to associate samples with the PlutoSDR which they arrived from. Initialization is also where the iio.h and AD9361.h libraries are used to initialize all of the hardware settings in each PlutoSDR. All command line arguments can be found on GitHub at https://github.com/williambannas/MQP, they can also be found by running the sampling script with the argument "- -help".

- **Sampling thread:** The sampling thread collects IQ samples from the PlutoSDR in real time, and stores them in a circular buffer. This buffer is four times the size of each Fourier transform.

- **Grab Samples:** This step of the code copies 2048 IQ samples from the circular buffer. The copied samples are then removed from the buffer which ensures that the samples do not get overwritten before being processed.

- **Perform FFT:** The FFT is performed on the copied samples by processing them using the fftw3 library. This gives the cross correlation of every sample. A power measurement is obtained by taking each bin of the FFT, squaring it, and taking the 10 log-base 10 of that value, represented in the following equation:

$$10 * log_{10}(FFT_i{}^2)$$

Equation 11

- **Identify Max Bin:** This is done by going through the power measurement of each bin of the FFT, and finding the maximum. The reason that this is done is because we are only localizing on a single tone, so we expect the power of the signal to be within a single bin.

- **POST Request:** In order to send the data from the PlutoSDR to the server a POST request is used. The POST request is a JSON formatted request. The format of each request is:

```
{   "PlutoID": 1,
    "RSS": -40,
    "AGC": 70,
    "RSSI": 140,
    "bin": 16   }
```

Where the PlutoID is the ID assigned to each PlutoSDR, the RSS is the power recorded from the max power bin, AGC is current gain setting for the auto gain control, RSSI is the power of the entire band that is read off of the AD9363 chip using Libiio, and the bin is the frequency bin that the RSS value corresponds to. This format allows the data to be readily usable by the host machine.
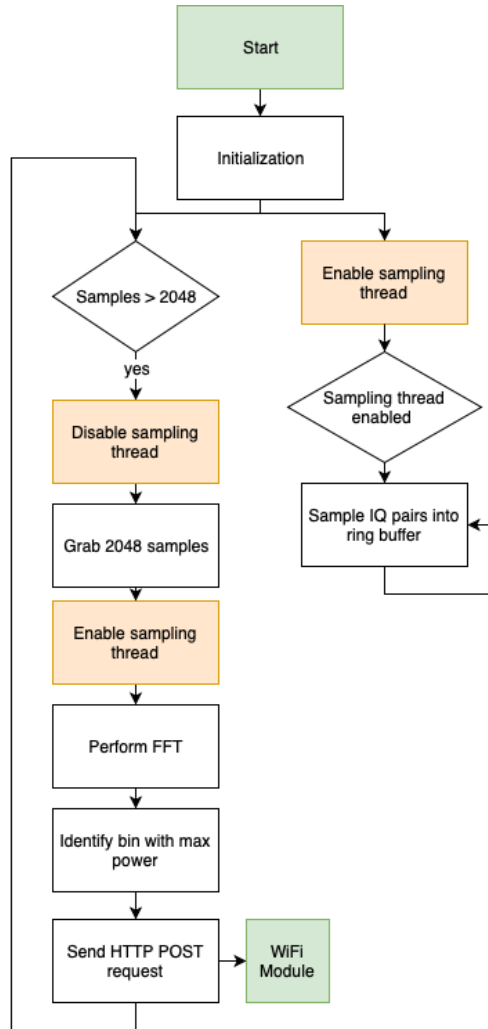

Figure 15: Implementation of Code Block Diagram

4.2. Server

The server is implemented in python, and uses the Flask web framework to create a simple web server. The first task that the server executes is the starting the PlutoSDRs. This is done through a script that secure copies the compiled sampling program to each of the PlutoSDRs. Then SSHs into each of the PlutoSDRs and starts the sampling scripts as a background process. All the PlutoSDRs then start to stream samples over to the server which are stored in CSV files specific to each device.

Once there are more than 10 samples from each of the devices 10 samples are averaged together to help mitigate some of the noise, then the server preforms the MLE algorithm for each of the receiving devices.

When a `localization request` is made to the server, the server responds with the location estimates that were calculated from the MLE algorithm. One important note to make about our implementation is that we subtract the amplifier gain value from our recorded power measurement, to get the power measurement that we use in our implementation.

When the server gets a `request receivers` call, the server responds with the known location of the receiving pluto devices. This process is explained visually in the following diagram:
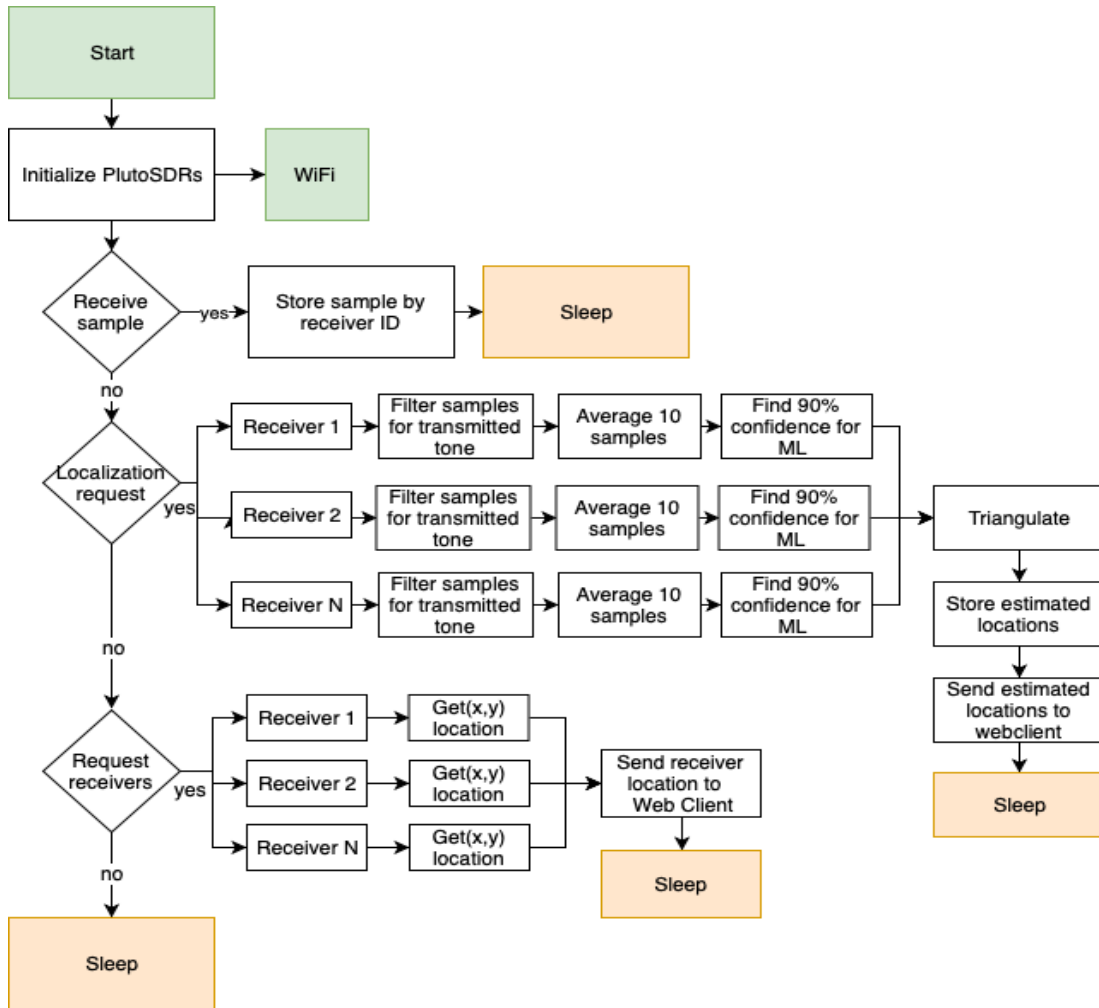
Figure 16: Localization Implementation Block Diagram

4.3.　Web Client

To visualize the localized signal transmissions, we are plotting them overtop a floor plan of the area being considered. As opposed to only plotting the exact point of our estimated location for each signal, we are plotting it as a heat map with varying colors to demonstrate the potential distance that the actual position could deviate from our estimated position. Additionally, shown in the plot are the PlutoSDR device locations. An example of this visualization with one signal localized and plotted is shown below.

As signals are localized within the main host software written in Python, the program send the position data to the server through POST requests. Parallel to this we have JavaScript software that fetches the signal position data from the server and creates the visualizations as described. The heat map is created utilizing an existing, open source, data visualization library 'heatmap.js'. This visualization is then posted to the web server for viewing. The visualization updates automatically every second to add any new data coming in to the existing plot.
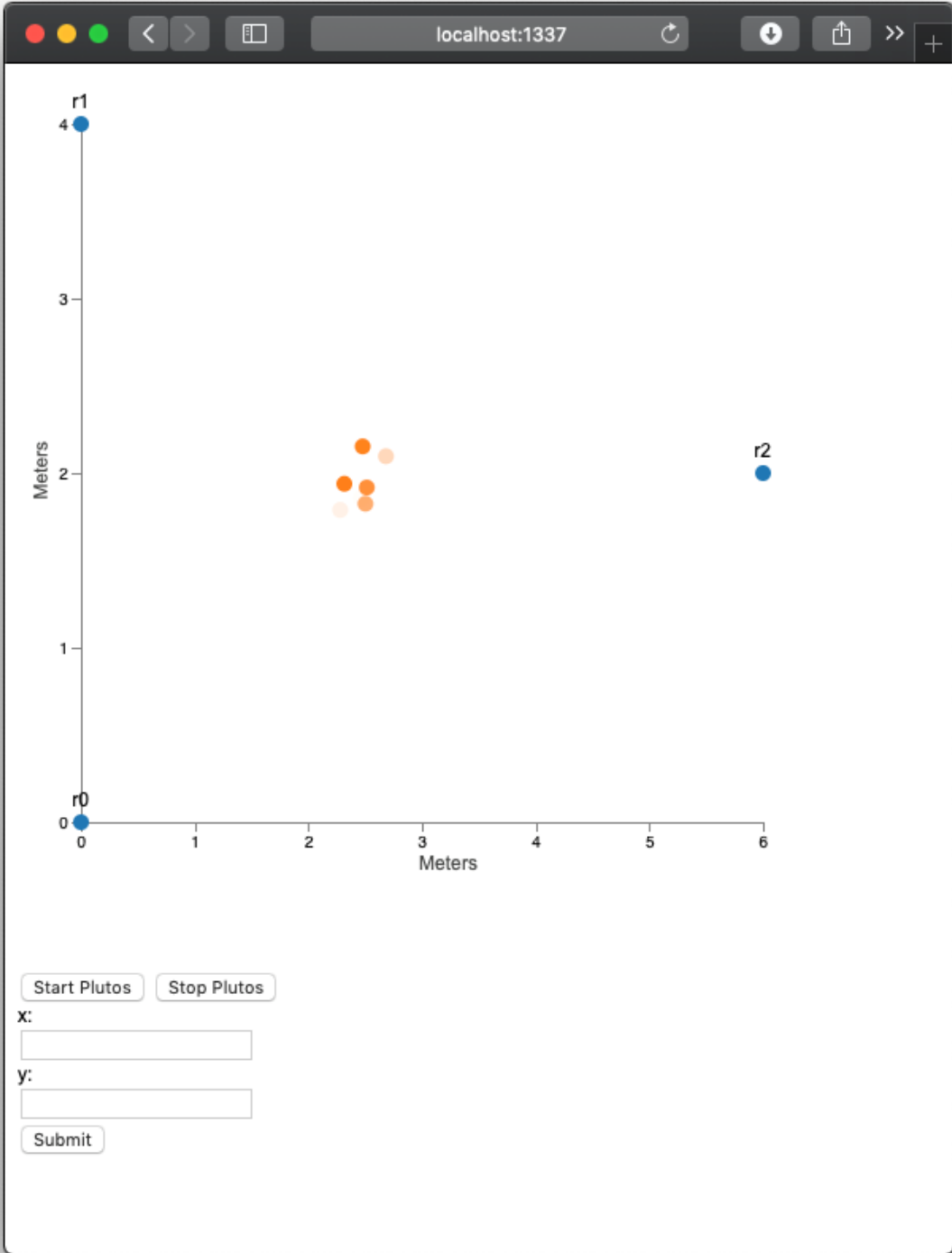
Figure 17: Signal Localization Implementation

This image shows the location of each PlutoSDR as each blue dot. The predicted location of the incoming signal is shown by the orange dots, which fade as time goes by.
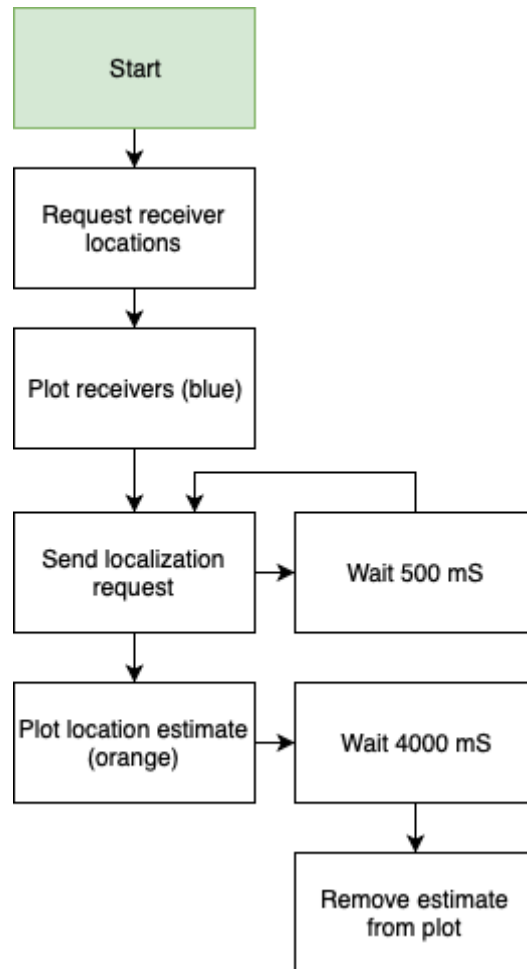

Figure 18: Localization Protocol

This diagram shows the protocols the host server follows. These commands are based off the data received in by the PlutoSDRs.

4.4. Transmitter

Below is a GNU radio flow graph that was used to turn one of the PlutoSDR's into a simple tone generator that we could localize on. Instructions on how to set up the Pluto with GNU radio can be found at the Analog Devices website. [52]

This flowchart has two main data streams, the first uses PlutoSDR source. This uses the transmitter on the PlutoSDR. The second data stream using the PlutoSDR as a sink, this uses the PlutoSDR transmitter. For this we passed in a 70 KHz sine wave into the PlutoSDR transmitter, this sine wave gets mixed up from base band with the 908 MHz local oscillator.
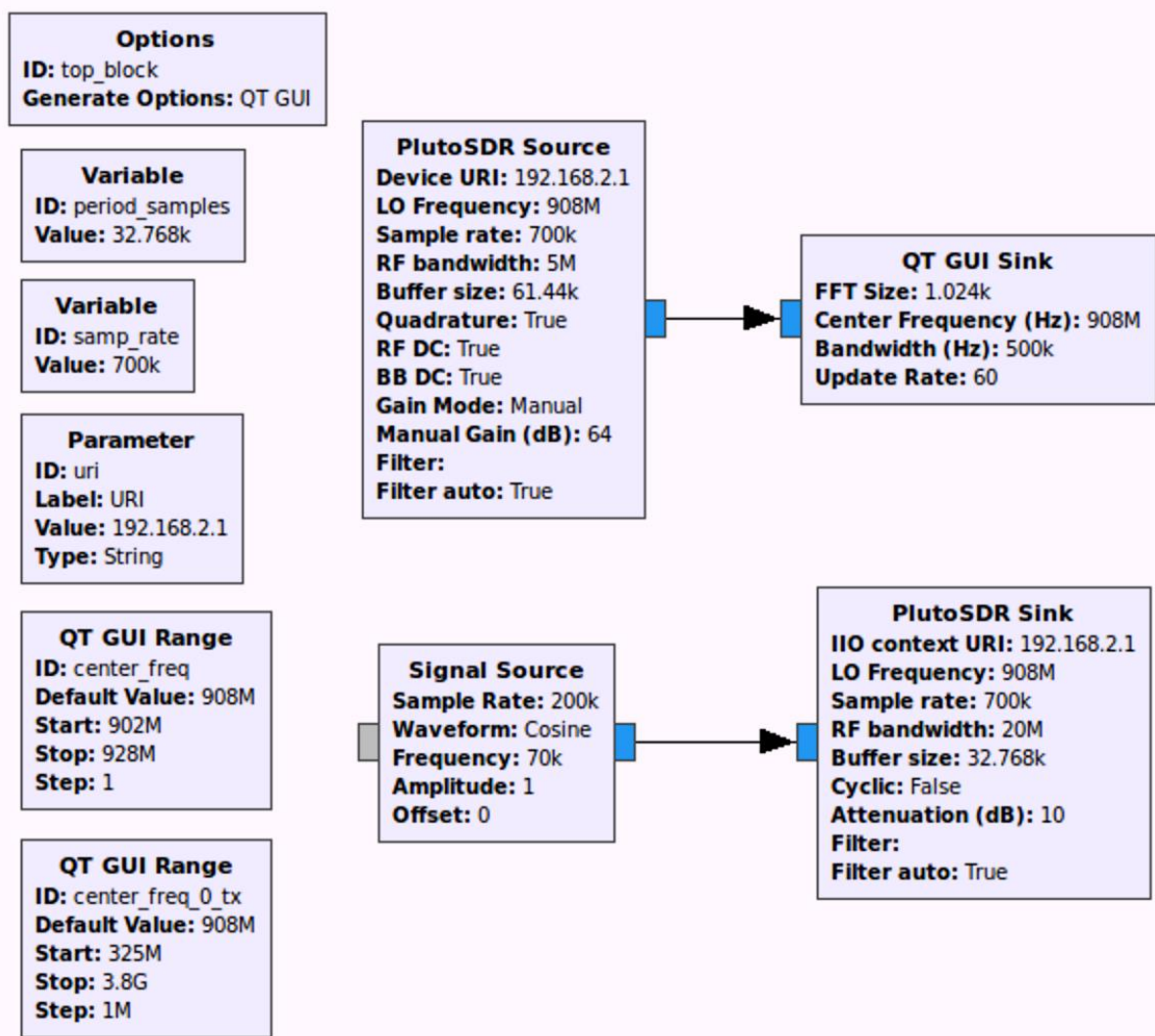

Figure 19: Full GNURadio Flow Graph

4.5.   Channel Model

In order to have a standardized test procedure, we made sure to document the steps required. We started by sending a sine wave from one PlutoSDR to be received by a secondary PlutoSDR. The transmitting PlutoSDR represented the fake IoT signal we would be using for our test purposes. The receiving PlutoSDR acted as the agent which will be a part of our full system including the WiFi dongle and power supply.

Once we had both PlutoSDRs appropriately set up to be transmitting and receiving respectively, we set a tape measure on the floor of a long hallway. We began by measuring the noise floor of the room, by collecting samples without the transmitter on.

We then turned the transmitter on and again walked down the hall to get an intuition of the maximum distance that we would be able to measure with our set up. We found that at 36 meters we could not discriminate our signal from the noise floor.
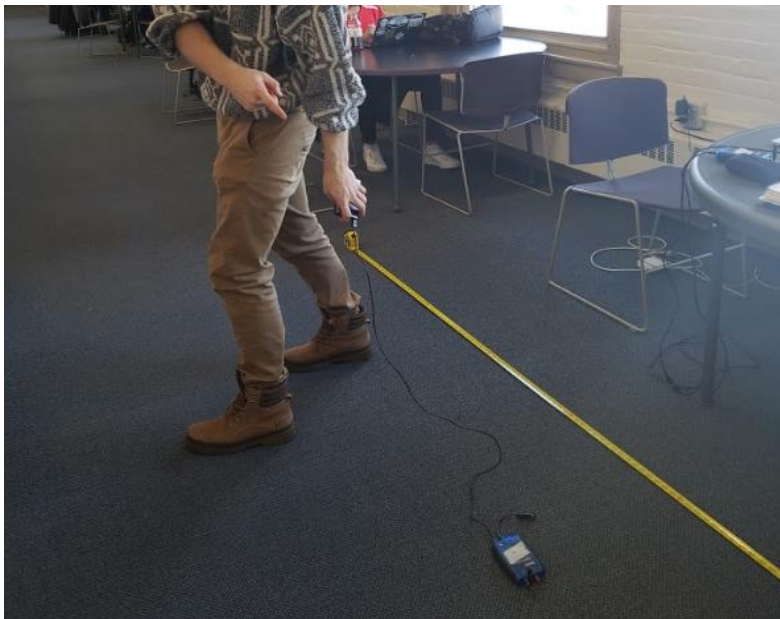
Figure 20: Preliminary Testing Set up

This is a picture showing the procedure of walking down the hallway with the PlutoSDR to collect measurements.

Then we took 500 samples at incremental measurements of signal power by 2-meter intervals starting at 1 meter to 35 meters in order to create a path loss model with parameters specific to the hall way we were attempting to measure.

The pathloss model we obtained was as follow:

$$P_r = P_0 - 10 * \alpha * log_{10}(r) + X(\sigma)$$

P_r : is the power received

P_0 : -87 dBm is the received signal strength at 1 meter

α : 2.61 is the exponential decay

σ : 6.43 is the standard deviation of shadow fading

For the equations used to create this channel model please refer to chapter 2.11. More information about the channel model can be found in the chapter 5.

5. Testing

In order to test if transmissions from these bands are being received, localized accurately, and stored, we will be testing with a PlutoSDR transmitting a single tone using GNU Radio. By controlling the signals being transmitted we will be able to accurately evaluate the device's capabilities.

The RSS of the transmitter was captured at 1 meter and 2 to 36 meters in 2-meter increments. The transmitter was placed at the end of the hall as the position of the receiver varied. Attenuation of the signal ranged from -87dB at 1 m to -128dB at 36 m. The RSS values were plotted against distance. Distance ranged from 1 meter to 36 meters. Figure 26 shows the plot of RSS (dBm) vs 10*log10(d)[dB].
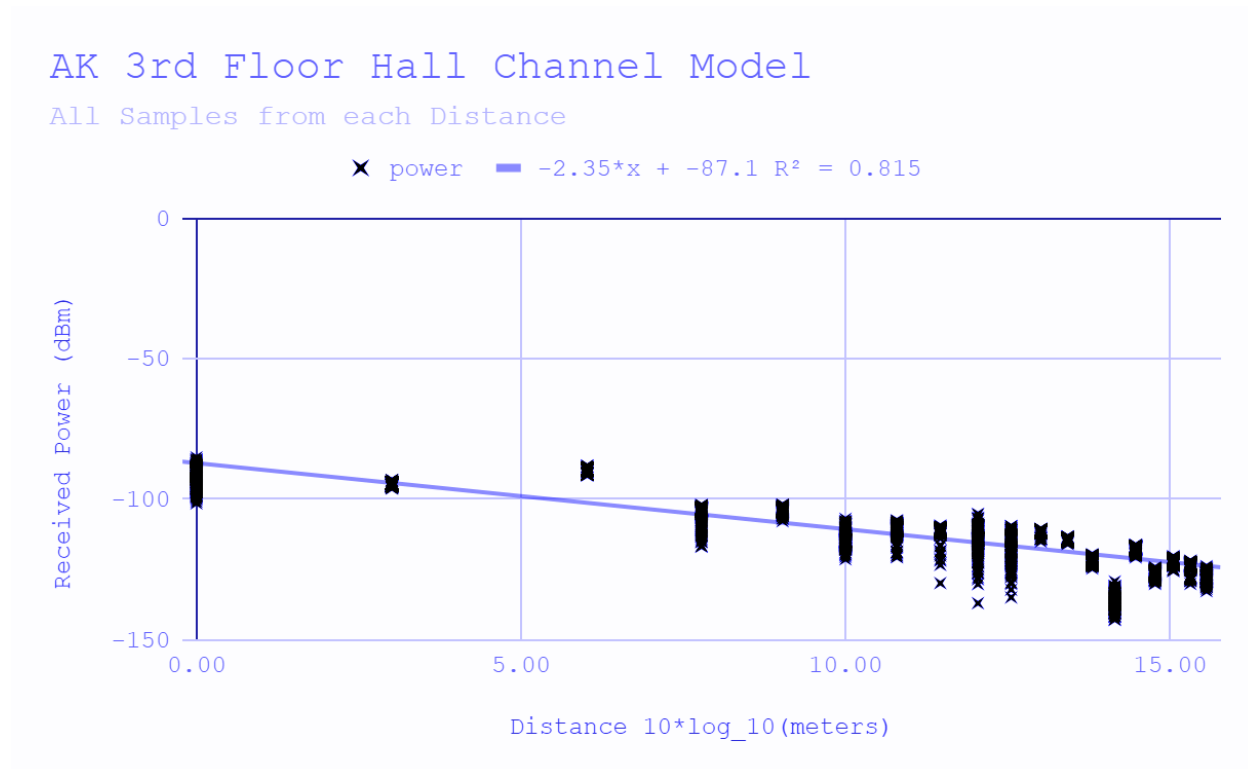

Figure 21: Channel Model

Figure 26 shows the linear regressive best fit line used to create a channel model where all samples are weighted evenly. You can get a sense of the variance of the data, from this plot and clearly see how the variance of the measurement starts out quite small, then increases, and then drops down again.

The following figure shows the model that we used to get our α value for our path loss model:
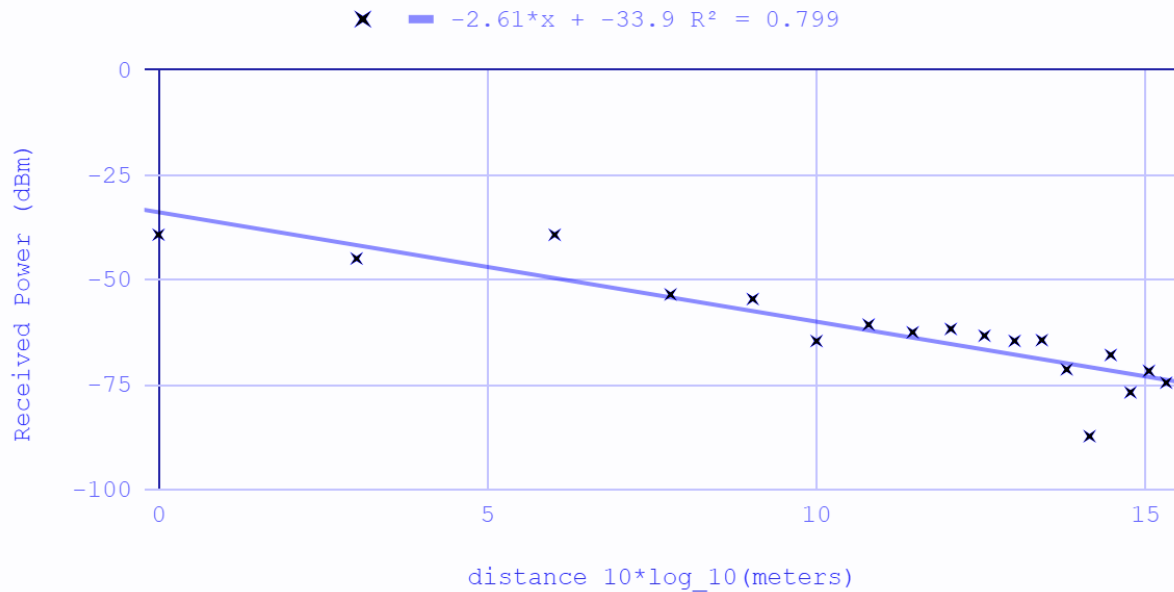
Figure 22: Average Samples

Figure 23 shows the averages of all the samples at each location to a single point, and preformed the same linear regressive best fit. The x axis is $10log_{10}(distance)$ in meters. With the averaged data we can see two large outliers. The first outlier is at 4 meters. It still cannot be explained what causes this, but at 4 meters it is recorded the same RSS as is done at 1 meter, but at 2 meters, and 3 meters the RSS continues to decrease as expected. This same phenomenon happened in every testing environment, so it is something intrinsic to the implementation, or the frequency band being examined, as the effects are independent of the space. The second outlier that we see is at 26 meters. Though it is not confirmed, it is believed this drop-in signal power to be from shadowing and interference in the first fresnel zone. At 26 meters there is a trash bin and a recycling bin that partially obstruct the hall leading to this large deviation from the model.

This deviation at 26 meters is more clearly seen in the following diagram of the distance measurement error for 1-dimentional ranging.
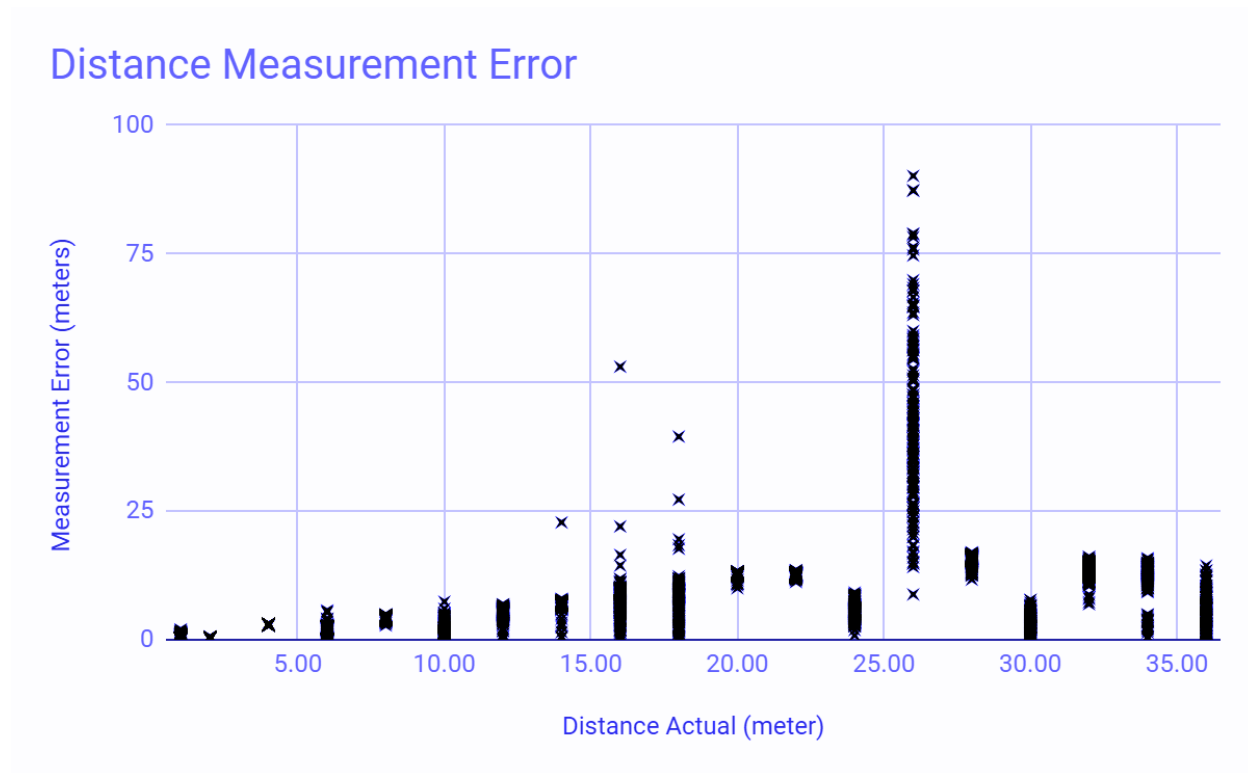

Figure 23: Distance Measurement Error

Figure 24 shows the distance measurement error. To create this graph, rearrange the pathloss model to solve for distance, and find the difference between the actual distance and distance the path loss model predicted. This graph further exaggerates the error seen at 26 meters do to the obstruction to direct line of sight between the transmitter and receiver. The data represented in this graph was also use calculate our value for shadow fading. Calculating the standard deviation of the distance measurement error to find the σ value for the channel model.

In order to really gauge the performance of our system, we must compare it to the Cramer Rao lower bound (CRLB). The CRLB gives a lower bound on the variance of the data. By

comparing the variance in DME to the CRLB we are able to measure how close our

implementation is to the theoretical best it can be. The CRLB was calculated using:

$$CRLB = \frac{\ln(10)^2}{100} \frac{\sigma^2}{\alpha^2} r^2$$

Equation 13

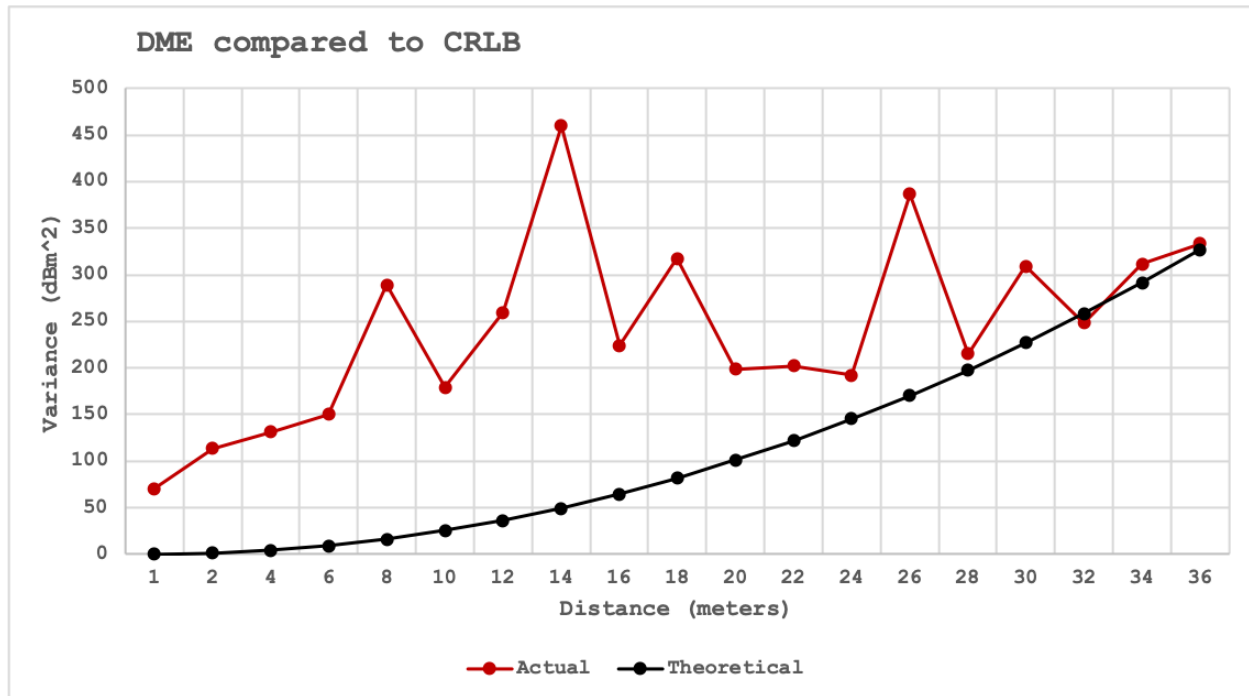The σ and α values are the parameters in channel model, and r is the ranging distance [12].

Figure 24: DME compared to CRLB

This comparison of our variance in DME to the theoretical CRLB, shows that our

solution could use some improvement. In a perfect world we would want our DME to vary

closely follow the CRLB. This lack of a tight curve might be attributed to the fact that left Auto

Gain Control within the PlutoSDR's on while sampling, and attempted to compensate for the

RSS by subtracting the current gain for each power measurement in decibels.

With a good understanding of the system performing 1-dimentional ranging estimations,

we can now move on and show how our system performs in 2 dimensions in figure 26.
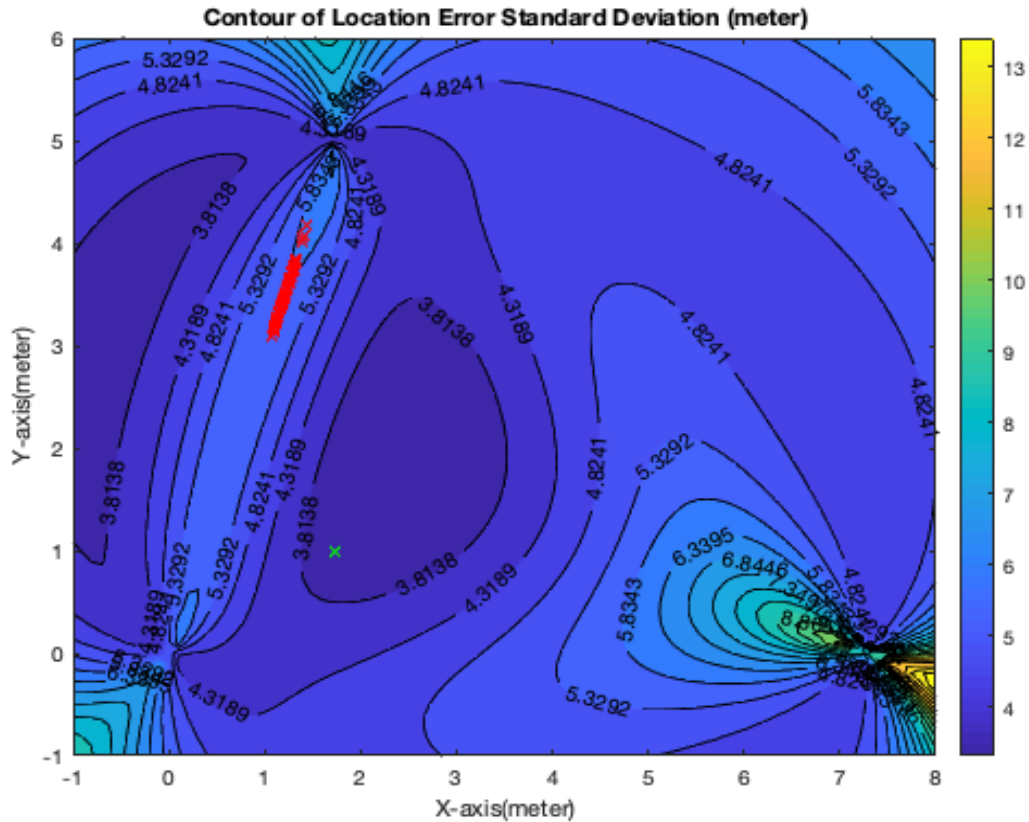
Figure 25: Location Estimations Plotted on top of Standard Deviation Contours

The standard deviation of position error is higher along the edges of the area. In the central areas, there are lower errors. In general, the positioning error is on the order of the ranging error. However, the distribution of errors in the area is different and fluctuate around the values of the ranging error from different APs. When in the central area, the results get equally accurate ranges from all RPs, which gives a better estimate of location shown in figure 26.

The green X is the actual location of the transmitter, and the red Xs are 244 estimates collected and calculated using the localization system described in this report under ideal conditions. Ideal condition meaning direct line of sight, and no one walking through the experiment. This test yields a maximum error of 2.5 meters.

5.1. Goals

Our goal set fourth for the success of this project are listed in table 3.

Table 3: Testing Objectives

Objective	Goal	Testing
Receive known transmissions	Receive a minimum of 80%	Evaluate ratio of known transmissions to received transmissions
Localize source of signals	Localize to 3m accuracy (1m stretch goal)	Evaluate known distance to transmission source to predicted localized distance
Store signal energies	Store 100% of received signals	Evaluate the ratio of known received transmissions to transmissions successfully stored

All of the goals were achieved. With our mock IoT transmitter we were able to receiver 100% of all of the transmissions, as a single consistent tone is very easy to detect with enough resolution in the Fourier Transform. Our maximum localization error for ours tests was 2.5 meters, which is within our 3-meter accuracy goal.

Our server was able to handle all the network traffic from all of the PlutoSDRs without any dropped packets, so we are able to successfully store 100% of all the received signals.

6. Future Work

We decided to use a few IoT devices to simulate the signals we would be receiving and localizing in order to better test our set up. We started by purchasing the z-stick and a smart light bulb. We were able to set these devices up properly, however we found due to the duration of the signal being transmitted (a few nanoseconds) we were worried about dropping samples during testing. This made implementation with devices using different transmitting protocols a good option for further work.

The z-stick works by creating a mesh network with all smart devices connected to it. We connected the z-stick to the smart light bulb following the given instructions with the z-stick, and connected the z-stick to an operating computer. Typically, the z-stick is used by consumers to create a cheap smart home network using a raspberry pi, but for our project sticking to computer for testing purposes is fine. The z-stick talks to the light bulb through the computer input and sends out 3 signals: send, data, and acknowledge. We were able to pick up these signals using the PlutoSDR and begin testing distances and signal strength.

This system will continue to be developed to eventually scan multiple bands and localize binary frequency shift keying (BFSK) modulated transmissions.

7. Bibliography

[1] C. Folk, D. C. Hurley, W. K. Kaplow and J. F. X. Payne, "THE SECURITY IMPLICATIONS OF THE INTERNET OF THINGS," AFCEA, 2015.

[2] "technavio," [Online]. Available: https://blog.technavio.com/blog/top-33-indoor-location-based-services-lbs-companies-in-the-us. [Accessed 22 April 2019].

[3] "signal hound," [Online]. Available: https://signalhound.com/products/usb-sa44b/. [Accessed 22 April 2019].

[4] "analog," [Online]. Available: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html. [Accessed 22 April 2019].

[5] "Hacker Warehouse," [Online]. Available: https://hackerwarehouse.com/product/hackrf-one-kit/. [Accessed 22 April 2019].

[6] "iot-analystics," [Online]. Available: https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/. [Accessed 22 April 2019].

[7] "statista," [Online]. Available: https://www.statista.com/statistics/688762/north-america-iot-market-by-application/. [Accessed 22 April 2019].

[8] B. Krishanamachari, D. Estrin and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," 2002. [Online]. Available: http://ieeexplore.ieee.org.ezproxy.wpi.edu/document/1030829/.

[9] "nrao," [Online]. Available: https://public.nrao.edu/telescopes/radio-frequency-interference/. [Accessed 22 April 2019].

[10] T. Brewster, "A Basic Z-Wave Hack Exposes Up To 100 Million Smart Home Devices," Forbes, 24 May 2018. [Online]. Available: https://www.forbes.com/sites/thomasbrewster/2018/05/24/z-wave-hack-threatens-to-expose-100-million-smart-homes/.

[11] C. Cheng, Lamarca and Krumm, "Accuracy characterization for metropolitan-scale Wi-Fi localization".

[12] K. Pahlavan, INDOOR GEOLOCATION SCIENCE AND TECHNOLOGY, Denmark: River Publishers, 2019.

[13] "ecnmag," [Online]. Available: https://www.ecnmag.com/blog/2017/06/understanding-rf-spectrum. [Accessed 22 April 2019].

[14] "wikimeadia," 2016. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/c/c7/United_States_Frequency_Allocations_Chart_2016_-_The_Radio_Spectrum.pdf. [Accessed 22 April 2019].

[15] "mtu," [Online]. Available: http://pages.mtu.edu/~scarn/teaching/GE4250/EM_theory_lecture.pdf. [Accessed 22 April 2019].

[16] [Online]. Available: https://books.google.com/books?hl=en&lr=&id=4LtmjGNwOPIC&oi=fnd&pg=PR7&dq=rf+reflection+refraction+diffraction+multipath&ots=06iwpx5S3k&sig=ktrkqAfqNQSavG13mwEEmLCbZls#v=onepage&q=rf%20reflection%20refraction%20diffraction%20multipath&f=false. [Accessed 22 April 2019].

[17] . R. Mishra, "3.3.6 Radius of Fresnel Zone," in *Fundamentals of Cellular Network Planning and Optimisation: 2G/2.5G/3G... Evolution to 4G*, United Kingdom, Wiley, 2004, p. 62.

[18] [Online]. Available: https://dot11ap.wordpress.com/cwna/radio-frequency-rf-technologies/line-of-sight-and-fresnel-zone-issues. [Accessed 22 April 2019].

[19] "wpi," [Online]. Available: https://wpi.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma9936740871804746&context=L&vid=01WPI_INST:Default&lang=en&search_scope=MyInst_and_CI&adaptor=Local%20Search%20Engine&tab=Everything&query=any,contains,frequency%20unlicensed%20bands&sortb. [Accessed 22 Apri; 2019].

[20] "ntia," [Online]. Available: https://www.ntia.doc.gov/page/2011/manual-regulations-and-procedures-federal-radio-frequency-management-redbook. [Accessed 22 April 2019].

[21] [Online]. Available: https://wpi.primo.exlibrisgroup.com/discovery/search?query=any,contains,radio%20frequency%20allocations&tab=Everything&search_scope=MyInst_and_CI&sortby=rank&vid=01WPI_INST:Default&lang=en&mode=basic. [Accessed 22 April 2019].

[22] "Jean Baptiste Joseph Fourier," [Online]. Available: https://www2.stetson.edu/~efriedma/periodictable/html/Fe.html.

[23] E. W. Weisstein, "Fourier Transform," MathWorld--A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/FourierTransform.html.

[24] "ams," [Online]. Available: http://www.ams.org/journals/mcom/1965-19-090/S0025-5718-1965-0178586-1/. [Accessed 22 April 2019].

[25] "purdue," [Online]. Available: https://engineering.purdue.edu/~ee538/DSP_Text_3rdEdition.pdf. [Accessed 22 April 2019].

[26] E. W. Weisstein, "Nyquist Frequency," MathWorld--A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/NyquistFrequency.html.

[27] [Online]. Available: https://machinaresearch.com/news/press-release-advancing-lte-migration-heralds-massive-change-in-global-m2m-modules-markets/. [Accessed 22 April 2019].

[28] "wired," [Online]. Available: https://www.wired.co.uk/article/internet-of-things-what-is-explained-iot. [Accessed 22 April 2019].

[29] "ieee," [Online]. Available: https://standards.ieee.org/initiatives/iot/stds.html. [Accessed 22 April 2019].

[30] "amca," [Online]. Available: https://www.acma.gov.au/theACMA/spectrum-at-434-mhz-for-low-powered-devices. [Accessed 22 April 2019].

[31] "efcr," [Online]. Available: https://www.ecfr.gov/cgi-bin/text-idx?SID=57e3d98742373709e9f8f17ed3759834&node=47:1.0.1.1.16.3.236.21&rgn=div8. [Accessed 22 April 2019].

[32] "eetimes," [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1273378. [Accessed 22 April 2019].

[33] "semtech," [Online]. Available: https://www.semtech.com/uploads/documents/fcc_part15_regulations_semtech.pdf. [Accessed 22 April 2019].

[34] "wired," [Online]. Available: https://www.wired.com/2010/09/wireless-explainer/. [Accessed 22 April 2019].

[35] "ti," [Online]. Available: http://www.ti.com/lit/an/swra048/swra048.pdf. [Accessed 22 April 2019].

[36] "fccid," [Online]. Available: https://fccid.io/frequency-explorer.php?lower=5000&upper=5000. [Accessed 22 April 2019].

[37] "fcc," [Online]. Available: https://www.fcc.gov/5G. [Accessed 22 April 2019].

[38] "ieee," [Online]. Available: https://ieeexplore-ieee-org.ezproxy.wpi.edu/document/7460875. [Accessed 22 April 2019].

[39] "ti," [Online]. Available: http://www.ti.com/lit/wp/swry013/swry013.pdf. [Accessed 22 April 2019].

[40] "electronic design," [Online]. Available: https://www.electronicdesign.com/what-s-difference-between/what-s-difference-between-ieee-802154-and-zigbee-wireless. [Accessed 22 April 2019].

[41] "ieee," [Online]. Available: https://standards.ieee.org/project/1900_1.html. [Accessed 22 April 2019].

[42] "analog," [Online]. Available: https://www.analog.com/media/en/training-seminars/design-handbooks/Software-Defined-Radio-for-Engineers-2018/SDR4Engineers.pdf. [Accessed 22 April 2019].

[43] "rtl-sdr," [Online]. Available: https://www.rtl-sdr.com/about-rtl-sdr/. [Accessed 22 April 2019].

[44] "amazon," [Online]. Available: https://www.amazon.com/gp/product/B00KCDF1QI/ref=as_li_tl?ie=UTF8&camp=1789&creative=390957&creativeASIN=B00KCDF1QI&linkCode=as2&tag=book0674-20&linkId=XHRIQAZC3JVLUJWM. [Accessed 22 April 2019].

[45] "analog," [Online]. Available: https://wiki.analog.com/university/tools/pluto. [Accessed 22 April 2019].

[46] "gnuradio," [Online]. Available: https://www.gnuradio.org/about/. [Accessed 22 April 2019].

[47] "Mathworks," [Online]. Available: https://www.mathworks.com/hardware-support/adalm-pluto-radio.html. [Accessed 22 April 2019].

[48] "mathworks," [Online]. Available: https://www.mathworks.com/hardware-support/rtl-sdr.html. [Accessed 22 April 2019].

[49] "gnuradio," [Online]. Available: https://www.gnuradio.org/grcon/grcon18/presentations/plutosdr/. [Accessed 22 April 2019].

[50] "analog," [Online]. Available: https://wiki.analog.com/software/linux/docs/iio/iio. [Accessed 22 April 2019].

[51] "fftw," [Online]. Available: http://www.fftw.org. [Accessed 22 April 2019].

[52] "analog," [Online]. Available: https://wiki.analog.com/resources/tools-software/linux-software/gnuradio. [Accessed 22 April 2019].

[53] "pentestpartners," [Online]. Available: https://www.pentestpartners.com/security-blog/hijacking-philips-hue/. [Accessed 22 April 2019].

[54] "arxiv," [Online]. Available: https://arxiv.org/pdf/1709.01015.pdf. [Accessed 22 April 2019].

[55] S. M. I. A. G. S. M. I. K. K. L. F. I. Faheem Zafari, "A Survey of Indoor Localization Systems and Technologies," arxiv.

Appendix 1.1 Building the Firmware

INSTALL:

1) PREREQUISITES

   a) Make sure you have at least 50 GB of storage

   b) 2 CPU cores

   c) 4 GB ram

2) UBUNTU 16.04

   https://drive.google.com/file/d/13BN4VjMsQL-VFKCNr83i3VSXkjBfUhJ9/view?usp=sharing

3) XILINX WEB INSTALL 17.02

   a) Download web installer .bin from website

   b) In terminal chmod +x *.bin

   c) ./*.bin to run the installer

   d) install should be around 32GB, when installing make sure that all boxes are checked, especially the Zynq 7000's

   e) MAKE SURE THE SDK IS ALSO INSTALLED

   f) Add source /opt/Xilinx/Vivado/2017.2/settings64.sh to ~/.bashrc

4) Install all necessary packages

   a) Sudo dpkg --add-architecture i386

   b) Sudo apt-get update

   c) Sudo apt-get upgrade

   d) sudo apt-get install libc6:i386 libstdc++6:i386

   e) sudo apt-get install git build-essential ccache device-tree-compiler dfu-util fakeroot help2man libncurses5-dev libssl-dev mtools rsync u-boot-tools

https://wiki.analog.com/university/tools/PlutoSDR/building_the_image

Appendix 1.2 Expanding Pluto Frequency Range

https://wiki.analog.com/university/tools/PlutoSDR/users/customizing

https://unix.stackexchange.com/questions/144029/command-to-determine-ports-of-a-device-like-dev-ttyusb0

1 sudo apt install screen

2 Bash script to look for USB serial devices

```
#!/bin/bash

for sysdevpath in $(find /sys/bus/usb/devices/usb*/ -name dev); do
    (
        syspath="${sysdevpath%/dev}"
        devname="$(udevadm info -q name -p $syspath)"
        [[ "$devname" == "bus/"* ]] && continue
        eval "$(udevadm info -q property --export -p $syspath)"
        [[ -z "$ID_SERIAL" ]] && continue
        echo "/dev/$devname - $ID_SERIAL"
    )
Done
```

Sudo screen /dev/tty(the port you found from the previous script)

username : root

Password: analog

```
# fw_printenv attr_name
## Error: "attr_name" not defined
# fw_printenv attr_val
## Error: "attr_val" not defined
#
# fw_setenv attr_name compatible
# fw_setenv attr_val ad9364
# reboot
```

Appendix 1.3 Installing GnuRadio

Link used:

https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux

```
 sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.13-0 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg  libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk2.8 git-core libqt4-dev python-numpy ccache python-opengl libgsl0-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq1 libzmq-dev python-requests python-sphinx libcomedi-dev
```

```
cd $HOME
  mkdir workarea-gnuradio
  cd workarea-gnuradio
```

Next, clone the repository.
```
  git clone --recursive https://github.com/gnuradio/gnuradio
```

Next, go into the repository and check out the desired GNU Radio version.
```
  cd gnuradio
  git checkout v3.7.12.4 // or whatever the newest version is in the git repo
```

```
git submodule update --init --recursive
```

Next, create a build folder within the repository.
```
mkdir build
cd build
```

Next, invoke CMake to create the Makefiles.
```
cmake ../
```

Next, run Make to build GNU Radio.
```
make
```

Next, you can optionally run some basic tests to verify that the build process completed properly.
```
make test
```

Next, install GNU Radio, using the default install prefix, which will install GNU Radio under the /usr/local/lib folder. You need to run this as root due to the permissions on that folder.
```
sudo make install
```

Finally, update the system's shared library cache.
```
sudo ldconfig
```

At this point, GNU Radio should be installed and ready to use. You can quickly test this, with no USRP device attached, by running the following quick tests.
```
gnuradio-config-info --version
gnuradio-config-info --prefix
gnuradio-config-info --enabled-components
```

There is a simple flowgraph that you can run that does not require any USRP hardware. It's called the dialtone test, and it produces a PSTN dial tone on the computer's speakers. Running it verifies that all the libraries can be found, and that the GNU Radio run-time is working.
```
python $HOME/workarea-gnuradio/gnuradio/gr-audio/examples/python/dial_tone.py
```

You can try launching the GNU Radio Companion (GRC) tool, a visual tool for building and running GNU Radio flowgraphs.
```
gnuradio-companion
```

If "gnuradio-companion" does not start and complains about the PYTHONPATH environment variable, then you may have to set this in your $HOME/.bashrc file, as shown below.
```
export PYTHONPATH=/usr/local/lib/python2.7/dist-packages
```

On Fedora 21/22/23/24, the PYTHONPATH environment variable will need to be set to:
```
export PYTHONPATH=/usr/lib/python2.7/site-packages:/usr/local/lib64/python2.7/site-packages/
```

Drivers for the PlutoSDR

https://wiki.analog.com/resources/tools-software/linux-software/gnuradio

Install GNU Radio and other dependencies
```
apt-get -y installlibxml2 libxml2-dev bison flex cmake git libaio-dev libboost-all-dev swig
```
Download and build libiio
```
git clone https://github.com/analogdevicesinc/libiio.git
cd libiio
cmake .
make
sudo make install
cd ..
```
Download and build libad9361-iio
```
git clone https://github.com/analogdevicesinc/libad9361-iio.git
cd libad9361-iio
cmake .
make
sudo make install
cd ..
```
Download and build gr-iio
```
git clone https://github.com/analogdevicesinc/gr-iio.git
cd gr-iio
cmake .
make
sudo make install
cd ..
sudo ldconfig
```
GNURadio will recommend you include
```
/usr/local/lib${type}/python${PYVER}/site-packages/gnuradio
```
or
```
/usr/local/lib${type}/python${PYVER}/dist-packages/gnuradio
```
in your PYTHONPATH during installation. If this is not the case you will need to modify the cmake command for the gr-iio configuration above with:
```
cmake -DCMAKE_INSTALL_PREFIX=/usr .
```

Getting example running:

Finally, launch GNU Radio Companion from Ubuntu with command

gnuradio-companion

IIO Examples

Several sample flow graphs that use the FMCOMMS-2/3/4 IIO blocks are provided in our GNU Radio repository. They can be found in the "iio-example" folder.

Create this flowchart inside gnuradio-companion
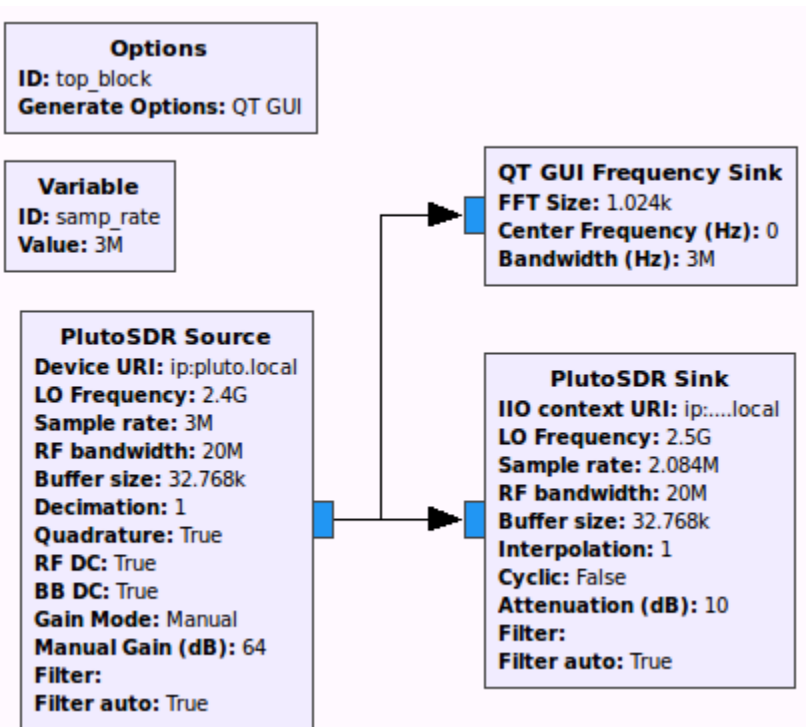
Appendix 1.4 Installing Matlab

This example assumes a working version of Matlab 2017 or later is available a person's machine.

Also required is the Communications toolbox which can be found here:

https://www.mathworks.com/help/comm/

Communications Toolbox™ Support Package for Analog Devices® ADALM-Pluto Radio which can be found here:

https://www.mathworks.com/help/supportpkg/PlutoSDRradio/ug/install-support-package-for-PlutoSDR-radio.html

OR

https://www.mathworks.com/help/supportpkg/PlutoSDRradio/installation-and-setup.html

A full list of documented examples and support for the Pluto using Matlab can be found here:

https://www.mathworks.com/help/supportpkg/PlutoSDRradio/index.html

Follow the tutorial as shown in the installer to connect your PlutoSDR and test the transmit and receive functions. Once it is properly installed you can follow several matlab tutorials outlining different SDR uses and test programs, including turning the Pluto into a spectrum analyzer.

Spectrem analyzer example:

https://www.mathworks.com/help/supportpkg/PlutoSDRradio/examples/spectral-analysis-with-adalm-PlutoSDR-radio.html
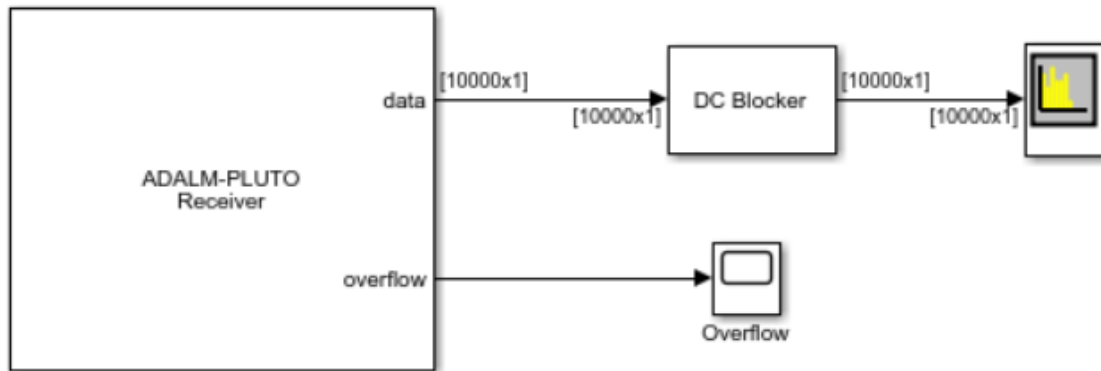
This example is done through simulink which provided a very user friendly compatibility.

There are also other "fun" examples such as plane tracking found here:

https://www.mathworks.com/examples/communications/mw/plutoradio_product-plutoradioADSBExample-airplane-tracking-using-ads-b-signals-and-adalm-pluto-radio

Many other functional matlab tutorials exist such as frequency offset and and more specific transmitting/receiving which can be found here:

https://www.mathworks.com/examples/search?q=PlutoSDR

We cross checked the results of the Matlab spectrum analyzer and the gnu radio spectrum analyzer and found very similar results.

Appendix 2.1 Device Specifications

PlutoSDR Specifications

- RF Performance: [ADI AD9363, RF Agile Transceiver](#)

    - Tuning range: 325 MHz - 3.8 GHz

    - Tunable channel bandwidth: 200 kHz - 20 MHz

    - Integrated 12-bit DACs (Tx) and ADCs (Rx)

    - Variable output data rates: 61.44 MSPS - 65.1 kSPS

    - Modulation Accuracy (EVM): -34 dB (2%)

    - No RF Shielding

- Programmable Chip: [Xilinx Zynq](#) XC7Z010-1CLG225C

    - FPGA

        - Logic Cells: 28k

        - Block RAM: 2.1Mb

        - DSP Slices 80

    - ARM Processing System

        - Single-core ARM® Cortex™-A9 MPCore™

        - 667 MHz

    - USB 2.0 (included in the Zynq)

        - Streams up to 4MSPS with no dropped samples

- Memory: Micron DDR3L MT41K256M16, Micron QSPI Flash MT25QU256ABA

    - DDR3L

        - 1066 Mbps (16-bit interface)

        - 512 Mb

- QSPI Flash

    - 32 Mb

    - Quad I/O provides throughput up to 54 Mbps

- Power

    - 5V DC input via USB connection

- Physical

    - Dimensions: 117 mm $\times$ 79 mm $\times$ 24 mm

    - Weight: 114 g

    - Temperature: 10°C to 40°C

Wifi Dongle Specfications

- Hardware

    - USB 2.0 Interface

    - Green status LED

    - Dimensions: 18.6mm x 15mm x 7.1mm

    - Weight: 2.1g

    - Temperature: 0°C to 40°C

- Wireless Features

    - Standards: IEEE 802.11b, IEEE 802.11g, IEEE 802.11n

    - Frequency: 2.400-2.4835GHz

    - Signal Rate: Up to 150Mbps (dynamic)

        - Automatically adjusts to lower speeds due to distance or other operating limitations

- ○ Transmit Power: <20dBm