Spring 2019

Radio Frequency Toolbox for Drone Detection and Classification

Abdulkabir Bello
Old Dominion University, temidee4real07@yahoo.com

RADIO FREQUENCY TOOLBOX FOR DRONE DETECTION AND CLASSIFICATION

by

Bello Abdulkabir
B.S Computer Engineering
M.S. May 2019, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
May 2019

Approved by:

Dr. Sachin Shetty (Director)

Dr. Dimitrie Popescu (Member)

Dr. Chunsheng Xin (Member)

ABSTRACT

RADIO FREQUENCY TOOLBOX FOR DRONE DETECTION AND CLASSIFICATION

Bello Abdulkabir
Old Dominion University, 2019
Director: Dr. Sachin Shetty

The continuous development of inexpensive embedded sensors has led to rapid proliferation of new civilian use of unmanned aerial vehicle (UAVs) or drones. It is now easier for civilians to own drones as the cost falls. As we all know drones have a variety of important applications and can also be used for negative effects too. These drones can pose a threat to the security of the population either civilian, organization or industry. There is a need for Radio Frequency Signal Classification (RF-Class) toolbox which can monitor, detect, and classify RF signals from drone communication system. The ability to accurately classify over-the-air radio signals will provide insights into spectrum utilization, device fingerprinting and protocol identification. These insights can help the Warfighter to constantly be informed about adversaries transmitters capabilities without their knowledge. The advantage of the drone detection and classification toolbox is extracting information about transmitters and providing receivers information about transmitted signals. The classification of RF signals will be done based on the modulation scheme, in this case orthogonal frequency division multiplexing (OFDM). The signal energy and features from the signals leveraging its orthogonal frequency division multiplexing(OFDM) parameter information will be used to classify the signal. This classification will be done using the capabilities of machine learning to train and test the information collected. The content of this thesis discuss how drone detection and classification can be achieved using software defined radio. GNU radio and other hardware components will be used to implement a simulation of the module.

Dedicated to my family,friends and my supervisor!

ACKNOWLEDGEMENTS

Firstly, I would like to thank my thesis advisor Dr. Sachin Shetty of the MSVE/ECE department at Old Dominion University for giving me the opportunity to work and be part of this interesting project. He was always available to help me with relevant and necessary information and guided me all through the research progress, which formed the bases of this thesis.

I would also like to acknowledge Biswajit Biswal of the CS department at South Carolina State University for his contributions and support to the successful implementation of the project, and I am appreciative of all the time spent on discussion and working on the project.

Finally, I must express my very profound gratitude to my parents Mr. and Mrs. Bello, my friends and fellow students for providing me with invaluable support and continuous encouragement throughout my academic years of study and also through the process of carrying out the research and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 UNMANNED AERIAL VEHICLE

Unmanned aerial vehicle (UAS). The first question a person not familiar with the term will ask is, What is a UAS? In literature and in general, the entire operating equipment, which is comprised of an aircraft, the control station, and the wireless link between the aircraft and its control system is described as the UAS. Unmanned aerial vehicle (UAV) is the designation that is mostly used to define flying objects for various applications ranging from recreational, civilian, and professional use. The terms UAS and UAV can be interchangeable in their usages in this thesis but can be regarded to mean the same thing. Drone seems like the more used term among hobbyists and is an unmanned or autonomous aircraft which is commonly used in a military context while it is also used to designate any of the numerous available classifications or types of aerial unmanned vehicle in the common language. It can be observed that in recent years there has been a lot of interest and research on the unmanned aerial vehicle (UAV) due its ubiquities today; the continuous development of inexpensive embedded sensors has led to rapid proliferation of new civilian use of unmanned aerial vehicles (UAVs) or drones. [1] highlighted some various applications of drones in the society with its application ranging from educational and commercial use. It is now easier for civilians to own drones as the cost falls. As we all know drones have a variety of important applications and can also be used for negative effects too. Detection of drone signals presents an interesting challenge to researchers and hobbyists in general. Some of the challenges are: 1. Drones can operate and appear in all directions so therefore a detection and monitoring equipment should also be able to monitor multiple directions at the same time. 2. It is difficult to effectively distinguish the drone appearance from that of other flying objects such as kites, birds etc. most especially in a case where the drone is far from the detection module

3. As an electronic device which still relies on power to work it presents a limita-

tion to its battery and communication and as a result consumer-grade drown essentially operates a very low altitude. Objects and environment present an obstruction to the drone usage often.

3. As an electronic device which still relies on power to work it presents a limitation to its battery and communication and as a result consumer-grade drown essentially operates a very low altitude. objects and environment present an obstruction to the drone usage often.

1.2 DRONE DETECTION METHODS

In literature there exist different methods of drone detection, some notable detection methods are audio, video, thermal, radar and radio frequency detection. [2] discusses some of the challenges faced in drone detection and presented an approach to detection using an audio assisted array. [3] implemented a passive radar technique approach to detection of drone signal. [1] discussed some principles of drone detection using the radio frequency approach. In [4] a thermal approach to drone detection was investigated.

1.2.1 AUDIO DETECTION

The approach to audio detection of drone involves the use of an array of microphones in multiple directions to capture the ambient sound from the drones. Most of the microphones used for detection can pick up sound from 25ft to 30ft. After the sound waves are recorded, they are then processed and filtered, and the target frequency is analyzed. It is known that many classes of drones are equipped with brushless direct current motors which generate a hissing high frequency sound around 40 KHz which are unique for most of the drones. Using digital signal processing, the specific frequency of interest can be identified, and the presence of a drone nearby can be determined. The pros of this method are that it works in quieter environments and will perform badly in urban areas or noisy environments where the signal to noise ratio is very low.

1.2.2 VIDEO DETECTION

For video detection of drone signals, cameras can see out to about 350ft with a usable resolution, which is very economical. The challenge to these methods is that

they have great difficulty in accurately distinguishing birds from drones. Unfortunately, as we have discovered, this notion fails in a place where birds glide.

1.2.3 THERMAL DETECTION

Due to the temperature characteristics, objects warmer than absolute zero emit infra-red radiation which can be detected by thermal imaging systems.this can be extended to drone detection.This method of detection works much better on drones that have a propulsion engine, mostly on fixed wing drones and the performance can be good for up to a distance of 350ft. Propulsion engines like the turbo-fan or the turbo-jet engines generate hot gases from the exhaust which makes it easy to detect. This method performs poorly, however, when the large percentage of drones body are made of large amount of plastic or radiate less heat, like most plastic quad-copters with electric motors. In such cases this mechanism is more likely to consider a birds or other flying object with more heat than a drone. As a result of such unreliability in detection, this mechanism can be used as an addition with other detection mechanisms rather than standalone.

1.2.4 RADAR DETECTION

RADAR is very useful in detecting large aircrafts but not as much when the size is small like a quad-copter. Radars find it difficult to pick up these small, plastic, electric-powered drones commonly used in the society right now because they were not primarily designed for this purpose. [3] implemented a passive radar technique approach to detection of drone signal.

1.2.5 RADIO FREQUENCY DETECTION

Detection of drone signals using its radio frequency (RF) signature is one of the effective ways for long-range detection of drones. The drones communication protocol is designed in such a way that the aircraft communicates with the ground control station (remote controller) and transmits video image over . Due to the characteristics of a wireless signal, such radio frequency signals can be detected from a long distance. It will be very difficult to design a drone that can completely escape RF detection. Although some UAVs with fixed wing propulsion engines can fly at very high altitude and can escape detection using this technique, mostly the

detection success rate is highly dependent on the power of the transmitter as well as the sensitivity of the receiver.

1.3 HOW DRONE WORKS

The drone is made of two fundamental parts which are the remote control and the aircraft. Both communicate with one another using a radio frequency communication link. The figure below presents the architecture and architectural design and components of most drones.

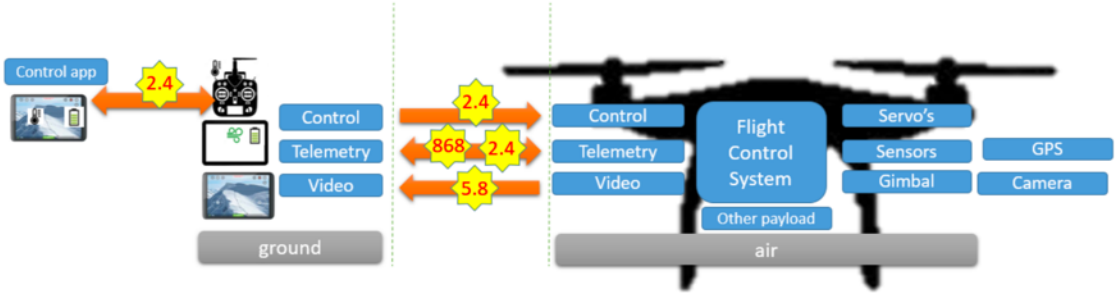


Fig. 1: The Components of a drone.

For our RF classification we will be focusing on the communication between the ground (remote controller) and air (aircraft). The remote controller can directly control the drone by means of control, data and video transmission communicate and send information between each other. The majority of controllers use the 2.4 GHz spectrum with a proprietary frequency hopping spread spectrum (FHSS) modulation [6]. FHSS is used to maximize robustness and controlling distance. Both of these signal modulation types are used in military applications as well as some available electronic devices such as cell phones. Another important characteristic is they are very resilient to interference due to the fact that the transmitter and receiver are paired together. In the military this technology can have an important application as it can be used for low probability of detection (LPD) to keep the enemy from knowing that the signal is there to begin with. This characteristic however makes a drone more difficult to detect. FHSS controllers can be effective against jamming as well due to the fact that they are not just transmitting informations on a static frequency but the whole width of the spectrum the drone is hopping in. The remote controller and the aircraft transmit and receive control, video and other kind of signal

between them.

Table 1 shows the list of information associated with control signals of some available drones used by hobbyist, student or researchers

Brand	**Frequency**	**Modulation**	**Technology**
DJI Phantom	2.4Ghz/5.8Ghz	FHSS/DSSS	FASST/Lightbridge
Futaba	2.4Ghz	FHSS/DSSS	FASST
Spektrum	2.4Ghz	FHSS/DSSS	DSMX
JR	2.4Ghz	FHSS/DSSS	DMSS
Hitec	2.4Ghz	FHSS/DSSS	AFHSS
Graupner	2.4Ghz	FHSS/DSSS	HOTT
Yuneec	2.4Ghz	DSSS	ZigBee
Parrot AR2		OFDM	Wi-Fi

TABLE I: Popular controls used by drones, frequencies they operate on and the used modulation technique and technology.

Table 2 shows some information associated with the communication signal for the video transmission between the remote controller and the aircraft,

Brand	**Frequency**	**Modulation**	**Technology**
DJI Phantom	2.4Ghz	OFDM	Lightbridge/Wi-Fi
Immersion	2.4Ghz	FM	
Yuneec	5.8Ghz	OFDM	Wi-Fi
Connex	5.8Ghz	OFDM	
Immersion	5.8Ghz	FM	
Boscam	5.8Ghz	FM	

TABLE II: Popular video transmission used by drones, frequencies they operate on and the used modulation technique and technology .

1.3.1 OVERVIEW OF FREQUENCY AND MODULATION TECHNIQUE USED BY DRONES

As mentioned earlier drones generally work the same as other radio-controlled devices where the controller is the transmitter and the aircraft has a receiver capable of understanding what the remote controller is sending to it. [7] Normally, the frequency

band allocated by the FCC for RC drones is either 27MHz or 49MHz.However, the control frequency for the newer drones are in the 2.4GHz or 5GHz range from the manufacturer. It is important to note that transmitters and receivers are readily available in different frequency ranges so a custom-built drone for instance could be operating on a different frequency than ones listed below. Some other license exempt spectrum bands that are used by newer drones are the 433 MHz, 868 MHz and the 5.8 GHz. The 2.4 GHz band in most cases are used for control, while the 5.8 is mostly used for video. The 433MHz and 868 MHz bands are mostly used for telemetry, besides the 2.4 GHz. Table 3 shows an overview of the different frequencies and modulation techniques used by drones

Frequency	**433Mhz**	**868Mhz**	**2.4Ghz**	**5Ghz**
Control	No	No	DSSS+FHSS/DSSS/OFDM	OFDM
Video	No	No	OFDM	OFDM, FM
Telemetry	Divers	OFDM	OFDM,DSSS	OFDM

TABLE III: Popular video transmission used by drones, frequencies they operate on and the used modulation technique and technology .

1.3.2 PROBLEM STATEMENT

With the available information on UAV, the major goals of the thesis are to develop a drone detector system based on RF signal analysis and classify the drones using a machine learning algorithm. The system will be implemented using a software defined radio. GNU radio companion as the software component part and HackRF will be used as the hardware component part. Some questions to answer are:

1. Is it possible to detect drones using RF signal analysis?
2. Which characteristics features of signal can be extracted?
3. What are the best features to extract and use for detection and classification?
4. How accurate is the RF detection module?
5. Can we accurate distinguish between drone from the same or different class

1.3.3 OUR APPROACH

Our approach is based on the concept discussed in [5]. We intend to leverage the

knowledge of RF detection to achieve the detection, feature extraction, and classification of drone signals using machine learning. We intend to extract the OFDM parameters of the drone signal in addition to the energy and other statistics associated with the received signal. In other to detect the signal we will be implementing energy detection of a wireless signal. There are other available detection techniques available in literature and in practice, but we have chosen to use energy detection because of its simplicity, as it does not employ a lot of computational complexity. After the features are extracted from the incoming signal of interest, we will carry out additional processing. This processing involves the labeling and training of the collected features. This is necessary because we will be using the supervised learning method of machine learning to classify the drone signals. GNU radio blocks will be used to implement the detection, extraction, classification and testing. The data training and processing will be performed using python. After the training is done in Python, a testing file will be created and loaded into the GNU radio block for real time testing and analysis.

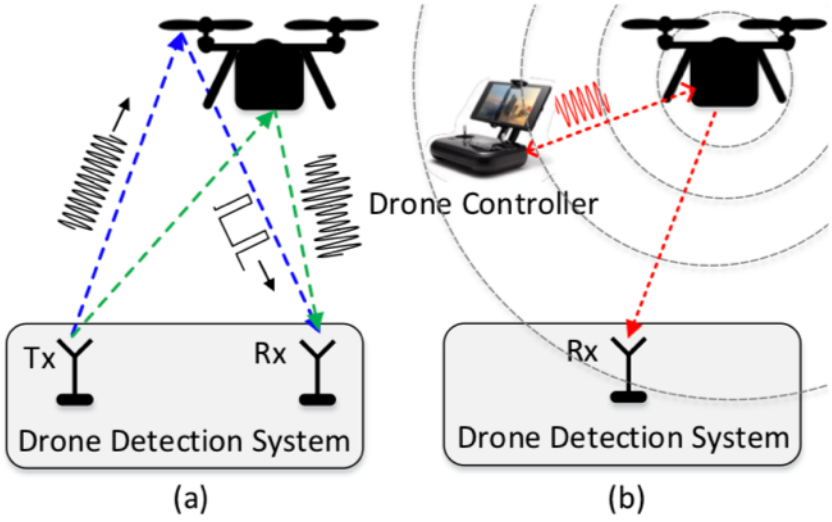


Fig. 2: The overview of drone detection system: (a) active and (b) passive approaches.

.

CHAPTER 2

BACKGROUND

2.1 OVERVIEW OF SOFTWARE DEFINED RADIO

There has been an exponential growth in the means by which people to communicate, a lot of options exists such voice communication, data communication, video communication, command and control communications, etc. As such having an easily modifiable and cost- effective radio device has become very important and critical business. Software defined radio has various definitions in literature, [8] simply defined it as " a radio in which some or all of the physical layer functions are software defined"

2.2 BACKGROUND OF SDR

Joe Mitola of Mitre is generally credited with being the father of the software defined radio [9]. Although the history of the SDR can be traced to begin in the mid 1980s. In literature one of the first major developments for SDR was the SpeakEasy, which basically is a transceiver platform designed by Hazeltine and Motorola, based on SDR technology for Rome Griffiss Air Force Base. The SpeakEasy was primarily designed to provide tactical military communications from 2 MHz to 2 GHz and also to provide interoperability between the different air interface standards of the different branches of the armed forces. The SpeakEasy technology utilized many of the wireless techniques and concepts to provide multi-band, multi-modes of operations. Many people have made valuable contributions to the development of SDR from then until what it is today. This history is beyond the scope of my thesis

2.3 BENEFITS OF SOFTWARE DEFINED RADIO

Some of the benefits of software defined radio includes but not limited to the listed point:

1. Software defined radio provides interoperability as it supports multiple standard through its multiband and multimode radio capabilities

2. For users it is cheap to purchase there by reducing ownership cost, requires less maintenance and easy to deploy.

3. Its sustainable as a result of increased utilization through generic hardware platforms

4. SDR are adaptable and makes it easier and faster to migrate towards new standards and technology through programmability and reconfiguration

2.4 ARCHITECTURE OF SOFTWARE DEFINED RADIO

The architecture of SDRs can be broken down into three main sections: the RF section, the IF section and the Baseband section. The RF signals are received via antennas and converted to IF signals. The IF signals are then converted to baseband and the digital signal processing carried out in software [9]. The conversion of RF to IF signals makes it possible to accommodates the hardware and software speed limit of Commercial Off-The-Shelf (COTS) components. Digitalization by the ADC also prepares the signal for further digital signal processing.

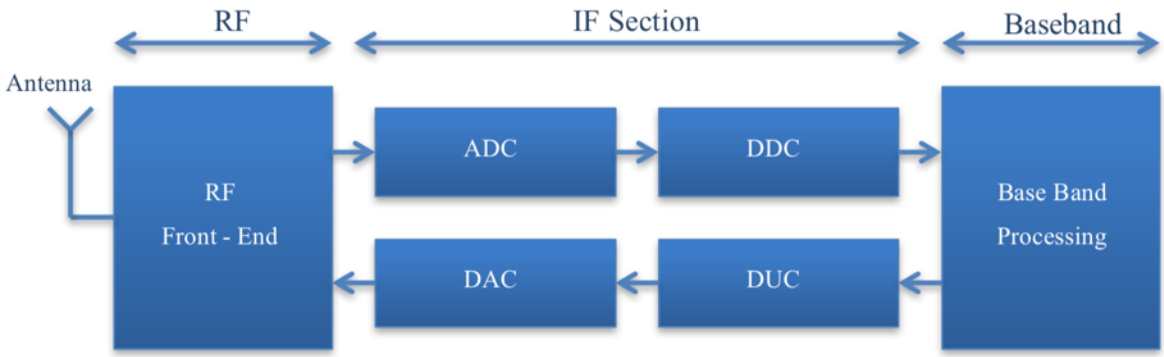


Fig. 3: The architecture of software defined radio.

2.5 GNU RADIO CONCEPT

GNU Radio is an open source toolkit which has been one of the frameworks for SDRs. Its active community support, extensive processing tools library and versatility are the major advantages that draw academic researchers, commercial companies and students to use the framework [9]. GNU Radio is also compatible with all personal computer operating systems (Ubuntu/Linux, Windows and Mac

OS) although performance varies on different platform. Ubuntu currently has been noted to perform better as it allows users with more flexibility and access to other interoperable frameworks. It can be used with the USRP, HackRF, Raspberry Pi and other hardware components for SDR development [9]. C++ is used as the language to write the signal processing, Python can also be used to write the signal blocks, but they are slower in terms of performance than blocks created using C++. Python generally is the language used for scripting to connect the signal processing blocks. The C++ and Python modules are glued together by the Simplified Wrapper and Interface Generator (SWIG). There are a lot of ready-made blocks in the GNU radio library however enabled by SWIG, blocks can also be created by users as an Out of tree (OOT) module to achieve custom or desired result. GNU radio recently has added multi core support to its interface and is able to thread different blocks unto different processing unit there by taking advantage of the number of multi-core computers. GNU radio consists of modules that are used in signal processing such as FFT modules, coding and decoding, modulation, demodulation, equalizers and signal filters. These modules are connected by data streams to form flowgraphs whic indicate the data flow direction.

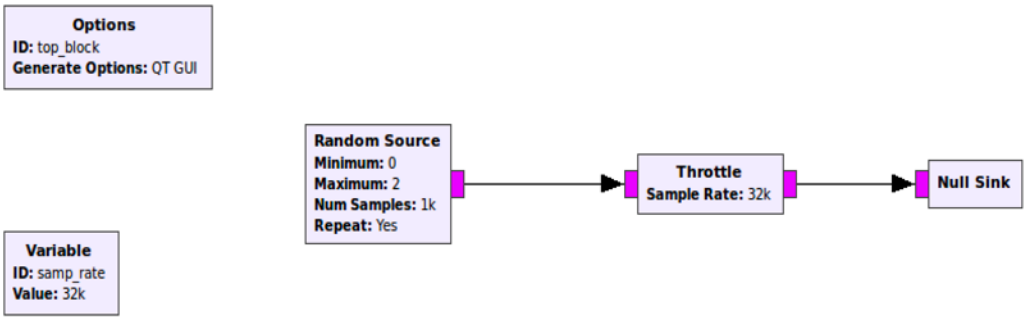


Fig. 4: Generic Flowgraph of GRC

GNU Radio Companion (GRC) is a graphical tool used for creating signal flow graph and generating flow graph source code, Thanks to a large online community that works on improving the project GRC is a very intuitive and powerful platform. Users can construct a graphical representation of a flow graph by inter-connecting blocks from GNU Radio libraries. These modules in GNU Radio Companion can be

grouped into three classes:

1. Source Block: These are GRC blocks with output ports only and their primary function is to inject data into the flow graph. An example is the random source block.

2. Input or Output Block: These GRC blocks receive streams of data from the input port, convert it and inject back into the flow graph. The output is passed to the next block through the output port. An example is the throttle block

3. Sink Block: These have input ports only, which receive streams of data or signals. One example is null sink that stores received data into a file and another example is a graphical analyzer that converts received data into graphical form.

Figure 4 provides a simple example of a flow graph comprising of a source, input or output and sink module of GNU Radio companion

2.6 STRUCTURE OF GNU RADIO MODULE

[9] GNU Radio modules come with the following subdirectories: apps, cmake, docs, lib, swig, python, include and grc These subdirectories have autoconf, libtool and automake tools that make them compile independently.

Subdirectories	**Description**
apps	Consists of complete applications installed in the system as well as GRC blocks.
Cmake	instruction files for autotools,cmake command used to locate GNU Radio libraries.
docs	Consists of instructions to extract documents from the C++ and Python files.
lib	storage for source (.cc and .h) files. These are meant for all other languages except python.
Swig	SWIG tool instructions used to build python interfaces used by C++ classes.
Python	Used to store all python files.
Include	C++ header files are stored in include/ for them to be exported

TABLE IV: Description of GNU Radio module structure .

The choice of using the GNU radio platform depends on the context what we want to detect. In this case, the drone to be detected is a device that operates with the characteristics of radio systems. GNU Radio respects in total the radio system propagation approach because it's a software platform that implements a radio system without a dedicated hardware. Additionally, GNU radio is a cheap

system because it's open source and the USRP, HackRF and other hardware are low cost device which can be easily purchased. Probably GNU Radio might be seen as less efficient than other testbeds, but its performances depend largely in part with the host computer, so is not easy to compare the computational efficiency of other testbeds. Its simple architecture allows for a easy utilization also because it has a functional graphic interface and as stated earlier it's supported by a lively web community. GNU Radio is a very versatile system used in many applications, so its applicability is guaranteed in the future.

2.7 HACKRF

HackRF One by Great Scott Gadgets is a transmit and receive capable SDR with 8-Bit ADC, 10 MHz to 6 GHz operating range and up to 20Mhz of bandwidth. It has a max sample rate of 20 million samples per second which limits its constant viewing of the spectrum to 20MHz in GNU radio environment. However, there is a limitation Since the 2.4-2.5Ghz range is 100MHz wide, it could not constantly view the entire spectrum. Although, it can scan very quickly if the channel set is narrowed down enough. The drivers for the hardware need to be installed in GNU radio companion in other to use it for transmitting and receiving signals.

2.8 DETECTION OF DRONE SIGNAL

In literature and research articles there exist some different detection methods. The detection method should be able to overcome many obstacles that could make the detection difficult.Among the existing detection methods that have been implemented so far,not all of them are independent from these factors so they cannot be efficacious. Some of the factors are that may affect detection are:

1. The typical low signal to noise ratio (SNR) in the transmissions;
2. Multi-path and fading in wireless communications;
3. The instable noise level in the channel;
4. The need for a low sensing time

2.9 DETECTION METHODS

Below some of the most important detection methods are presented and how these are limited by the negative factors is also explained.

Fig. 5: HackRF Hardware

2.9.1 MATCHED FILTER

It has been identified that the best way to detect signals with maximum SNR is to use a matched filter receiver. Its most important characteristics is the low execution time, but the signal properties to be detected is known before the process begins. This method includes the demodulation of the signal which means that the receiver should agree with the source, estimate the channel conditions and to know the signal nature. This method is mostly useful for dedicated receivers like in TV transmissions.

2.9.2 ENERGY DETECTION

This is the most basic and common approach to spectrum sensing and detection due to the fact that it has low computational and execution complexity. Unlike

matched filter detector, prior knowledge on the transmitted PU signal is not required in case of energy detector. An energy detector sets a threshold according to the noise floor and compares it with the energy of the data stream in input. This detector only requires minimal information, such as the signal bandwidth and carrier frequency. Digitally, implementation of this method uses the Fast Fourier Transform (FFT), so the absolute value of the samples is squared and integrated over the observation band.

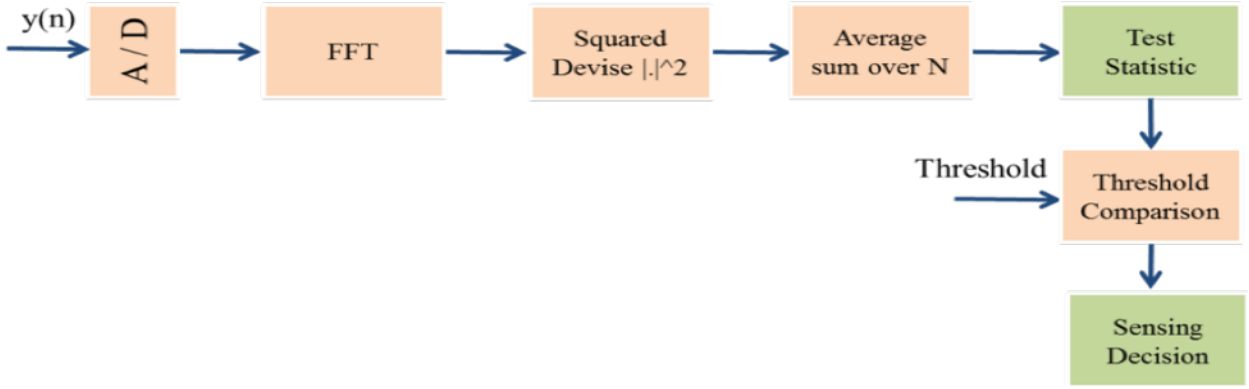


Fig. 6: Energy detection diagram.

2.9.3 FEATURE DETECTION

Most signals have statistical properties that vary periodically with time, which are called cyclostationary features. Hence, more accurate detection can be achieved by exploiting the inherent periodicity of the autocorrelation function of the signals. Modulated signals have a SCF with specific and unique characteristics so, comparing them with a database containing list of typical features, the signal detection is possible. The limitations of this method are that it needs a great computational complexity and also the knowledge of some signal parameters of the signal under test like the carrier frequency.

2.9.4 EIGENVALUES DETECTION

The main characteristics of this method is the ratio of the eigenvalues of the covariance matrix of the received signal. Eigenvalues based method of detection are of two types: maximum-minimum eigenvalue detection, which compares the ratio of the maximum eigenvalue and the minimum eigenvalue with a threshold and energy

with minimum eigenvalue detection, which compares the ratio of the average energy and the minimum eigenvalue with a threshold. The first method of detection does not require a prior knowledge of the SNR unlike the second that needs to know the SNR value. Ideally The maximum-minimum approach can overcome the noise uncertainty problem and also retains the major advantages of the energy approach. In this way the method detects signals with unknown source, unknown channel and unknown noise power. This method requires a lot of complexity to its approach.

2.10 DETECTOR METHOD COMPARISON

A Comparison of each detection method presented has advantages and disadvantages in signal detection. Table 5 presents a brief summary of different aspect that the methods are compared [21]:

1. Execution time. The ideal signal detector should work in real time so that the execution time is the shorter possible.

2. Noise rejection. Skill of the method to be immune to the white noise.

3. Knowledge a-priori. How much information the method needs to detect the signal. In CR this information should be minimum.

4. Computational complexity. Capacity calculation required to detect the signal.

5. Interference rejection. Skill of the method to be immune to the disturbs different from white noise.

	ETime	**NR**	**PK**	**CC**	**IR**
Matched filter	GOOD	MEDIUM	HIGH	LOW	HIGH
Energy Detection	HIGH	LOW	NONE	LOW	LOW
Feature detection	LOW	HIGH	MEDIUM	HIGH	HIGH
Eigenvalues Detection	MEDUIUM	HIGH	NONE	MEDIUM	LOW

TABLE V: Comparison of methods of different detection methods.

For the table the ET-Execution time, NR-Noise reduction, PK-Prior knowledge, CC-Computational complexity,IR-interfeerence rejection

CHAPTER 3

METHODOLOGY

3.1 DETECTING THE DRONE SIGNAL

Energy detection will be implemented and demonstrated using software defined radio. This will be achieved using two computers with GNU radio companion software installed and two HackRf hardware for transmit and receive.

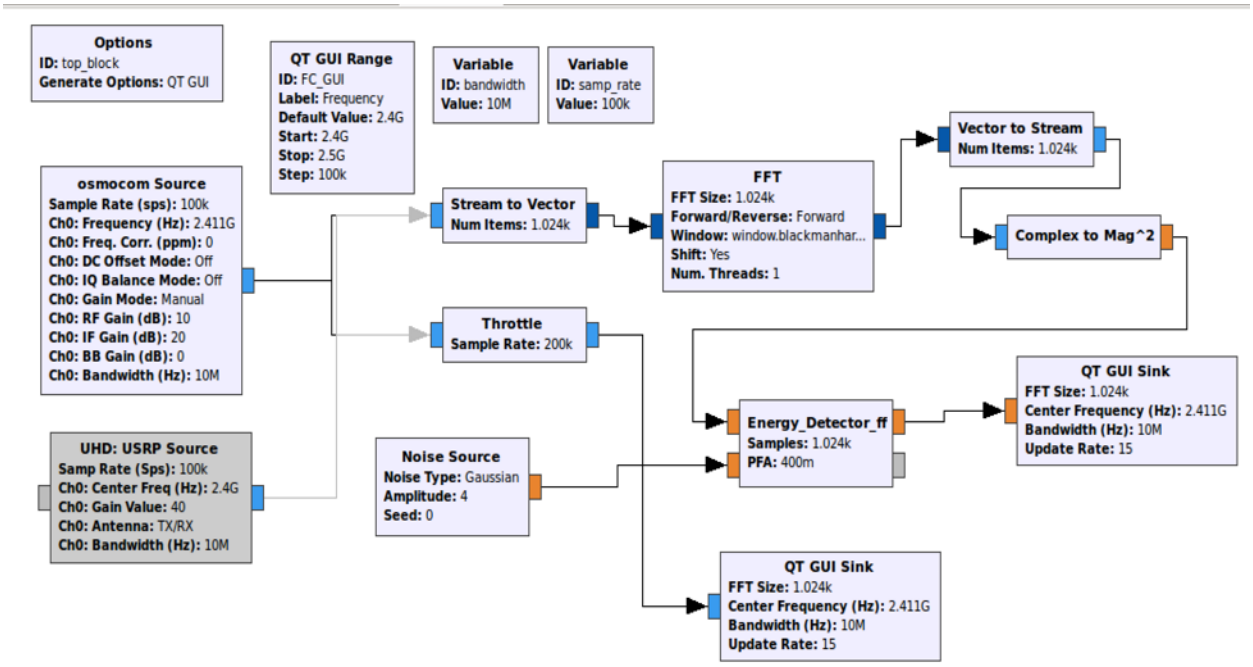


Fig. 7: Flowgraph of Energy Detector on GNU radio.

3.2 DESCRIPTION OF THE FLOWGRAPHS

Fig. 6 shows the receiver flow graph where the energy detection of the signal transmitted is performed. The Osmocom source is the HackRF where some parameters such as sample rate, frequency and bandwidth are configured. The received signal data is processed in the computer where the GNU radio is installed. The

throttle block is used to control the CPU usage of the computer. The signal received is converted into a vector of 1024 points using the stream to vector block. At the same time, the output is plotted using the QT GUI Sink block for reference purpose. This has power spectral density (PSD), Waterfall, Time Domain, and Constellation displays. Following the conversion to 1024 points is the FFT block. The FFT block performs fast Fourier transform on the incoming stream of data and the output is sent to complex to mag^2. Windowing is also done in this block in order to minimize edge effects and prevent spectral leakage in the FFT spectrum. Blackman Harris window was chosen for windowing purpose. Next is removal of the scaling made by the FFT block itself. Then, the magnitude of the complex value is squared ($a + -jb => a^2 + b^2$) by the complex to mag^2. The output of this block is then sent to the custom-made $Energy - detection - ff$ block which is where the detection algorithm is implemented. The block allows user to choose the number of samples and the probability of false alarm to be used for calculation. From the input data, the block calculates the mean (average) of the received signal, its variance and standard deviation. This parameter is used to calculate the dynamic threshold for detection. The log value of signal average and threshold are calculated. The average signal energy is then compared with the threshold to determine if the signal is present or not. If the signal energy is greater than the threshold then signal is present, and the signal energy is sent to the output. The QT GUI Sink block is used to display of the power of the received signals in the frequency domain for visualization. We use the formula below to implement averaging in the detector block [10]:

Total power in N samples set

$$Y = \sum_{n=1}^{N} \frac{1}{N} X(n)^2$$

To make the decision if signal is present or absent, we need to determine the Threshold,

$T = \mu + Q^{-}1(pfa)\sigma$

where μ and σ are the mean and standard deviation of the time averaged FFT points respectively, inverse Q () represents the inverse Q-function and Pfa is the probability that the detection module chooses H1 while the correct decision is H0. The decision rule can be defined as

Detection Decision

$Y > T \rightarrow H1(signalpresent)$

and

$Y <= T \rightarrow H0(Signalabsent)$

3.3 ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING (OFDM

[1] Orthogonal Frequency Division Multiplexing (OFDM) is a basic building block for many of the current modulation schemes including; 802.11 WLAN, 802.16 WiMAX, and 3GPP LTE. Orthogonal Frequency Division Multiplexing (OFDM) is a digital multi-carrier modulation scheme that uses multiple subcarriers within the same single channel. It makes use of a large number of closely spaced orthogonal subcarriers that are transmitted in parallel. Each of the subcarrier can be modulated with one of the conventional digital modulation schemes (such as QPSK, 16QAM, etc.) at a low symbol rate. However, the combination of more than one subcarrier enables data rates similar to conventional single-carrier modulation schemes within equivalent bandwidths.

There is a notable difference between OFDM scheme and that of traditional FDM in the following ways:

1. Multiple carriers (called subcarriers) carry the information stream
2. The subcarriers are orthogonal to each other, and
3. A guard interval is added to each symbol to minimize the channel delay spread and inter symbol interference.

In the figure shown below, one will get a better understanding of the main concepts of an OFDM signal and the inter-relationship between the frequency and time domains. In the frequency domain, many adjacent subcarriers or tones are each independently modulated with complex data. Then an Inverse FFT transform is performed on the frequency-domain subcarriers so as to produce the OFDM symbol in the time-domain. After which in the time domain, guard intervals are inserted between each of the symbols to prevent inter-symbol interference at the receiver which is caused by multi-path delay spread in the radio channel. It is possible for multiple symbols to be concatenated and used to create the final OFDM burst signal. At the receiver an FFT is performed on the OFDM symbols to recover the original data bits.

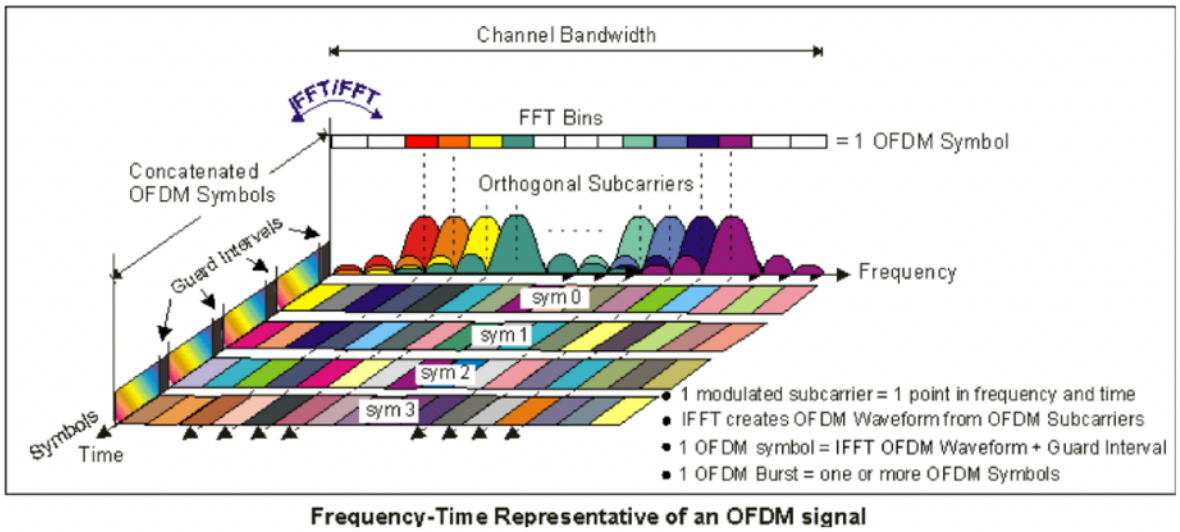


Fig. 8: Representation of OFDM Signal

3.3.1 UNDERSTANDING THE CONCEPT OF ORTHOGONALITY

In the frequency domain of signal processing, each of the transmitted subcarrier usually results in a sine function spectrum with side lobes that produce overlapping spectra between subcarriers. This therefore results in subcarrier interference except at orthogonally spaced frequencies since the individual peaks of subcarriers are all line up with the nulls of the other subcarriers. In addition, this overlap of spectral energy in noted to not in any way interfere with the systems ability to recover the original signal. At the receiver end, the receiver multiplies (i.e., correlates) the incoming signal by the known set of sinusoids to recover the original set of bits sent.It can be noted that the use of orthogonal subcarriers allows more subcarriers per bandwidth resulting in an increase in spectral efficiency. In a perfect OFDM signal, orthogonality prevents interference between overlapping carriers. In OFDM systems, the subcarriers will only interfere with each other if there is a loss of orthogonality. For example, frequency error will cause the subcarrier frequencies to shift so that the spectral nulls will no longer be aligned resulting in inter-subcarrier-interference.

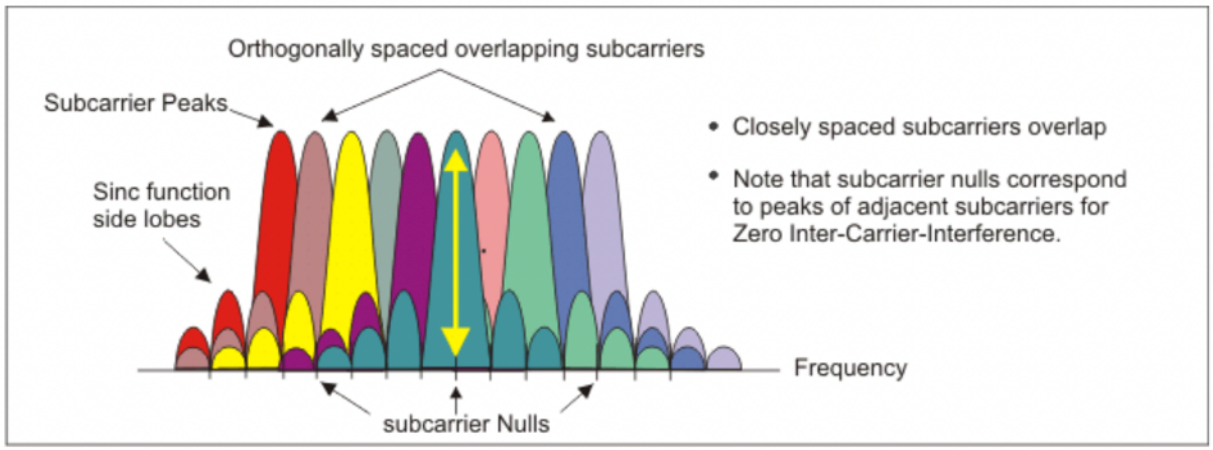


Fig. 9: OFDM Signal Breakdown

3.3.2 SOME KEY PARAMETERS: OFDM MODULATION CLASSIFICATION

OFDM signal classification based on parameters center frequency, OFDM symbol duration, CP length, and number of subcarriers [2]. Cyclostationary Feature based OFDM classification [3] considers Frame Preamble Cyclostationary Signature, Pilot Subcarrier Cyclostationary Signature. Also, IEEE 802.11 Major OFDM physical layer Parameters Authors of [4] reported that for the classification of OFDM Modulation the following parameters are chosen number of carriers, symbol duration, bandwidth and bit rate.

The OFDM system parameters which are used for Modulation Classification (MIMO OFDM) [15]: Carrier frequency, Total bandwidth, Number of subcarriers, Subcarrier spacing, Length of cyclic prefix, Sampling period and OFDM Frame duration (CP included). The following parameters can also be considered for OFDM classification [16]: Uncoded data rate = $390Mbs$, Source symbol duration $TD = 15.385ns$, Transmission symbol duration $TS = 12.5ns$, Total OFDM symbol duration $TS = 3.60ms$, Guard time $TG = 0.4ms$, Subcarrier spacing $FS = 0.3125MHz$, Nyquist bandwidth BN (including pilot and DC subcarriers) = $76.5625MHz$

The spectrum of OFDM is considered to be rich in features and can be used to estimate some OFDM parameters such as number of subcarriers and length of the CP in an OFDM symbol [17]. These OFDM waveform parameters can be used to

automatically recognize or classify different OFDM waveforms, which are important for cognitive radios, coexistence of heterogeneous networks and signal intelligence. Besides distinguishing of OFDM signal from single carrier, some vital parameters of OFDM signal should be extracted for further processing [18]. These parameters include, but not limit to, the number of subcarriers, OFDM symbol duration and cyclic prefix (CP) duration. A cyclostationarity test is applied to the incoming signal to detect the OFDM symbol duration Ts. The CAF of an OFDM signal exhibits peaks at$\alpha = c/T$ and $\tau = Tu$, where $Tu = T - Tg$ is the data duration time, normalized by the sampling period Ts [19]. In an OFDM signal, these peaks are introduced by built-in periodicity due to equally spaced sinusoidal carriers, cyclic prefix and pilot patterns. In fact in a non-cyclostationary signal, such as AWGN, the CAF does not show any peak for $\alpha \neq 0$. The AWGN has a peak only for $\alpha = 0$ and $\tau = 0$ and we use this feature to distinguish between the presence or absence of PU signals.

3.3.3 EXAMPLE OF SOME OFDM KEY PARAMETERS

Given that there are different available parameters in literature that researchers have used as a feature to classify OFDM signal from non OFDM signal.below is an example of what some of these parameters look like

Number of data subcarriers	48
Number of pilot subcarriers	4
Subcarrier frequency spacing	156.2kHz
Occupied Bandwidth	8.28125MHz
Short training sequence duration	16μs
Long training sequence duration	16μs
Training sequence guard interval	3.2μs
PLCP preamble duration	32μs
Guard interval duration	1.6μs
POFDM Symbol duration	8μs

TABLE VI: Comparison of methods of different detection methods.

In our experiment we make use of four parameter.The OFDM parameter estimation block of the GR-Inspector which estimates these parameters was incorporated to our already developed energy detection block and the result from the two block

merged together to form one block of the signal parameter estimation. Below is a brief description of the parameters :

Cyclic Prefix Length Determination: The useful symbol length (Tu) is the inverse of the subcarrier spacing. Then, the CP length is ($Tg = G * Tu$), where G is (Tg/Tu) ratio. The choice of G is made according to channel parameters. The total OFDM symbol length consists of the useful symbol length and the CP length ($TOFDM = Tg + Tu$). [20]

1. Number of Subcarriers: it consist of both the pilot and data carriers
2. FFT size: the number can vary from 64, 128, 256, 512 and so on
3. OFDM Symbol duration: This is the Inverse of subcarrier spacing
4. Subcarrier Spacing: $spc = 1/Ts$, Ts is symbol time

3.3.4 IMPLEMENTATION OF OFDM PARAMETER ESTIMATION

In the OFDM parameter estimation block in the GNUradio, some algorithms were used to implement the estimation of some OFDM parameters. Below are the necessary step that were taken to implement the algorithms:

OFDM parameter estimation algorithm steps are as follows:

1. Calculate the autocorrelation function for input vector to estimate subcarriers length (FFT) using the discrete autocorrelation function

2. Calculate the CAF, the easiest way to describe the CAF (cyclic autocorr function) is that it is a coefficient in the Fourier series expansion of the time-varying autocorrelation of a cyclostationary random process (cyclostationary signal). Calculate time variant cyclic correlation function to find cyclic prefix length (CP).

3. Then determine subcarrier spacing = $subspc = (sample_rate/FFTlength) and symboltime = symtime = (1/subspc)$.

3.3.5 IMPLEMENTING OFDM PARAMETER ESTIMATOR IN GNU RADIO

From the Gr-inspector module which was part of the Google Summer of Code (GSoC) program 2016, We culled one of the available blocks to suit our experiment. The Gr-inspector module has a module developed which implements the parameter estimation in GNU Radio Companion. This GNU Radio module is particularly developed so as to realize signal analysis abilities in typical block-structure. The module is capable of the following:

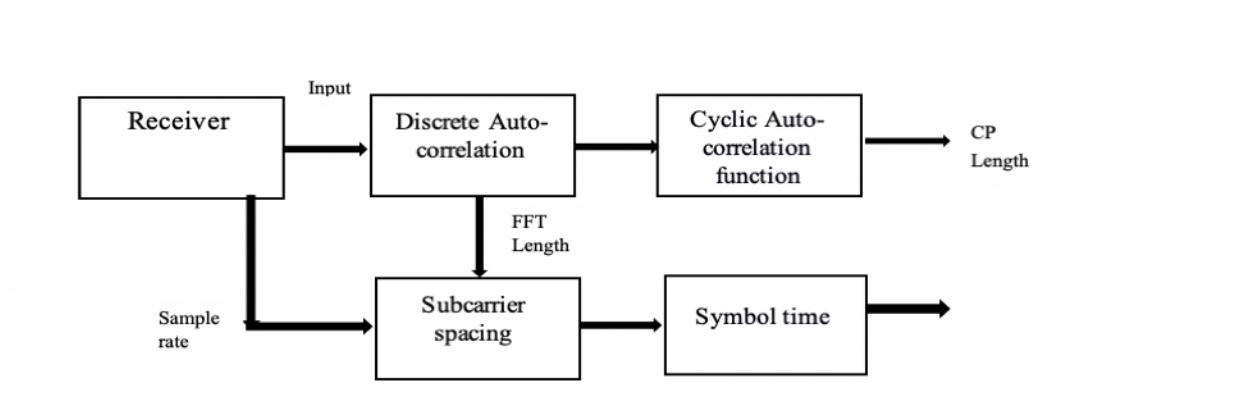


Fig. 10: Breakdown of OFDM parameter estimation

1. Energy detection of continuous signals
2. Filtering of detected signals
3. OFDM parameter estimation (carrier spacing, symbol time)
4. Blind OFDM synchronization
5. Resampling of signals to constant rate
6. 3D Visualization of FAM data, from gr-specest
7. Using TensorFlow models for AMC

For our problem we are mainly interested in the OFDM parameter estimation block which performs estimation of four OFDM parameters and output them as a message block (carrier spacing, symbol time,FFT size,Symbol duration). We incorporated this block with the energy detector implementation and combined the output from both blocks to get the parameter for detection and estimation together as one unit

3.4 MACHINE LEARNING CLASSIFICATION

What is machine learning? There are different definitions but generally it is a branch of artificial intelligence that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data. The idea is for machines are able to learn and make predictions without explicitly being programmed. As intelligence requires knowledge, it is necessary for the computers

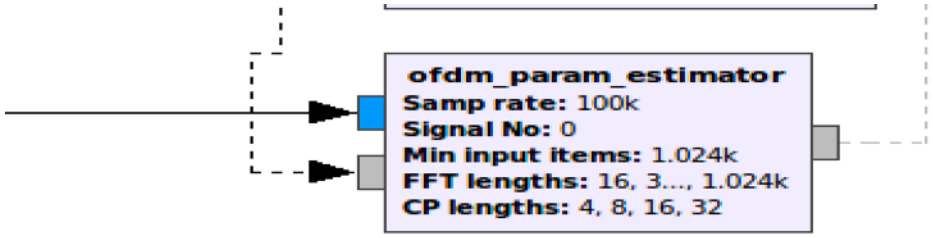


Fig. 11: OFDM estimator in GNURadio

to acquire knowledge. We incorporated a machine learning block to our work by developing a custom machine learning testing block on GNURadio

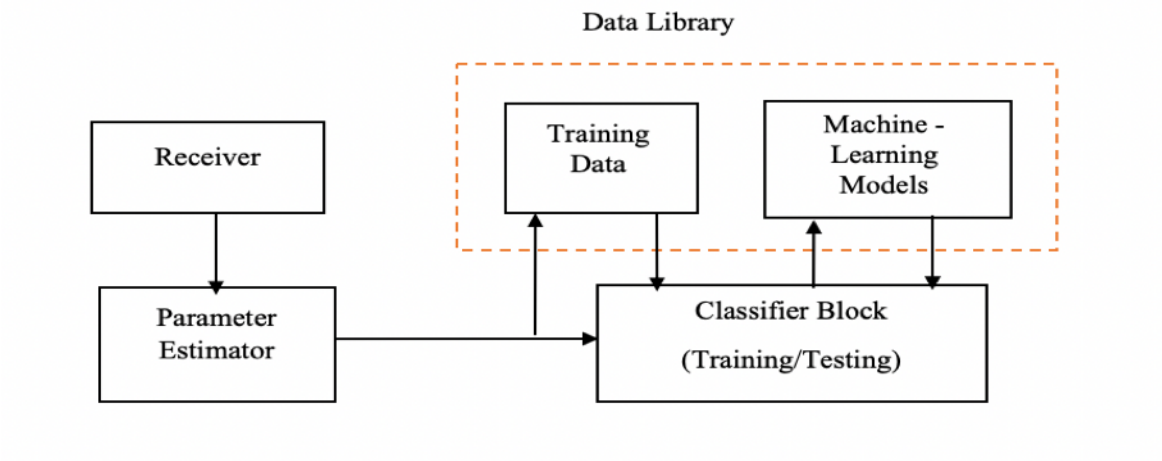


Fig. 12: Breakdown of the machine learning module

The estimated parameters are initially used to construct training data and stored in data library for further processing. The classifier in training mode creates models based on selected machine learning classification algorithm. Then, testing of the new signal is done with the selected model from data library. The machine learning classifiers are discussed below.

3.4.1 MACHINE LEARNING CLASSIFIER

Generally, machine learning algorithm are usually organized into different taxonomies which are based on the final outcome. Some of the common algorithm types

are:

1. Unsupervised learning: in this case it the algorithm models a set of inputs based on some characteristic. Labelling of the data samples is not available in this situation

2. Supervised: A set of function is generated by the algorithm that maps the data inputs to a desired output. The learner in this case is required to learn how to approximate the behavior of a function which maps an input vector into one of many classes by cross checking and taking a close look at different input-output examples of the function. Our experiment is based on this type of algorithm

3. Semi supervised learning: this combines the concept or examples of both supervised and unsupervised algorithm and generates an appropriate classifier.

4. Reinforcement learning: In this case the algorithm learns a policy of how to act when given an observation of the world.

5. Learning to learn: in this case the algorithm learns by its own inductive bias taking into consideration previous experience

Below some of the machine learning classifiers are discussed briefly

Decision Tree

A decision tree has a flowchart-like structure in which each of its internal node represents a test on an attribute for example in a coin toss scenario, whether a coin flip comes up heads or tails, each of the branch represents the outcome of the test, and each leaf node represents a class label which is the decision taken after computing all attributes. The paths represented from the root to leaves denotes the classification rules. Typically, a decision tree algorithm consists of three types of nodes:

1. Decision nodes - this are usually represented by squares
2. Chance nodes - usually represented by circles
3. End nodes usually represented by triangles

Support Vector Machine

A Support Vector Machine (SVM) is a classifier, which distinguishes the various classes of data by the use of a hyper-plane. SVM is modelled with the training data and it outputs the hyper-plane in the test data. [26]. The SVM model works by trying to find the space in the matrix of data where different classes of the investigated data can be widely differentiated and draws a hyper-plane.

Random Forest Classifier

The random forest machine learning algorithm is a classifier which consist of a collection of different decision trees, whereby each of the tree is constructed by applying an algorithm A on the training set S and an additional random vector V where V is the sampled (i.i.d).(independently and identically distributed) from some available distribution. The prediction of the random forest is obtained by a majority vote over the predictions of the individual trees [13,255]. Usually the Random Forest algorithm works in the following steps:

1. Picks random K data points from the training data.
2. Builds a decision tree for these K data points.
3. Chooses the Ntree subset from the trees and performs step 1 and step 2
4. Decides the category or result on the basis of the majority of votes.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x1, ..., xn$ with responses $Y = y1, ..., yn$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, ..., B$:

1. Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
2. Train a classification or regression tree fb on Xb, Yb.

Nave-Bayes classifier

For this classifier ,when Given a classification scenario with a multinomial event model for example, a given samples (feature vectors) in this case represent the frequencies of which certain events have been generated by a multinomial $(p-1, , p-n)$ where $p - i$ is the probability that event i occurs (or K such multinomial in the multiclass case). A feature vector $x = (x - 1, , x - n)$ is then a histogram, with $x - i$ counting the number of times event i was observed in a particular instance.

K-Nearest Neighbor classifier

K-nearest Neighbor algorithms are among the simplest of all machine learning algorithms. The idea is to memorize the training set and then to predict the label of any new instance on the basis of the labels of its closest neighbors in the training set. The rationale behind such a method is based on the assumption that the features

that are used to describe the domain points are relevant to their labelling in a way that makes close-by points likely to have the same label. Furthermore, in some situations, even when the training set is immense, finding a nearest neighbor can be done extremely fast.K in the k-nearest neighbor is the number of the data points closest to the new instance. We can say for example, if k =2 then the algorithm will choose the nearest two instance or if k = 5, then the algorithm will choose the closest five neighbor instances and based on these will classify them accordingly.

Logistic Regressionclassifier

it will be easier to begin an explanation of logistic regression with an explanation of the standard logistic function. The logistic function is a sigmoid function, which takes any real input t, $t \in R$, and outputs a value between zero and one; for the logit, this is interpreted as taking input log-odds and having output probability.

3.4.2 MACHINE LEARNING IN GNURADIO

Currently there are no readily available blocks in GNU radio that can perform machine learning classification. But as we all know machine learning is easily implemented on Python and GNU Radio works well with Python programming. We developed a custom OOT block in GNU Radio which will be used to perform real time testing of the incoming input signal to the classification module. Even though we carry out real time testing on GNU Radio platform the training phase of the experiment is still done outside of GNU Radio for now.

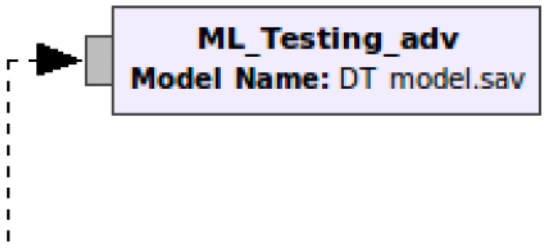


Fig. 13: Custom Machine learning testing block on GNURadio

3.4.3 TOOLS USED FOR THE MACHINE LEARNING

In order to carry out training and testing in machine learning there are some additional libraries which are important in the realization of the task. All of these too are free and open source which is one of the reasons we chose to use them. Below are the list of some of the tool and a brief description of each:

1. Python 3.5
2. NumPy 1.11.3
3. Matplotlib 1.5.3
4. Pandas 0.19.1
5. SciPy and Scikit-learn 0.18.1

Python

Python is a high-level general programming language and is very widely used in all types of disciplines such as general programming, web development, software development, data analysis, machine learning etc. [2] We choose to use Python for our project because it is very flexible and also easy to use, there is readily available documentation and online community support is very large.

NumPy

[23] NumPy is very powerful package that enables us for scientific computing. It comes fully loaded with a lot of sophisticated functions and is able to perform N-dimensional array operations, basic and complex algebra and Fourier transform as in the case of signal processing. NumPy is used very well in image processing, data analysis and also different other libraries are built above NumPy and NumPy acts as a base stack for those libraries.

Pandas

Pandas is an open source BSD licensed software which is specially written for python programming language [24]. It provides a complete set of data analysis tools for python and is best competitor for R programming language. Different operations such as reading data-frames, reading csv and excel files, slicing, indexing, merging, handling missing data and so on, can be easily performed using Pandas. The most important feature of Pandas is that it can perform time series analysis.

SciPy

SciPy is a collection of fundamental mathematical functions and is built on the top of Numpy which was mentioned earlier, while Scikit-learn is widely used popular library for machine learning, it is third party extension to SciPy [25]. Scikit-learn includes all the tools and algorithms needed for most if not all the machine learning tasks. Scikit-learn supports regression, classification, clustering, dimensionality reduction, and data pre-processing. For our task, Scikit-learn is used because it is based on Python and can interoperate to NumPy library. It is also very easy to use just like Python.

3.5 DESCRIPTION OF THE FEATURES USED

For the RF-toolbox implementation in GNU Radio all the blocks are connected to create a single data segment which will be used for detection and classification. Below is brief description of the features used in the experiment

3.5.1 ENERGY DETECTOR PARAMETERS

In the detection block two parameters which are among the features were extracted :

1. Signal power: Average of the input signal magnitude after converting to frequency domain using FFT block.The value of the signal power is important to determine the presence of activities in the observed spectrum and can also be a factor that shows how far or close the detected drone is. for example if during the experiment a signal power of 8dB $\pm$ 4dB is observed and in another instance a signal power of -2dB $\pm$ 4 is observed, this shows the the detected drone is at different distance when the test is done.

2. Threshold/Decision : Threshold is used to make decisions of signal availability, if signal is greater than the calculated threshold, signal is present and decision is 1.If less signal is absent and decision is 0. When drone is On and transmitting we expecct that the decision is 1 since the average signal is greater than the threshold which represents the base of spectrum occupation. When the decision is 0 we can discern that either the drone is not transmitting in the immediate environ or the drone is transmitting from a distance far away to be greater than the threshold

3.5.2 OFDM PARAMETER ESTIMATOR

In the parameter estimator block, four OFDM parameters were extracted and used as features:

1. Cyclic Prefix Length: As discussed in earlier section CP which is circular extension associated with OFDM transmitted signal is one of the key OFDM parameters which is used for the purpose of eliminating inter-carrier-intereference (ICI) and inter-symbol-interference (ISI). Usually OFDM system uses fixed and large CP length to tolerate poor channel condition and we assume that drone communication system will be optimized to perform well in different condition hence the adoption of a method that estimates CP length of an OFDM signal.

2. FFT size: The fast Fourier transform is an algorithm which computes the discrete fourier transform. its essential in the transformation of the digital signal from the to time domain into frequency domain and this is why FFT is widely used in digital signal processing and also many other applications for example communications. This transform is done using different bin sizes 64, 128 ,256 and 1024. The OFDM parameter does this an the FFT size used will be provided as one of the output.

3. OFDM Symbol time or duration: OFDM symbol consist of the FFT period and the CP. The symbol duration corresponds to the time for every symbol. The symbol time is Inverse of subcarrier

4. Subcarrier Spacing: OFDM system break down the available bandwidth into narrower sub carriers before transmitting the datas in parallel streams spacing of the subcarriers of the OFDM symbol $= 1/Ts$, Ts symbol time

CHAPTER 4

EXPERIMENT AND RESULT

4.1 SETUP AND TESTING

In this chapter, the detection and classification module using GNU radio companion will be discussed. For our experiment, a customized machine learning testing block, energy detection and parameter estimation block were developed to detect and classify drone signals using gnu radio companion and hackRF hardware. The RF signal raw data will be collected after implementing and connecting all the blocks. In order to carry out the experiment and collect data from the detection module, it is necessary to be able to ascertain and know for sure the drone transmit frequency. This way, we can monitor the spectrum and collect data for training and then test the classification using the custom machine learning testing block. We collected data for different scenarios that will be used as the base of our classification. First instance is when the phantom drone is turned ON and is the only signal being transmitted; the second instance is when the Mavic drone is ON and also is the only signal being transmitted. From the drone settings, the DJI phantom and Mavic RF transmission mode was changed from Auto to Custom. The Custom setting allows user to choose the specific frequency the drone is transmitted on. This makes it easier for us to tune and monitor the GNU radio companion frequency spectrum during testing. In order to collect data from the detection block, we need to be able to transmit at a specific frequency, monitor and collect data for training, and then test the classification using the custom machine learning testing block. For this experiment, the drone is transmitting on channel 13 and the receiver module on GNU radio is centered at 2.411Ghz so as to detect the presence of activity on the channel. Table 7 shows the setup for the experiment.

S/N	Hardware	Software
1	DJI phantom 4	GNU Radio-Companion
2	DJI Mavic	
3	HackRF	
4	Hp Laptop	

TABLE VII: Hardware and software needed for the experiment.

4.2 HOW IT WORKS

Datas for the two scenarios was collected for over 10 minutes at different conditions so as to have varying instances for each scenario. These collected data were used for machine learning training to build a class. The Testing block uses data trained based on support vector machine model of machine learning training to make predictions. The features used for the classification are signal power, decision/threshold for detection (energy detection) and OFDM parameter (subcarrier space, symbol time, fft length, cp length). Using different machine learning model to train the features we expect the decision model to predict Mavic when signal transmitted is from Mavic and phantom the signal is from the phantom drone in real time

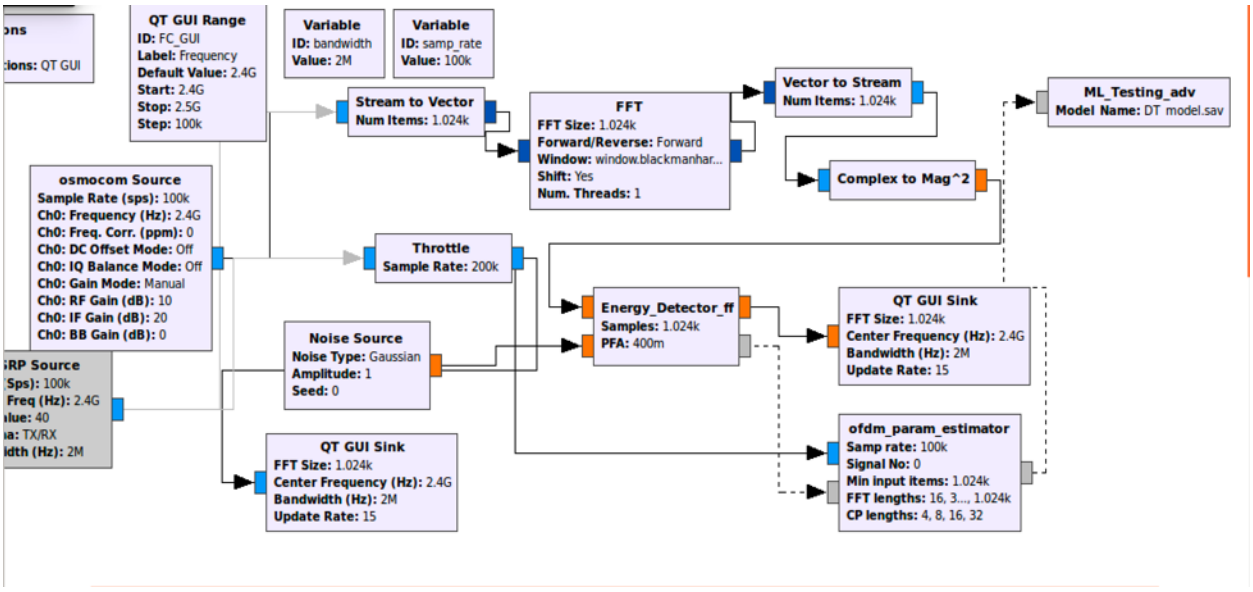


Fig. 14: GNURadio flowgraph for detection and classification.

4.3 MACHINE LEARNING DATA TRAINING AND TESTING CASE 1

For our experiment we will be using the supervised method of machine learning classification to train the models which will be used for detection and classification, and the data used for classification will be labeled accordingly. The training data contains parameters which will be used for classification, and same amount of data for each object to be classified is collected. For the first test of supervised training. data were collected for 200 samples of when the phantom drone was ON and 200 samples of when Mavic was ON. Both samples were labeled to create the class for classification purposes. The total sample used for the training data is 400 in total. We decided to start with smaller training samples and later increase the number of samples used for training. For the second test, the total sample used for the training is 3000 in total; 1500 for when Mavic is turned ON and another 1500 for when phantom is turned ON.

Subcarrier-Spacing	Symbol-time	fft-length	cp-length	Signal-Power	Detection	class
97.65625	0.01024	1024	32	463.655823	1	Mavic
390.625	0.00256	256	32	466.324219	1	Mavic
97.65625	0.01024	1024	32	462.016113	1	Mavic
195.3125	0.00512	512	32	470.765259	1	Mavic
97.65625	0.01024	1024	32	470.765259	1	Mavic
1562.5	0.00064	64	32	452.020081	1	Phantom
195.3125	0.00512	512	32	452.105621	1	Phantom
1562.5	0.00064	64	16	450.157654	1	Phantom
390.625	0.00256	256	16	450.232574	1	Phantom
390.625	0.00256	256	32	265.731476	1	Phantom

Fig. 15: Sample of collected data used for training

4.3.1 CASE 1 TESTING AND RESULT

Based on the data used for training and testing below is the performance classification metric in Python after the training data was split into training and testing for validation.

The figure below shows the chart consisting of the performance metric generated in Python using the KNN classifier for trainning the data samples.

We decided to use KNN after performing an accuracy comparison among different classifiers like decision tree,linear regression and support vector machine

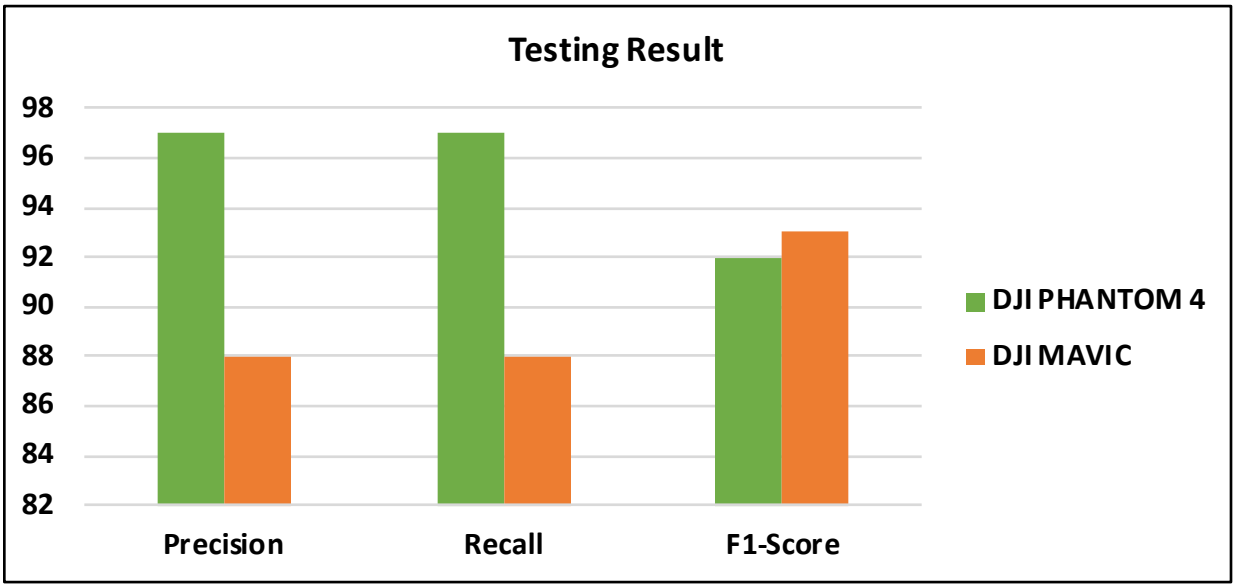


Fig. 16: Experimental Setup.

4.3.2 EXPECTED OUTCOME OF REAL TIME TESTING

After implementing the training model and creating a testing file, we need to test the performance of the trained model in real time to also determineits accuracy. When No signal is transmitted, we expect the machine learning result to be:

Fig. 17: Prediction output for mavic drone .

When its a drone signal, we expect the machine learning result to be:

Fig. 18: Prediction output phantom drone .

The class label during training was done to output phantom for when a phantom signal is detected and Mavic when a Mavic signal is detected. We decided to use this label instead of binaries 0 and 1 so that it will be easy to explain to observers what is really going on during the test.

4.3.3 REAL TIME TESTING

We carried out a real time detection and classification of drone signal on GNU radio and observed the output of the classification module

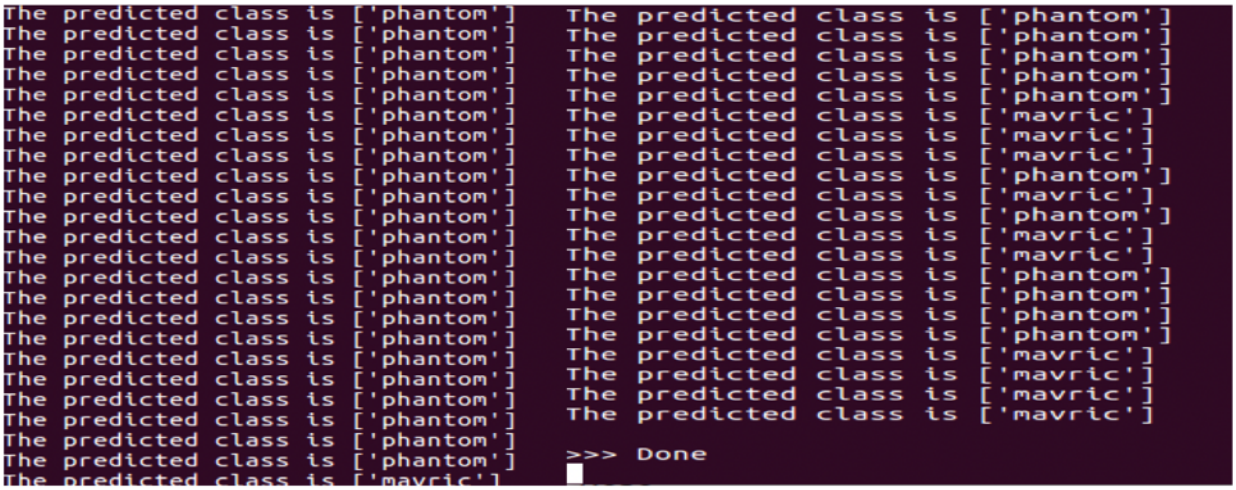


Fig. 19: Prediction output a. when phantom is ON and b. when Mavic is ON.

4.3.4 OBSERVATION OF REAL TIME TESTING

As we can see from the above capture in Figure 19, our machine learning testing block correctly predicts signal presence based on the trained data but not with very good accuracy. Analysis was carried out to determine the accuracy percentage among a set of the tested data. Below are our findings in Table VIII and Table IX.

	Actual Positive	**Actual negative**
Predicted Positive	2460	390
Predicted negative	863	2168

TABLE VIII: ML Testing Metric 1.

Recall	0.740294914
Precisionnegative	0.863157895
True Positive rate	0.740294914
False Positive rate	0.152462862

TABLE IX: ML Testing Metric 2.

4.4 DRONE DETECTION SNR TESTING

In order try to evaluate and test the detection capability of the module RF-class toolbox, we decided to carry out distance analysis test to determine how far away the drone can be from the detection module and its presence not to being detectable even when its transmitting.

For this test the Distance between the drone and the detection module was varied and detection data was collected and used for the further analysis.

Fig. 20: SNR Test Setup.

4.4.1 RESULT OF DRONE DETECTION SNR TESTING

Row Label	Detect Rate	No of wrong prediction	No of right prediction
Less than 5m	0.9818	91	4909
5m	0.888	560	4440
10m	0.8462	769	4231
20m	0.8134	933	4067
50m	0.7162	1419	581
greater than 50m	0.5604	2198	2802

TABLE X: Hardware and software needed for the experiment.

From the distance test conducted we can observe that the detection probability decreases as the drone becomes farther from the detection module. The received signal energy decreases significantly whe the drone is away from the module by a distance of 60-100metrers. We can conclude that for now due to hardware capabilities we can only detect the presence of drone under a very good SNR. when the SNR is low the detection performance also reduces drastically

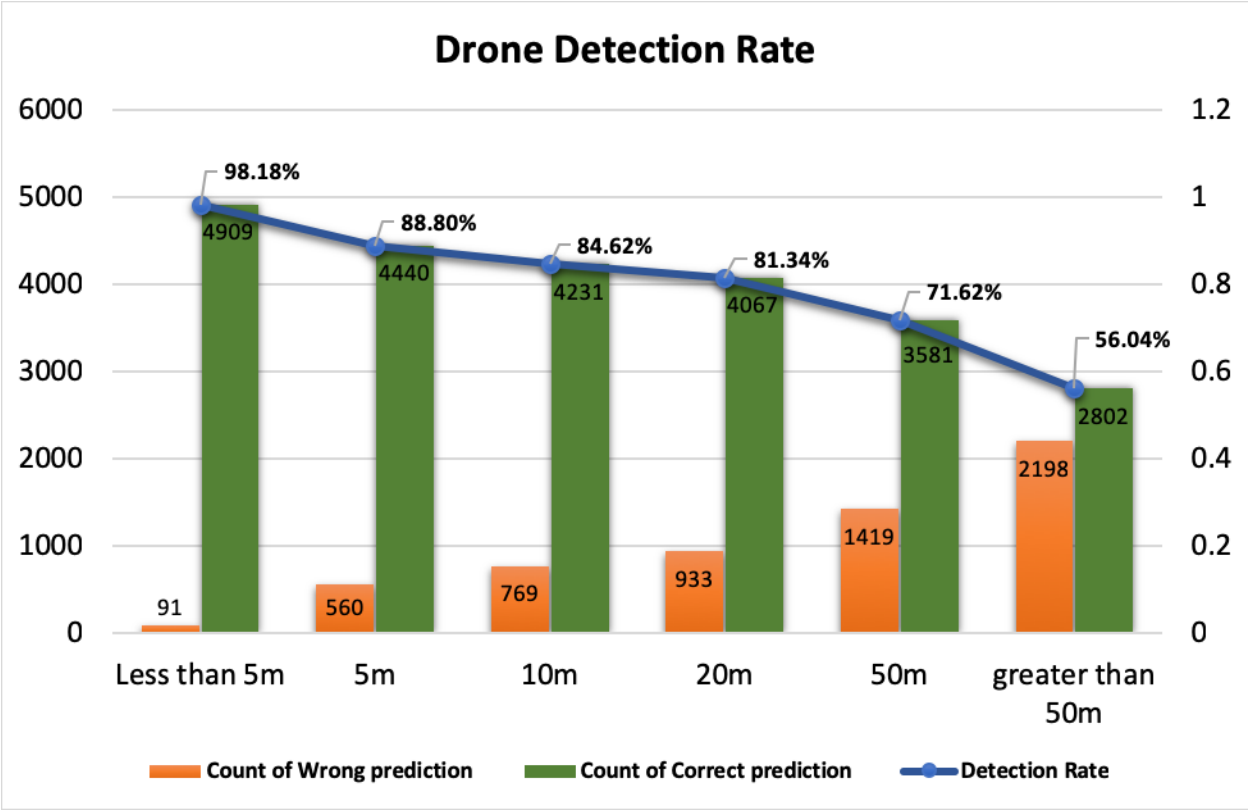


Fig. 21: Chart of the detection rate.

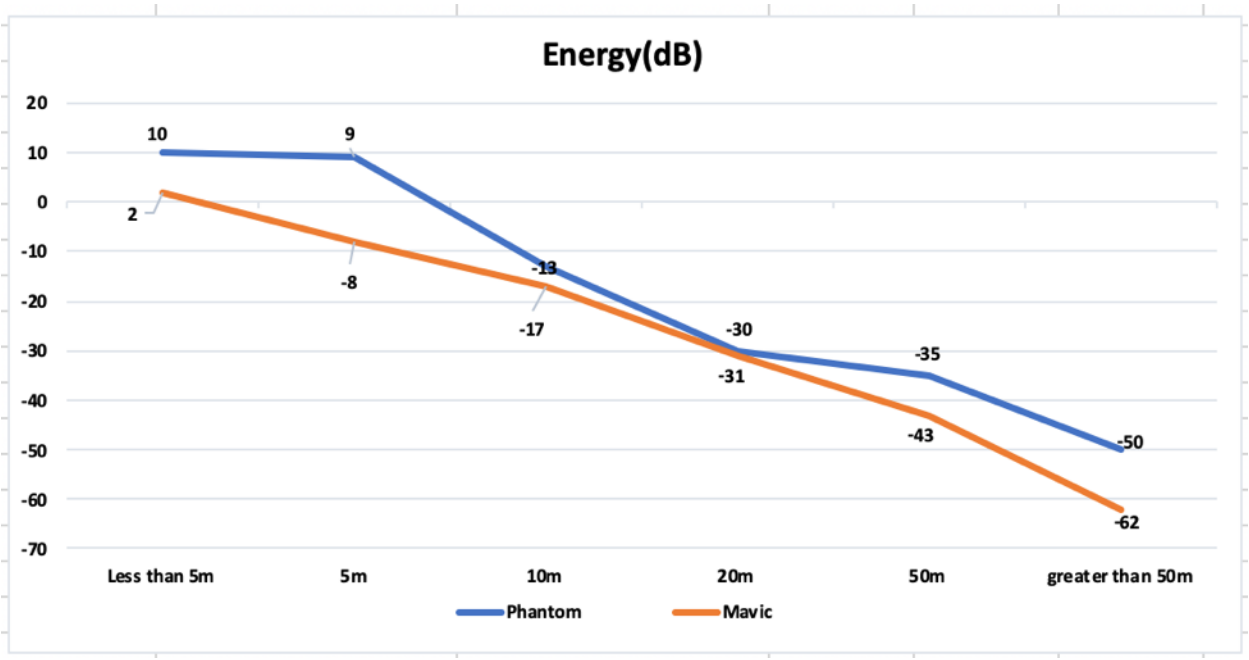
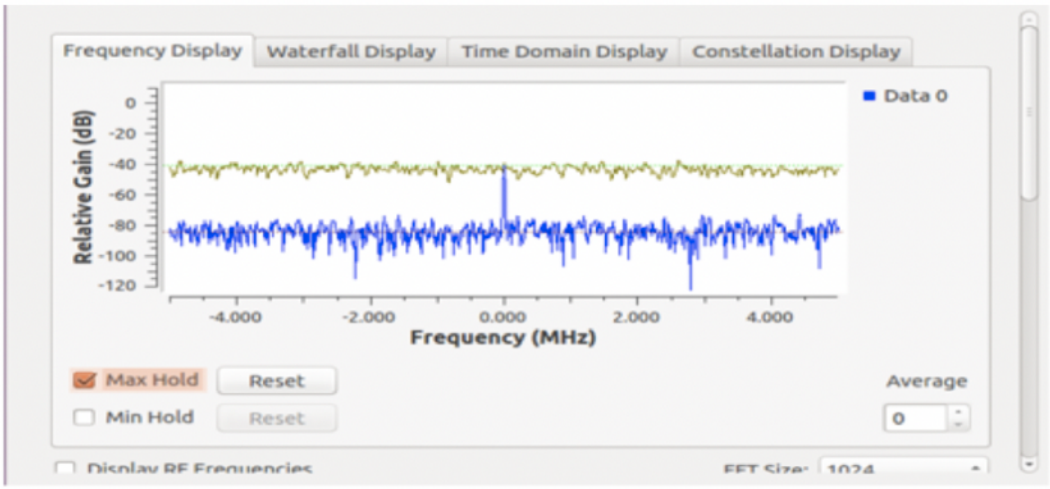
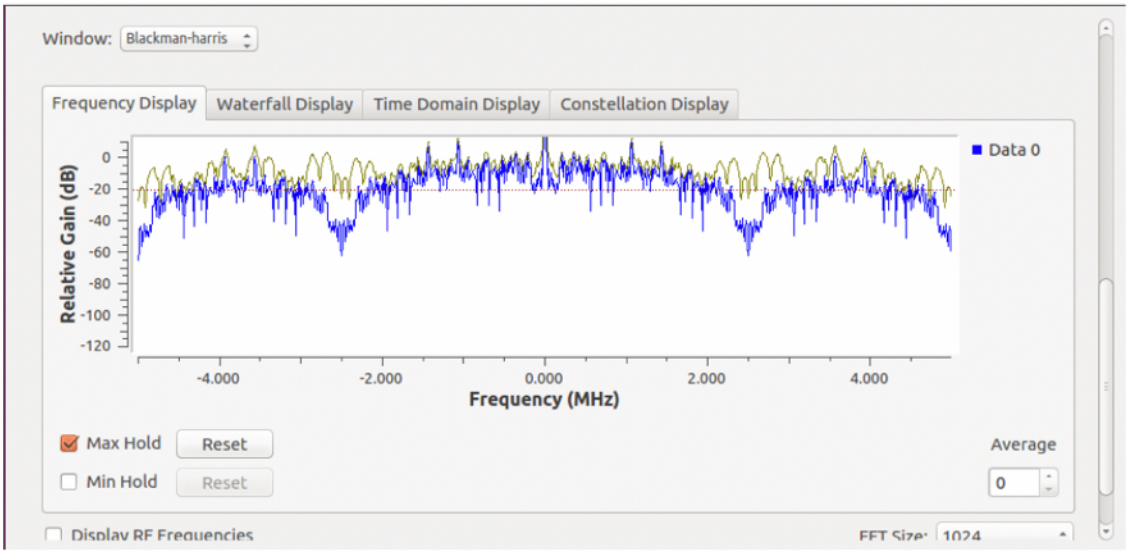


Fig. 22: dB plot of the avearage received energy.



Capture of Energy Detector sink when drone is turned OFF



Capture of Energy Detector Sink when drone is turned ON

Fig. 23: Energy Detection Output.

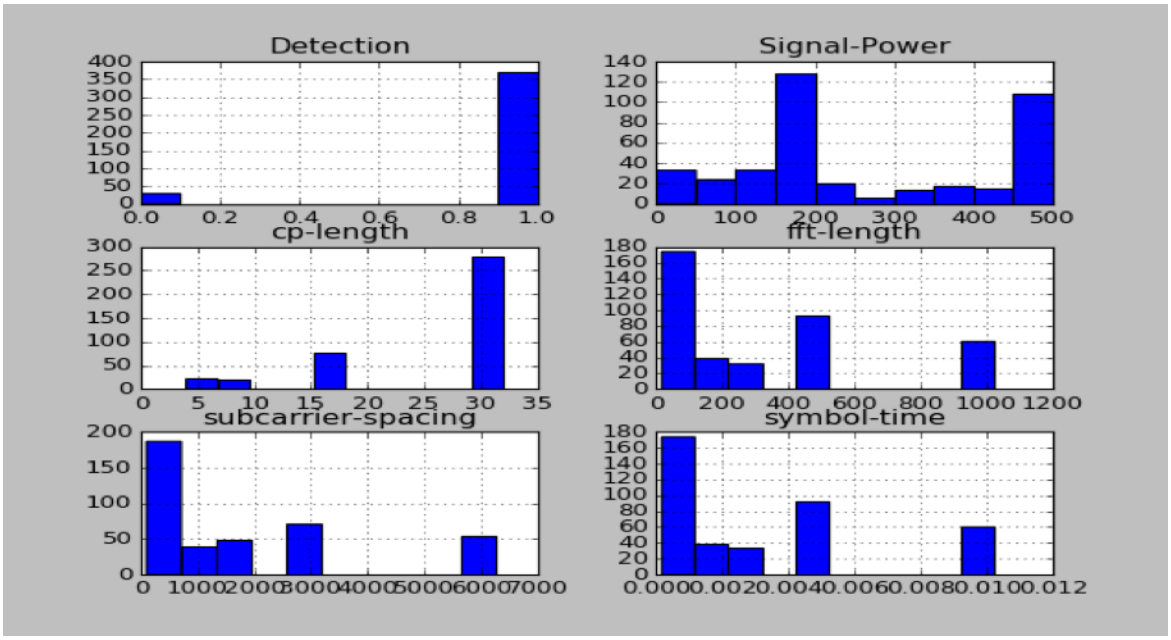


Fig. 24: Histogram plot of the features

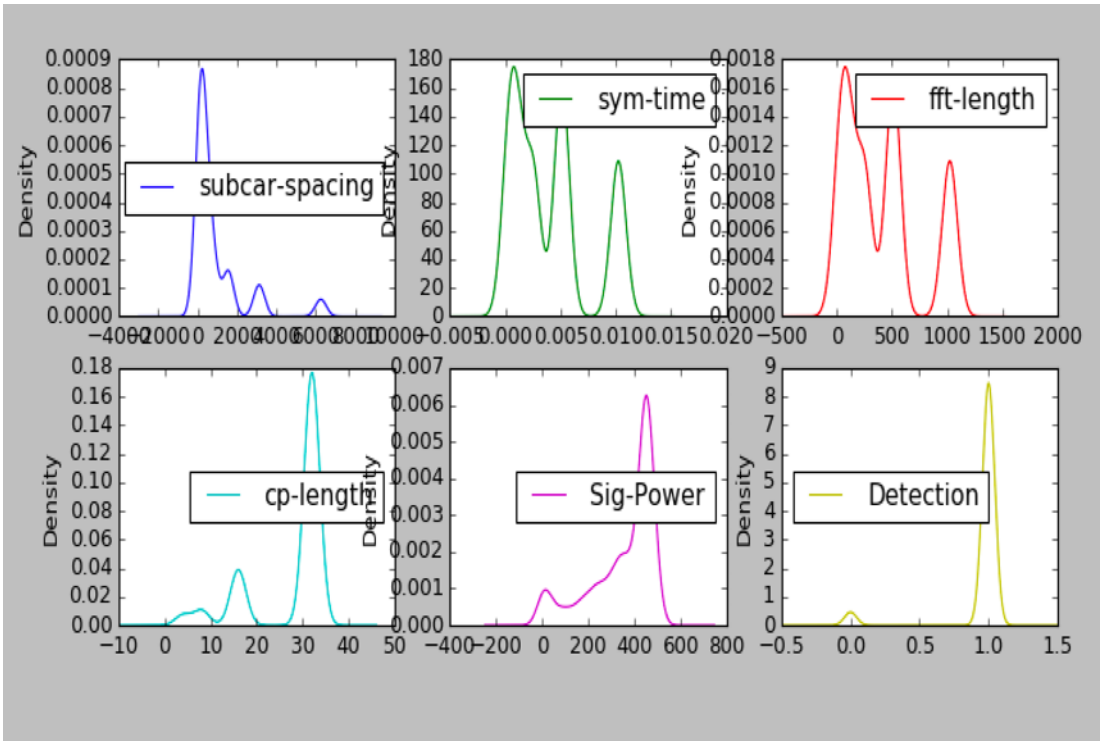


Fig. 25: Density distribution of the features

CHAPTER 5

DISCUSSION

5.1 OVERVIEW OF FINDINGS

From the experiment and results we can conclude that drone signal from different class or source can be detected and classified by its radio frequency parameters using machine learning algorithms. This work is still in its early stage, and we hope with more experiment and testing we can improve the overall classification accuracy. GNU Radio Companion was used towards the successful implementation of the drone detection and classification toolbox. We decided to use matlab simulations to validate some of the results and observations of the RF toolbox. Below are some of the initial findings.

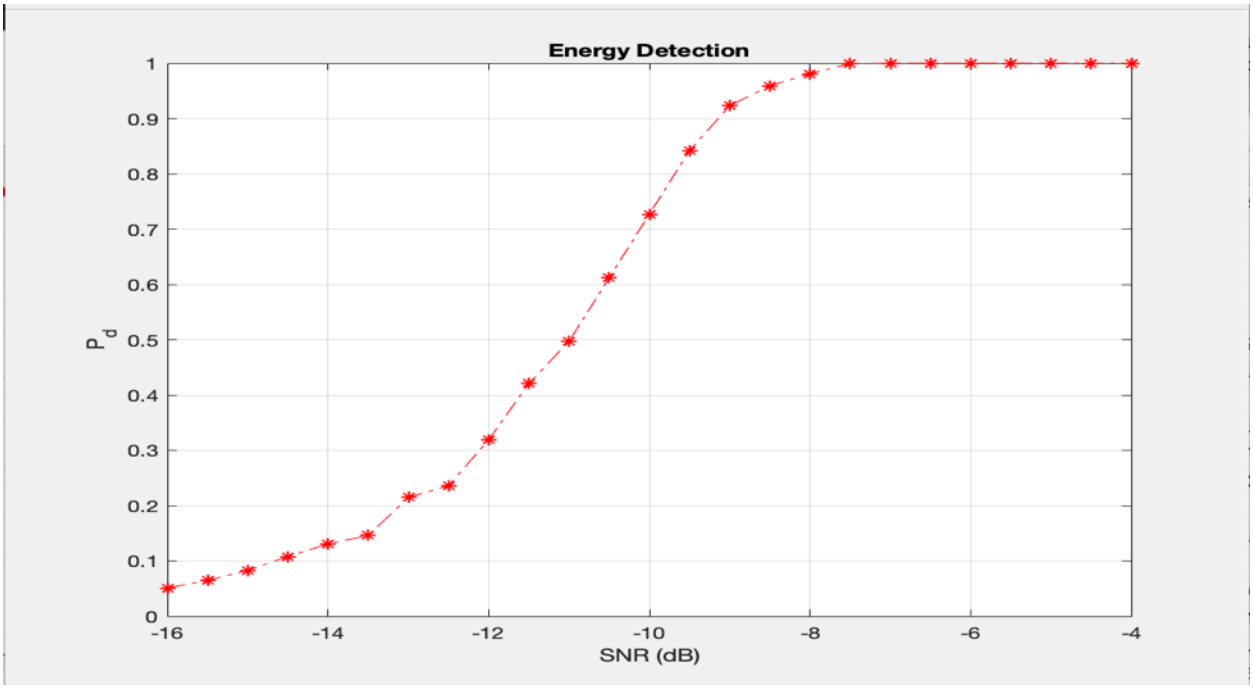


Fig. 26: Relationship between Detection probability and SNR

There is a correlation between SNR and Probability of detection. Detection probability will decrease as the SNR level decreases. In a scenerio where the drone is

very close to the receiver, the SNR will be high and this correlates with the results which shows that increase in distance will result in a low detection rate.

5.2 RESEARCH LIMITATIONS

In wireless communications there are different factors that affect the propagation of radio signals from source to destination. In many cases this is from the antenna transmitting the signal to the receiving antenna. As such for signals to be successfully transmitted and received from one distance to another some factors play an important role in the design. A major component is the pathloss, which can be defined as the reduction in the power levels of electromagnetic waves as it propagates through space. Simulation for some different pathloss models that can be associated with wireless signal propagation was done using the matlab tool. Below are some observations.

simulation for some different pathloss model that can be associated with wireless signal propagation was done using matlab tool.

5.2.1 FREE SPACE PROPAGATION MODEL

Free space propagation model assume the transmitter and receiver are located in an isolated environment and as such effect of reflection or absorption from obstacles are not considered in pathloss estimation.

$$PL = 20 * log_{10}(d) + 20 * log_{10}(f) + 20 * log_{10}((4 * pi)/c);$$

Where the distance d is 1-100 m, frequency f =2.4Ghz and c is the lightspeed constant. Ideally the model is used mostly in cases where the distance in the prediction are in kilometers. But for our experiment the distance is less than 100 meters.

5.2.2 TWO - RAY PROPAGATION MODEL

2-ray model uses the assumption of one line of sight path and one ground or refected wave to estimate path loss

$$PL = 40 * log_{10}(d) - (10 * log_{10}(gt) + 10 * log_{10}(gr) + 20 * log_{10}(ht) + 20 * log_{10}(hr))$$

Where the distance d is 1-100 m,gt and gr is the gain of the antenna , ht is transmit antenna height and hr is the receive antenna height.

5.2.3 EMPIRICAL PROPAGATION MODEL

In literature some authors have explored and used empirical formular to derive pathloss model for drones, cellular and wifi technology. [27] proposes a channel model that is based on the distance d and altitude θ of the drone. The formular is :

$$PL_{drone}(d,\theta) = L_0 + 10nlog_{10}\frac{d}{d_{ref}} + A(\theta - \theta_0)exp\frac{(-\theta - \theta_0)}{B} + N(\mu,\sigma)$$

where L_0 is the initial path loss and n is expressed by the scalar value of the path loss. The distance is 0 - 70 m, and the reference distance is 5 m. A is the scalar value of the path loss, and the slope is $0° - 55°$. θ_0 is the angle offset, which is the change in the loss with the angle. B is the scalar value of the altitude angle. N represents a normal probability distribution, is the mean value, and is the variance.

Below is the pathloss plot for the 3 propagation model considered

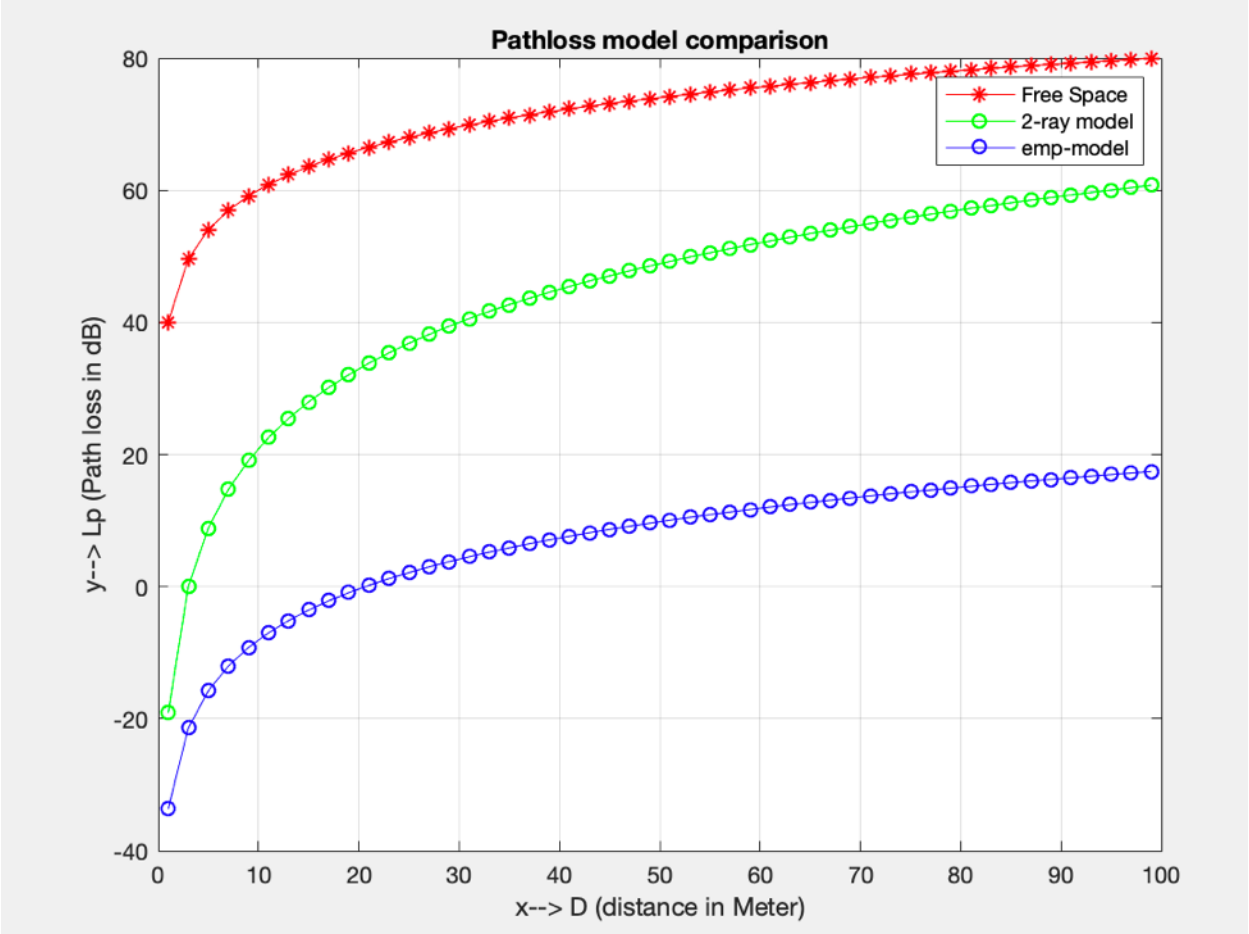


Fig. 27: 3 pathloss model comparison

In general it can be observed that the increase in distance will result in weakness of the signal strength due to the effect of pathloss.

CHAPTER 6

CONCLUSION

6.1 CONCLUSION

From the experiment and results shown so far, we can successfully classify signals that are being transmitted from DJI phantom and Mavic drones. Using the machine learning training and testing we observed that out of the different classification algorithms, KNN works best and provide the highest accuracy. Our goal is to be able to improve this accuracy by identifying new features from the test statistics that can be added to the classification model. So far, we have been able to test the addition of signal standard deviation and median and even though accuracy increased a little we think there is still more to be done in this regard. Real time validation of RF based signal classification using the trained model created from the machine learning algorithm so far has a prediction accuracy which is not very high at this point. And as stated earlier, we hope to change this and intend to continue to carry out more training, testing and analysis so as to increase the prediction of the classification. Different approach will be employed in collecting and testing the data.

REFERENCES

[1] Investigating cost effective RF-based drone detection of drones by phuc nguyen,Mahesh ravindranathan,Anh nguyen,Richard han and Tam Vu

[2] Drone detection based on an audi-assisted camera array by Hau Liu,Zhiqiang wei , yitong chen,Jie Pan , Le lin and Yufang Ren

[3] Passive radar system for detecting UAV Based on the OFDM communication signal by Xiaoqi yang,Kai huo.Weidong jiang, Jingjing Zhao and Zhaokun Qiu

[4] Night-time detection of UAVs using thermal infrared camera by Petar Andrasi, Tomislav Radisic,Mario mustra and Jurica Ivosevic

[5] Concept for a Software Defined Radio Based System for Detection, Classification and Analysis of Radio Signals from Civilian Unmanned Aerial Systems by Stefan Kunze, Alexander Weinberger and Rainer Poeschl

[6] Research into the Radio Interference Risks of Drones by Ir. Gerton de Goeij (Strict), ir. Eildert H. van Dijken (Strict) and ir. Frank Brouwer (FIGO)

[7] Detecting drones using machine learning by Waylon D. Scheller

[8] http://www.sdrforum.org/pages/documentLibrary/documents/

[9] Energy-Detection Based Spectrum Sensing for Cognitive Radio on a Real-Time SDR Platform by McBath John Rwodzi

[10] Framework for Automatic Signal Classification Techniques (FACT) for Software Defined Radios by Jithin Jagannath, Hanne M. Saarinen and Andrew L. Drozd

[11] http://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/wlan-ofdm/content/ofdm-basicprinciplesoverview.htm

[12] Ofdm Signal Classification and Synchronization by Ying Wang, Sujit Nair, Alex Young, Qinqin Chen and Charles W. Bostian

[13] https://calhoun.nps.edu/bitstream/handle/10945/4620/09Sep-Schnur.pdf?sequence=1isAllowed=y

[14] Recognition of Digital Modulated Signals based on Statistical Parameters by Khandker Nadya Haq , Ali Mansour, Sven Nordholm ,

[15] Modulation Classification of MIMO-OFDM Signals by Independent Component Analysis and Support Vector Machines by Yu Liu, A.M. Haimovich, Fellow, IEEE, Wei Su, Jason Dabin, Emmanuel Kanterakis,

[16] url : http://www.etti.unibw.de/labalive/experiment/wifi/

[17] Detection and Classification of OFDM Waveforms Using Cepstral Analysis by J. Jntti and S. Chaudhari and V. Koivunen,

[18] Ofdm Modulation Classification and Parameters Extraction by Hong Li, Yeheskel Bar-Ness, Ali Abdi, Oren S. Somekh, and Wei Suy,

[19] Ofdm Signal Type Recognition and Adaptability Effects in Cognitive Radio Networks by Anna Vizziello, Ian F. Akyildiz, Ramon Agusti, Lorenzo Favalli, Pietro Savazzi,

[20] Cyclic Prefix Length Determination for Orthogonal Frequency Division Multiplexing System over Different Wireless Channel Models Based on the Maximum Excess Delay Spread by Amar Al-Jzari and Kostanic Iviva

[21] A Signal Detector for Cognitive Radio System by Aldo Buccardo

[22] Python Documentation [online]. url: https://www.python.org/about/

[23] Numpy Documentation [online]. url: http://www.numpy.org/

[24] Pandas Documentation [online]. url :http://pandas.pydata.org/

[25] Fabian Pedregosa. Scikit-learn: Machine Learning in Python [online]. url: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[26] Machine learning final thesis by Abhishek Acharya

[27] Empirical pathloss modelling and RF detection scheme for various drones by Won-ho Jeong, Hong-rak Choi and Kyung-seok Kun

APPENDIX A

CODES FOR PARAMETER ESTIMATION AND CLASSIFICATION

```
/* -*- c++ -*- */
/*
 * Copyright 2018 gr-ofdm_param_estim author.
 *
 * This is free software; you can redistribute it and/or
    modify
 * it under the terms of the GNU General Public License as
    published by
 * the Free Software Foundation; either version 3, or (at
    your option)
 * any later version.
 *
 * This software is distributed in the hope that it will be
    useful,
 * but WITHOUT ANY WARRANTY; without even the implied
    warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
    the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public
    License
 * along with this software; see the file COPYING.  If not,
    write to
 * the Free Software Foundation, Inc., 51 Franklin Street,
 * Boston, MA 02110-1301, USA.
 */
```

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <gnuradio/io_signature.h>
#include "ofdm_param_estimation_c_impl.h"
#include <complex>
#include <volk/volk.h>

namespace gr {
  namespace ofdm_param_estim {

    ofdm_param_estimation_c::sptr
    ofdm_param_estimation_c::make(double samp_rate, int
       signal, int min_items, const std::vector<int> &typ_len
       , const std::vector<int> &typ_cp)
    {
      return gnuradio::get_initial_sptr
        (new ofdm_param_estimation_c_impl(samp_rate, signal,
           min_items, typ_len, typ_cp));
    }

    /*
     * The private constructor
     */
    ofdm_param_estimation_c_impl::
       ofdm_param_estimation_c_impl(double samp_rate, int
       signal, int min_items, const std::vector<int> &typ_len
       , const std::vector<int> &typ_cp)
      : gr::sync_block("ofdm_param_estimation_c",
              gr::io_signature::make(1, 1, sizeof(gr_complex)
                 ),
```

```
            gr::io_signature::make(0, 0, 0)) //no output
               stream
            //gr::io_signature::make(1, 1, sizeof(float)))
{
  //read the argument list into local variables
    d_samp_rate = samp_rate;
    d_signal = signal;
    d_min_items = min_items;
    d_typ_len = typ_len;
    d_typ_cp = typ_cp;
    d_fft = new fft::fft_complex(1024, true);
    //define a single output message port
    message_port_register_out(pmt::string_to_symbol("
       ofdm_out")); //message passing from output
    //// Set up input message ports:
    message_port_register_in(pmt::mp("in_ED")); //
       message input from Energy Detector
    set_msg_handler(pmt::mp("in_ED"), boost::bind(&
       ofdm_param_estimation_c_impl::message_handler_ED,
       this, _1));
}

/*
 * Our virtual destructor.
 */
ofdm_param_estimation_c_impl::~
   ofdm_param_estimation_c_impl()
{
    delete d_fft;
}

//Input message handler
void ofdm_param_estimation_c_impl::message_handler_ED(pmt
   ::pmt_t ed_msg)
```

```
    {
        if (pmt::is_tuple(ed_msg)) {
            if (pmt::length(ed_msg) != 2) {
                GR_LOG_ALERT(d_logger, boost::format("Error
                    while unpacking command PMT: %s") %
                    ed_msg);
                return;
            }
            d_signalAvg = pmt::to_double(pmt::tuple_ref(
                ed_msg, 0));
            d_Threshold = pmt::to_double(pmt::tuple_ref(
                ed_msg, 1));
        }
    }

// Normalized autocorrelation function of given vector
std::vector<float>
ofdm_param_estimation_c_impl::autocorr(const gr_complex *
   in, int len) {
  std::vector<float> acf;
  if(len == 0) {
    return acf;
  }
  __GR_VLA(gr_complex, Rxx, len); //variable length array
      to hold intermediate calculation
  gr_complex acf_temp;
  for(unsigned int k = 0; k < d_typ_len.back(); k++) {
    acf_temp = 0;
    volk_32fc_x2_multiply_conjugate_32fc(Rxx, in, &in[k],
        len-k); // save (x * conj(x)) to Rxx output
    // summation of all Rxx array elements
    for(unsigned int i = 0; i < len-k; i++) {
      acf_temp += Rxx[i];
    }
```

```
    acf.push_back(std::abs(acf_temp/gr_complex((len-k),0)
        ));//push the normalized result to acf
  }

  return acf;
}

    //Setting the sample rate function
void ofdm_param_estimation_c_impl::set_samp_rate(double
   d_samp_rate) {
  ofdm_param_estimation_c_impl::d_samp_rate = d_samp_rate
      ;
}

// calculate time variant autocorrelation for fixed shift
gr_complex*
ofdm_param_estimation_c_impl::tv_autocorr(const
   gr_complex *in, int len, int shift) {
  __GR_VLA(gr_complex, corr_temp, len);//Define a
      variable length array
  gr_complex *Rxx = (gr_complex*)volk_malloc(len*sizeof(
      gr_complex), volk_get_alignment());
  gr_complex R = gr_complex(0,0);
  volk_32fc_x2_multiply_conjugate_32fc(corr_temp, in, &in
      [shift], len);
  int k = 0;
  // begin at back and summarize up to front
  for(int i = len-1; i >= 0; i--) {
    R *= k;
    R += corr_temp[i];
    R *= 1.0/(k+1.0);
    Rxx[k] = R;
    k++;
  }
```

```
  return Rxx;
}

// round value to nearest list entry
int
ofdm_param_estimation_c_impl::round_to_list(int val, std
    ::vector<int> *list) {
  int result = -1;
  int diff = 99999; // "high value"
  for(unsigned int i = 0; i < list->size(); i++) {
    if(std::abs(list->at(i) - val) < diff) {
      diff = std::abs(list->at(i) - val);
      result = list->at(i);
    }
  }
  return result;
}

void
ofdm_param_estimation_c_impl::resize_fft(int size) {
  delete d_fft;
  d_fft = new fft::fft_complex(size, true);
}

// GUI message
pmt::pmt_t
ofdm_param_estimation_c_impl::pack_message(float subc,
    float time, int fft, int cp, double sig_avg, double
    threshold) {
  //pmt::pmt_t identifier = pmt::make_tuple(pmt::
      string_to_symbol("Signal"), pmt::from_uint64(
      d_signal));
  //pmt::pmt_t ofdm_info = pmt::make_tuple(pmt::
      string_to_symbol("OFDM"), pmt::from_float(1));
```

```
  pmt::pmt_t subcarr = pmt::make_tuple(pmt::
      string_to_symbol("Subc. space"), pmt::from_float(
      subc));
  pmt::pmt_t symtime = pmt::make_tuple(pmt::
      string_to_symbol("Sym time"), pmt::from_float(time))
      ;
  pmt::pmt_t fftsize = pmt::make_tuple(pmt::
      string_to_symbol("Subcarriers"), pmt::from_float(fft
      ));
  pmt::pmt_t cyclpre = pmt::make_tuple(pmt::
      string_to_symbol("CP len"), pmt::from_float(cp));

  pmt::pmt_t sigAvg = pmt::make_tuple(pmt::
      string_to_symbol("Signal Average"), pmt::from_double
      (sig_avg));
  pmt::pmt_t thres = pmt::make_tuple(pmt::
      string_to_symbol("Threshold"), pmt::from_double(
      threshold));
  //pmt::pmt_t msg = pmt::make_tuple(identifier,
      ofdm_info, subcarr, symtime, fftsize, cyclpre);
  pmt::pmt_t msg = pmt::make_tuple(subcarr, symtime,
      fftsize, cyclpre, sigAvg, thres);
  return msg;
}


int
ofdm_param_estimation_c_impl::work(int noutput_items,
    gr_vector_const_void_star &input_items,
    gr_vector_void_star &output_items)
{
  const gr_complex *in = (const gr_complex *) input_items
      [0];
```

```
//FILE *MLData = fopen("/home/cssdr/Datasets/
   MLTraining.dat", "a+");

 //for this block we don't need any output stream
 //<+OTYPE+> *out = (<+OTYPE+> *) output_items[0];
 //const float *out = (const float *) output_items[0];

 if(noutput_items <= d_min_items) {
  return 0;
 }

 //Estimate FFT length
 //argmax(sum(x * conj(x)))

 //Calculate autocorrelation function of input vector
 std::vector<float> acf = autocorr(in, noutput_items);
 //Find argmax
 int fft_len = std::distance(acf.begin(),
   std::max_element(acf.begin()+d_typ_len.front(), acf
      .end()));
 fft_len = round_to_list(fft_len, &d_typ_len); //
    round to possible values

 // calculate time variant autocorr and cyclic
    correlation function
 // and estimate CP length
 int shifted_fft_len = noutput_items-fft_len; //
    length of all following vectors because of shift
    fft_len
 resize_fft(shifted_fft_len); // resize FFT

 gr_complex *Rxx;
 Rxx = tv_autocorr(in, shifted_fft_len, fft_len); //
    calc time varaint autocorr
```

```
  // FFT to get CCF (cyclic correlation function)
memcpy(d_fft->get_inbuf(), Rxx, sizeof(gr_complex)*
    shifted_fft_len);
d_fft->execute();
volk_free(Rxx);
__GR_VLA(float, result, shifted_fft_len); // magnitude
    of CCF
volk_32fc_magnitude_32f(result, d_fft->get_outbuf(),
    shifted_fft_len);

// fftshift
d_tmpbuflen = static_cast<unsigned int>(std::floor((
    shifted_fft_len) / 2.0));
__GR_VLA(float, d_tmpbuf, shifted_fft_len / 2);
memcpy(d_tmpbuf, &result[0], sizeof(float) * (
    d_tmpbuflen + 1));
memcpy(&result[0], &result[shifted_fft_len -
    d_tmpbuflen],
        sizeof(float) * (d_tmpbuflen));
memcpy(&result[d_tmpbuflen], d_tmpbuf,
        sizeof(float) * (d_tmpbuflen + 1));

// only use positive frequencies
std::vector<float> Cxx(result+(shifted_fft_len)/2,
    result+shifted_fft_len);
// search for peak in possible area
long cp_len = std::distance(Cxx.begin(), std::
    max_element(
          Cxx.begin()+(int)(shifted_fft_len/(fft_len+
              d_typ_cp.back())),
          Cxx.begin()+(int)(shifted_fft_len/(fft_len+
              d_typ_cp.front()))));
```

```
        cp_len = shifted_fft_len/cp_len; // convert peak to
            length value
        cp_len = cp_len-fft_len; // calculate CP len from total
             len
        cp_len = round_to_list(cp_len, &d_typ_cp); // round to
            possible value


        // calculate subcarr spacing and symbol time and pub
            message
        float subspc = d_samp_rate/fft_len;
        float symtime = 1/subspc;
        pmt::pmt_t msg = pack_message(subspc, symtime, fft_len,
             cp_len, d_signalAvg, d_Threshold);
        message_port_pub(pmt::intern("ofdm_out"), msg);

        //message printing
        //printf("%f, %f, %d, %d ,%f , %f",subspc, symtime,
            fft_len, cp_len,d_signalAvg, d_Threshold );
        //printf("\n");
        //printf("%f, %f, %d, %ld",subspc, symtime, fft_len,
            cp_len);
        //printf("%f, %f, %f, %f",subspc, symtime, float(
            fft_len), float(cp_len));
        //printf("\n");


        // Tell runtime system how many output items we
            produced.
        return noutput_items;
      }

  } /* namespace ofdm_param_estim */
} /* namespace gr */
```

APPENDIX B

CODES FOR ENERGY DETECTION

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# Copyright 2018 gr-EnergyDetector author.
#
# This is free software; you can redistribute it and/or
    modify
# it under the terms of the GNU General Public License as
    published by
# the Free Software Foundation; either version 3, or (at your
     option)
# any later version.
#
# This software is distributed in the hope that it will be
    useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty
     of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
    the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
    License
# along with this software; see the file COPYING.  If not,
    write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
#
```

```
import scipy
import scipy.special as scs
import numpy
from gnuradio import gr
import pmt

class Energy_Detector_ff(gr.decim_block):
    """
    docstring for block Energy_Detector_ff
    """
    def __init__(self, samples, Pfa):
        self.samples = samples
        gr.decim_block.__init__(self,
            name="Energy_Detector_ff",
            in_sig=[numpy.float32, numpy.float32],
            out_sig=[numpy.float32], decim = self.samples)
        self.Pfa = Pfa
        self.message_port_register_out(pmt.intern("ED_out"))


    def work(self, input_items, output_items):
        in0 = input_items[0]
        in1 = input_items[1]
        out = output_items[0]
        #<+signal processing here+>
        Avg = scipy.mean(in0)
        signalAvg = round(Avg,8)
        NoisePower = in1**2
        NoiseAvg = scipy.mean(NoisePower)
        var = scipy.var(NoisePower)
        stdev = scipy.sqrt(var)
        Qinv = scipy.sqrt(2) * scs.erfinv(1 - 2*self.Pfa)
        Threshold = round((NoiseAvg + Qinv*stdev),8)
        if signalAvg > Threshold:
```

```
            out[:] = signalAvg
            #print("signal is present", signalAvg)
            detection = 1
    else:
            out[:] = Threshold
            detection = 0

    #print("signal is absent", signalAvg)

    msg = pmt.make_tuple(pmt.from_double(signalAvg), pmt.
        from_double(detection))
    self.message_port_pub(pmt.intern("ED_out"), msg);

    return len(output_items[0])
```

VITA

Bello Abdulkabir
Department of Electrical and Computer Engineering
Old Dominion University
Norfolk, VA 23529

www.linkedin.com/in/abdulkabirbello

PROFESSIONAL SUMMARY

A network and systems engineer with 3 years' experience as a radio frequency planning and optimization of cellular and wireless communication network technologies e.g. 2G/3G/4G/LTE/Wi-Fi. Over a year Research experience working with C++/python, gnu radio, software defined radios, cyber systems security, design and engineering. A reliable team player with excellent communication, management and leadership skills.

EDUCATION

MASTER OF SCIENCE | ELECTRICAL AND COMPUTER ENGINEERING **May 2019**

Old Dominion University (ODU) Norfolk, Virginia.

BACHELOR OF SCIENCE | COMPUTER ENGINEERING **May 2012**

Kharkov National University of Radio Electronics, Ukraine.

TECHNICAL SKILLS

Networking: Cisco/Huawei iOS, WLAN, LAN, TCP/IP, IPv4, IPv6, SMTP, DHCP, DNS, Routing and switching Technologies, UDP

Virtualization: VMware, Kernel based Virtual machine (KVM)

Wireless: Google earth, TEMS, MapInfo, GNU radio, Actix, FPGA, Arduino microcontrollers

Languages: C/C++, java, python, Matlab, HTML/HTML5, VHDL

Platform Administration: ubuntu, Unix/Linux, windows

Network Security: Wireshark, IPSec, Netwox, Netlogo, Blockchain, HackRF, HPC python scripting

COURSES

- Numerical methods for engineers
- Network system security, engineering & design
- -Wireless communication
- Cyber system engineering
- Computer Networks and architecture
- Object oriented programming
- Udemy CompTIA network + certificate course

WORK EXPERIENCE

Graduate Research Assistant | Virginia Modelling and Simulation Center-ODU **Feb 2018-Present**

- Developed a RF communications intrusion detection module for UAV (DJI phantom drone) signals in GNU Radio using python and C++.
- Implementation of a training and testing model using machine learning algorithm on python for RF signal automatic modulation classification.
- Testing UAV (DJI Phantom Drones) vulnerabilities through penetrations and threats analysis.
- Daily usage of software defined radio, hackRF and USRP to implement RF digital signal processing and analysis
- Modelling and simulation of RF signal environment using GNU Radio and Matlab

RF Planning and Optimization Engineer | Huawei Technologies Co. LTD. **Oct 2014 – Jul 2017**

- Planned new 2G/3G/4G sites for roll out, swap and expansion project and alarm clearing on cells/sites.
- Team lead for Globacom 2G/3G/LTE drive test, network bench marking, single site verification on new telecommunication site and physical optimization of RF antenna.
- Monitored 2G/3G/4G key performance indicator (KPI) and perform analysis of worst cells and fine tune parameters to improve Pilot Strength, RSCP, Ec/Io and FER
- Carried out network capacity and coverage planning, optimization of air interface and radio access, interference and RF performance optimization at different venues (stadium, airports, shopping mall).
- Responsible for NodeB, ENodeB commissioning and decommissioning management, LTE Frequency planning, scanning and analysis. detected interference source and resolved issue before deployment of sites.
- Monitors system performance indicators such as dropped calls, blocked calls, origination failures, RRC and RAB congestion, handoff failures, review capacity reports and forecast for short- and long-term site/system requirements.
- Performed detailed analysis of network performance metrics and drive test data to identify trends and anomalies in network performance.

Application Support Intern | Kaduna refining and petrochemical company **Nov 2013 – Jun 2014**

- Monitoring network to ensure network availability to all system users.
- Performed routine network start up and shutdown procedures and maintain control records
- Assist network team in performing basic network trouble shooting and issues resolution.
- Assisted in managing cases, requests and enhancements using online tools and processes
- Routine deployment of service releases, patches, requests on software applications

PROJECTS

IHS Tower Consolidation | Globacom Swap and Expansion of LTE sites | Huawei

- Planned the consolidation of MTN, AIRTEL and ETISALAT sites on a single tower without creating coverage holes.
- Planned over 3000 new 2G/3G and LTE telecommunication site

UAV Intrusion detection system (RF Classification) using Software Defined Radio | ODU

- Developing an intrusion detection module that can detect rogue drones' RF signal, classify the signal and counter the threat it poses.

Audio Spectrum Analyzer | KHNURE

- Used Arduino microcontroller hardware and software programming to develop an audio speech and sound analyzer

Linux Firewall attacks | DNS and Web Security | ODU

- Performed various network security vulnerability attacks on windows XP,7, Ubuntu Linux and Metasploit.

EXTRACURRICULAR ACTIVITIES

- International student association board (ISAB), NSBE and IEEE member