# A picture containing drawing Description automatically generated

# Final Report

## Team 04 – Dynamite

## 4/13/2020

## Team Members

Gerald Bankes

Yanda Cheng

Lauren Endicott

Aaron Smith

## Class Advisors

Dr. -Ing. Regina Hannemann

Dr. Daniel Lau

Dr. John Young

## Sponsors

Johnathan Andrews

Tommy Hogue

## Table of Contents

## List of Figures

## List of Tables

## Abstract

Modern devices can communicate with each other wirelessly by using radio waves as an information carrier. With many devices communicating simultaneously, any given area can have many different radio signals passing through it. Dynetics has tasked us with continuing a project that was given to an ECE senior design team last year: to build a portable device that can detect signals in the area and provide the user with important information about the signal. Using the existing hardware that we may expand upon or improve; we will code and implement software to analyze and categorize radio frequency signals. This document describes these requirements, the design, our project plan, and the functional decomposition for each of the four areas of focus that were identified: digital signal processing, software defined radio software, classification algorithm, and hardware optimization.

## Problem Statement

### Need

At any point in space, at any time, there are many signals being broadcast over a broad range of frequencies. Wi-Fi, Bluetooth, baby monitors, microwaves, and other devices operate at frequencies near 2.4 GHz. These signals may interfere and distort radio communication. Each of these signals serve a different purpose depending on certain characteristics of the signal such as the bandwidth, carrier frequency, and modulation type. In order to determine the purpose a specific signal serves, each of these characteristics must be known.

### Objective

The objective of this project is to design a useful tool for attempting to identify why a device connection may not be reliable. Our device shall characterize signals as Wi-Fi, Bluetooth, or “other” signals in close proximity. The signals will be in the range of 2.4 – 2.472 GHz. Our device places emphasis on portability and safety. The device shall run the algorithms that have been designed, classify the signals, and save the data to an external folder in a timely manner for future use.

### Background

The team that worked the Dynetics last year implemented an RF sensor with microcontroller modules, software defined radio systems, and an antenna receiver circuit. The team used an SDR and Intel NUC mini PC configuration to collect signals. All software on the device was later removed. The major emphasis in building their device was mobility, as such it is battery powered and small in size. Our team will reassemble the hardware and add functionality in the form of a characterization algorithm and user- friendly GUI.

Currently, spectrum analyzers are the main tool used for spectrum analysis, sweeping over each frequency in its range and displaying the magnitude of the signal at that frequency [1]. From the data is gathers, it can also display the signal power, bandwidth, and more. These devices, however, typically do not provide the front-end hardware necessary to capture radio frequencies natively and are not capable of making predictions on the likely information content of the signal.

A device like our intended system was created by Michael Ossmann and Dominic Spill called “What’s on the Wireless.” The goal of the project was the same as ours: to automatically identify captured radio frequency signals. Their implementation uses software defined radio and software that they created to make signal analysis easier. From the sole pieces of documentation that we were able to find, the implementation was not packaged as a single unit as the components were uncovered and a computer was not provided, the user interface would need high technical knowledge to use, and the software did not make a prediction for the likely content of the signal [2].

Another device like our system was made by General Dynamics for military radio monitoring called SignalEye. This implementation is more closely related to our project in that it reports much of the same information that we intend to report and uses neural networks to predict the threat of a given signal.

However, as it is marketed specifically as a military technology, the criteria and prediction that the device ultimately makes is likely much different than ours, as we only want to predict signal content and not threat level. The implementation is also software only; front-end radio hardware and computer system are not provided [3].

## Requirements Specification

### Marketing Requirements

The final goal of this project is to take the existing portable device and give it the ability to analyze signals and display the characteristics desired by Dynetics. The system will satisfy the following marketing requirements:

### Objective Tree

A screenshot of a cell phone

Description automatically generated

Each category and subcategory in the objective tree have a certain weight to signify how important that category/subcategory is to the team. These weights were determined by setting up a comparison matrix of the categories at each level and assigning them a level of “importance” relative to another, then taking the geometric mean of these comparisons and finding the percentage each mean has out of the sum of the means [4]. The comparison matrices can be found in the Appendix section.

### Engineering Requirements

Each marketing requirement from section 3.1 must correspond to at least one engineering requirement. Table 3.3.1 contains these engineering requirements, their associated marketing requirement(s), a justification, and a verification method.

## Design Impacts

### Health

There is an expectation with every product that the user should be safe when using it. As with any device that is electrically powered, there is the possibility that the user may get shocked if the power supply of the device is not properly secured or wiring is damaged. Since our device is battery powered, and can also be powered from an outlet, we have to be sure to conceal our wiring and battery pack, so the user does not physically come into contact with them.

### Social

At any time, there are many texts being sent and phone calls being made all around you. Each person that sends one of these texts or makes one of these calls has a reasonable expectation to privacy their information will not be received by anyone other than who they intended it for. With devices similar to ours, other spectrum analyzers, we would be able to intercept some of these signals. We are therefore restricted by these social standards as to what information we should be extracting from these received signals.

### Economic

Each project comes with a budget that must be followed. Dynetics has provided our group with a $2000 budget that we must stick to in the development of our device. All of the previous team’s hardware has been provided to us so there isn’t much that we will need to purchase, but there is the possibility that we will have to upgrade some of the previous team’s equipment to meet new requirements that we have set, such as a new antenna capable of capturing signals in the FM radio frequency band.

### Manufacturability

For a product to be able to be produced quickly and in bulk, it must be easily manufactured. Our device consists of mostly low-cost or easily produced materials, with the exception of the mini PC and the SDR, so it is easily manufactured in that sense. In order to help with manufacturability, we should also keep the device lightweight and small.

### Standards

The Federal Communications Commission (FCC) regulate any electronic products which communicate over radio frequency (RF) signal. These devices must be tested to demonstrate compliance to the FCC rules for all electrical function within the device. Any RF device must be approved by authorization procedures before marketed, imported, or used in the United States. The device will be used in various areas, so it has to meet the standards of FCC Part 15 regulation of electromagnetic interference [5].

## Functional Decomposition

Our functional decomposition is split into two sections: software and hardware. Each stage of decomposition starts with a look at the system with its highest level of abstraction. Each level is broken down into progressively more detail from there in order to make the planned implementation and design of our project clearer.

### Hardware Decomposition

#### Level 0 – System Overview

A screenshot of a cell phone

Description automatically generated

Figure 5.1.1.1 shows the highest functional level of our device. As specified in Table 5.1.1.1, the device will receive input from the 9V DC power source, a user input in the form of a range of frequencies to filter received signals, and radio frequency signals in the surrounding area. From these inputs, the device will calculate signal characteristics like the bandwidth and center frequency and give a prediction for the origin of the signal (FM radio, Wi-Fi, etc.).

#### Level 1 – CPU and Touch Screen Interface

A screenshot of a cell phone

Description automatically generated

Figure 5.1.2.1 shows a more in-depth look at our system’s hardware. Since all of the hardware was inherited from the previous team, there weren’t many design choices to be made here. The dotted section of the figure contains all of the pieces of the hardware that we will not be changing. This leaves us with two pieces that we will be working with: the CPU and the touch screen.

The CPU will receive power from the power supply, and also provide power to the touch screen, external storage device, and SDR. From the SDR, the CPU will receive a sample of all the signals in the area, within the frequency range specified by the user. This frequency range will come from the user, through the interaction with the touch screen. Once all of these inputs are received by the CPU, it will begin to process the signals according to the software we will outline in section 5.2. After the signal has been processed, a copy of the frequency display of the signal and its relevant metadata will be sent to the touch screen for the user to see, and the external storage device for later analysis. This series of inputs and outputs are defined in Table 5.1.2.1.1.

The touch screen will be powered from the CPU and will receive the user input. This input will be passed to the CPU to define its frequency range to search for signals in. Once the CPU finishes processing the signals, the touch screen will then receive the frequency display and signal metadata to display for the user. This process as well as the inputs and outputs are defined in Table 5.1.2.1.2.

### Software Decomposition

#### Level 0 – Software Overview

A close up of a logo

Description automatically generated

Figure 5.2.1.1 shows the most basic outline of our code design. Our software will receive the signal sample from the SDR, run the data through the algorithms we design, and display the signal metadata, classification, and frequency spectrum to the touch screen. The specifics of the signal processing software are explained more beginning in section 5.2.2.

#### Level 1

##### Level 1 – General Structure

A close up of a piece of paper

Description automatically generated

Figure 5.2.2.1 shows a deeper look into the software we are designing to process the captured signals. First, the buffer of samples we receive from the SDR will be sent through a Fast Fourier Transform (FFT) to produce the frequency spectrum of the signals. Once the FFT is performed, it will be passed into a band pass filter to limit the frequency range to only the 2.4-2.5GHz range. This filtered signal will then be sent to the bandwidth extraction algorithm and the modulation type extraction algorithm

The filtered signal will also be sent into the modulation extraction algorithm in order to determine the modulation type of the signal to be sent to the classification algorithm. It will also be sent to the bandwidth extraction algorithms to determine the bandwidth and center frequency of each signal in the frequency range.

All of these outputs will then be sent to two places: straight to the screen to be displayed for the user and to the classification algorithm to determine the origin of the signal. Once the classification algorithm determines the origin of the signal, it will also display it on the screen for the user to see.

##### Level 1 – Graphical User Interface

A screenshot of a social media post

Description automatically generated

The graphical user interface (GUI) allows the user to run the necessary algorithms and view the results of the system. The GUI displays the frequency spectrum of the captured signals and a table of the different signals with their corresponding calculated frequencies, classes, modulation schemes, bandwidths, and power. The GUI has been divided into four tabs providing views of the complete spectrum, only Wi-Fi signals, only Bluetooth signals, and only “Other” signals. The interface allows the user to run or stop the algorithm ant any time with the use of the “Run” and “Pause” buttons. This is to provide the user with flexibility of the timing of the run of the system for whenever the user is ready. The menu bar at the top of the interface allows the user additional functionality which is yet to be implemented. Currently, the “File” menu allows the user to save the data of the current run to an archive which can be reopened later. The menu also has a “Load” option which can load previously saved data back into the interface for viewing. This feature is necessary for signal processing over different periods of time. Signal analysis can be done at one time, saved, and further analysis can be done at a different time.

#### Level 2

##### Level 2 – Bandwidth Extraction

A close up of a sign

Description automatically generated

Figure 5.2.3.1.1 shows the flowchart for the bandwidth extraction algorithm. The algorithm receives the output from the power calculation and the filtered FFT then passes it first through a power filter. This step of the algorithm will determine a minimum power and replace any power below the cut off with a specific number to serve as a cutoff for each signal in the range. Once the signal has been passed through the power filter, it will be sent to the bandwidth calculation which will take the filtered signal and determine the lower frequency of each signal in the range and the bandwidth of each signal. Both of these outputs will be sent to the center frequency calculation, while only the bandwidths will be sent to the touch screen and classification algorithm. The center frequency calculation will take the lower frequency and bandwidth of each signal and determine the center frequency for each band, then send the list of center frequencies to the touch screen and classification algorithm.

##### Level 2 – Modulation Classification Algorithm

A picture containing table

Description automatically generated

Figure 5.2.3.2.1 shows the flowchart for the automatic modulation classification (AMC). The AMC takes in the radio frequency data. It goes through initial pre-processing to first detect the presence of the signal and to extract the central frequency and bandwidth of the signal. It uses a deep learning algorithm to classify the processed signal to compare against a database of known signals and modulation type. This step is crucial for determining the modulation scheme of the signal which can then be used in the classification algorithm to better differentiate Wi-Fi, Bluetooth, and other signals.

##### Level 2 – Classification Algorithm

The classification algorithm uses the metadata extracted from the signal sample to classify the signal as a Wi-Fi signal, a Bluetooth signal, or an “other” signal. The general process for the classification algorithm is shown in figure 5.2.3.3.1. The selected metadata that will be used are the center frequency, bandwidth, and modulation scheme of the signal. These pieces of metadata about the captured signals are stored in three arrays: the center frequency array, the bandwidth array, and the modulation type array. Each of these arrays is the same size, corresponding to the number of signals captured by the system. A given index for these arrays corresponds to the metadata for that same signal in all three arrays. For example, indexing the third position in these arrays corresponds to the frequency, bandwidth, and modulation scheme of the third captured signal.

A screenshot of a cell phone

Description automatically generated

The classification algorithm loop compares the measured input values to the expected values for each signal classification. The expected values of these signal characteristics are kept in the specifications for both the Bluetooth and Wi-Fi transmission protocols. Wi-Fi is broadcast on 11 channels in the United States with center frequencies starting at 2.412 GHz with a separation of 5 MHz between channels and a channel width, or bandwidth, of 20 MHz, 22 MHz, or 40 MHz depending on the Wi-Fi protocol used (Wi-Fi b/g/n). The modulation scheme used for Wi-Fi is Orthogonal Frequency Division Multiplexing (OFDM), Quadrature Phase Shift Keying (QPSK), and Quadrature Amplitude Modulation (QAM). Bluetooth is broadcast on 79 channels with a channel width of one or two MHz depending on the protocol used (Bluetooth/ Bluetooth Low Energy). The modulation scheme used for Bluetooth is Gaussian Frequency Shift Keying (GFSK). Error thresholds will be used to account for variation in calculation of center frequency and bandwidth, and the signal will be classified if its features are within the allowable range of expected values. The loop then outputs two logical arrays, a Wi-Fi check array and a Bluetooth check array, that contain three logical variables that state whether or not the current signal’s parameters match any of the expected parameters for Bluetooth and Wi-Fi. The algorithm then checks the Wi-Fi logical array for a complete match and returns Wi-Fi if true. If not, the algorithm checks the logical array for Bluetooth for a complete match and returns Bluetooth if true. If neither array matches completely, the output is “other.”

#### Level 3

##### Level 3 – Classification Loop

A close up of text on a black background

Description automatically generated

The signal classification loop looks at the parameters for a single captured signal and uses them to create a Wi-Fi logical array and a Bluetooth logical array representing whether or not the parameters match the expected parameters of Wi-Fi and Bluetooth. Four sets of error values are specified and will be tuned specifically for their parameters. The Wi-Fi and Bluetooth frequency errors represent the allowable deviation of the calculated frequency from one of the expected center frequencies. The Wi-Fi and Bluetooth bandwidth errors represent the allowable deviation of the calculated bandwidth from one of the expected bandwidths. The loop then checks if the calculated frequency is within the tolerance for one of the Wi-Fi channel frequencies and sets the corresponding logical bit to 1 if true and 0 if false. This is repeated for Bluetooth. The loop then checks if the calculated bandwidth is within the tolerance for one of the Wi-Fi channel widths and sets the corresponding logical bit to 1 if true and 0 if false. This is again repeated for Bluetooth. Finally, the loop checks if the calculated modulation type matches one of the Wi-Fi modulation types and sets the corresponding logical bit to 1 if true and 0 if false. This is again repeated for Bluetooth. The two resulting arrays represent how closely the input signal matches a Wi-Fi or Bluetooth signal. If all bits are 1 in a given array, we can safely predict that the signal matches the class associated with that array.

Other algorithms may be explored in the future if time and resources are available, such as machine learning algorithms, in order to improve the versatility of the device. The specific implementation of machine learning models is not discussed as there are many open-source models that can be used without needing to design the model completely from scratch. MATLAB hosts several libraries for implementing classification and machine learning algorithms. Python libraries such as SciKit learn and Keras exist as well to simplify the implementation of machine learning algorithms. Implementation of these algorithms from scratch is unnecessary with the correct choice of programming language.

##### Level 3 – Power Filter

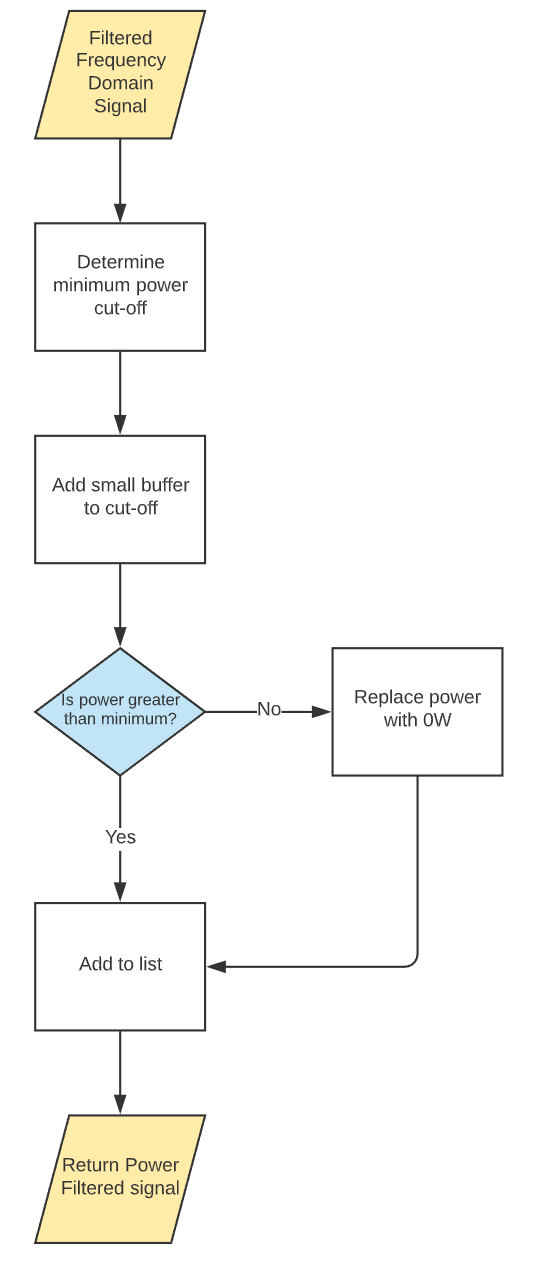


Figure 5.2.4.2.1 shows a more detailed look at how the power filter works. Once the filtered signal is received, we will determine a minimum cut off power. This cut off power will be determined to try to remove anything in the captured signal that is just noise or is not of a high enough power for us to be interested in. Once the cut off power is determined, we will add a small buffer to ensure that we are removing all frequency content that is not important to us. After this has been determined, we will start to sweep through the frequency list and for each frequency we will check if the received power is below the cut off. If it is not, then there will be no change and it will be added to a new list that will contain the now power filtered signal. If the frequency has a power below our cut off, we will replace the received power with 0 and add it to our list. This 0 is meant to act as a delimiter for the next step in the algorithm to see that it is at the beginning or end of a band once it finds a non-zero power.

##### Level 3 – Bandwidth Calculation

A close up of a piece of paper

Description automatically generated

Figure 5.2.4.3.1 explains the bandwidth calculation in more detail. This algorithm will receive the filtered signal from the power filter algorithm and will begin to sweep through the frequencies to determine the first frequency with a power that has not been set to zero. Once this frequency has been determined, it will be saved, and the algorithm will continue to look until it finds the last frequency before the power becomes zero again. It will then take both of these frequencies and pass them to the 3dB frequency calculation. This calculation will return the upper and lower 3dB frequencies and subtract them from each other to determine the bandwidth of that signal. This process will be repeated until the algorithm has swept through all of the frequencies it receives, adding each determined bandwidth to a list of bandwidths to be passed to the next step in the algorithm. Once the calculations are complete, the list of bandwidths and list of lower frequencies for each band will be passed to the center frequency calculation, while the bandwidth list will also be passed to the classification algorithm and directly to the touch screen for the user to see.

##### Level 3 – Center Frequency Calculation

A close up of a logo

Description automatically generated

The final stage of the algorithm is shown in Figure 5.2.4.4.1, the center frequency calculation. This calculation will receive both the list of lower frequencies and the list of bandwidths from the bandwidth calculation algorithm. For each bandwidth received, it will multiply the bandwidth by one half, then add it to the associated lower frequency. This will be repeated for each bandwidth-frequency pair that is received, then the list of center frequencies will be sent to the classification algorithm and to the touch screen to be displayed for the user.

#### Level 4

##### Level 4 – 3dB Frequency Calculation

A screenshot of a cell phone

Description automatically generated

Figure 5.2.5.1.1 outlines the flow of the 3dB frequency calculation. The function will receive the low- and high-end frequency lists from the bandwidth calculation function, as well as the power filtered signal. For each lower and higher frequency pair, the function will extract the portion of the signal between those frequencies to analyze. For the extracted portion, the function will find the maximum power frequency and then search for the frequencies where the power drops below that by 3dB. Once these frequencies are found, they will be added to an upper and lower 3dB frequency list. This will repeat for each set of upper/lower frequencies initially passed to the function. Once the iterations have finished, the new lists of upper and lower 3dB frequencies will be returned to the bandwidth calculation function.

## System Integration and Validation

### System Integration

A screenshot of a cell phone

Description automatically generated

The subsystem division within this project is very much software based, thus the integration of subsystems relies heavily on communication between algorithm files within the file system. The subsystems were divided into signal capture, frequency and bandwidth extraction, modulation type extraction, classification, and user interface. The integration of signal processing and extraction algorithms is simple, as the code for these algorithms can be directly called within the code for the GUI once the “Run” button is hit. However, results of the different algorithms must be saved in a consistent manner for the application to run well. The file system is divided into several folders to integrate the subsystems appropriately. The scripting directory holds all of the permanent code for the algorithms necessary to complete signal capture and processing. The runtime folder holds the necessary files and resultant calculations for the current run. The *data.mat* file stores the results of each algorithm to be output to the GUI, including the frequency array, bandwidth array, modulation array, classification array, and spectrum plots. The *sample.mat* file holds the signal sample output by the GNURadio signal capture code. The *sample.dyn* is a flag that tells the GNURadio code to place the sample file in the runtime directory for the algorithm’s use. The *ran.dyn* file is a flag to tell the GUI that the system has been run at least once, and thus can be saved to the archive if needed. The archive directory stores the *data.mat* and *sample.mat* files for a given run, which can be reopened at any time. Currently, the system has been run using MATLAB generated signal sample files. Future system integration requires the use of file export and import from GNURadio to MATLAB. Consistent file format will need to be developed for the output file of GNURadio so that the signal sample can be read by MATLAB, and so that the signal processing runs as expected.

### Current System Validation

The current validation that has been used to test the system involved generating a known signal in MATLAB and testing the various algorithms using this example signal.

#### Frequency and Bandwidth Validation

A “signal” was generated with frequency content centered around 150Hz and 350Hz, with bandwidths of 100Hz. These signals are shown in Figure 6.2.1.1 below. Once the signals were generated, they were smoothed with a median filter and ran through the bandwidth and frequency extraction algorithms.

Once ran through the power filter, any power below the average power value was set to -100 to distinguish it from our generated signal. This is different from the 0W standard outlined in section 5.3.2.2, but this was just to distinguish the value from the signals. Figure 6.2.1.2 shows the signal after it has been passed through the power filter.

The final results of the algorithm were center frequencies of 150Hz and 349Hz with 3dB bandwidths of 64Hz and 66Hz, respectively. These values are very close to the actual values that were used to generate the signals, but they are slightly different. Due to this inherent inaccuracy, we will consider the frequency correct if it is within 5% of the true frequency.

#### Classification Validation

Early tests of the classification algorithm used sets of frequency, bandwidth, and modulation values that were run through the algorithm and checked manually for correctness. The values of these parameters were shifted to change the class output to make sure the output was correct. The example signals generated in MATLAB consisting of two signals described above was also used and produced the correct output of “Other.”

#### GUI Validation

A screenshot of a social media post

Description automatically generated

The results of the system run using the example signal are shown above. The GUI outputs the correct spectrum of the signal and displays the correct information from the two signals present. The save and load functions were also tested by saving the processing results to the archive directory in the *data.mat* and *sample.mat* files and loading them back in, resulting in the same output being displayed, as expected. This test of the GUI included only the interface coding, frequency extraction, bandwidth extraction, and classification algorithms. Further tests will need to be run to test the integration of the signal capture code and modulation extraction code into the full system once those subsystems have been fully developed.

### Future System Validation

The current system validation has relied on using sample, unrealistic signals generated in MATLAB to determine whether the system behaves as expected. This has allowed us to show that the subsystems have integrated correctly, and the output is as expected. Further testing will require the use of more realistic signals to gauge its performance on real data. Talks are underway with Dynetics to receive isolated samples of real Bluetooth, Wi-Fi, and other signals for further testing. This will allow us to gauge the performance of the system on more realistic data with the benefit of having known parameters to test against. After this phase of testing, we will move toward testing ambient radio signals in the 2.4 GHz range, as this is the environment in which the system is designed to be able to work in.

## Design

Our design is broken down into four subsections: Digital Signal Processing, SDR Software Package, Learning and Classification Algorithm, and Hardware Optimization. Since we have all of the hardware of the previous team, most of our design research was conducted on software. There are some issues with the existing hardware, however, which prompted us to also include research on possibly upgrading the hardware.

### Digital Signal Processing

#### Programming Language

The language we chose from Table 7.1.1.1 for our digital signal processing was option 3: MATLAB/Octave. MATLAB is a language designed for high-performance technical computing. Processing the signals will mean taking in mass amounts of data and performing complex algorithms to determine their frequencies, wavelengths, phases, and amplitudes. In addition, we want to perform signal synchronization and demodulation algorithms. All of which are technically heavy and will require complex algebraic expressions and calculus to determine. So, the high technical load in addition to mass amounts of data makes MATLAB a viable contender. The selection process for the language can be found in section 10.2 of the appendix.

#### Modulation Classification Algorithm

We choose the feature-based for our automatic digital modulation classification algorithm. It is easier to implement and code into our program and, therefore, easier and more efficient editing and troubleshooting. While the likelihood-based algorithm transmits more information while it synchronizes, we do not plan to transmit much data and so, we do not need faster transmission—especially at the expense of ease and understanding.

### SDR Software Package

The software package that we chose from Table 7.2.1 was GNURadio. We chose GNURadio because of its capability of being modified and added to through code, its documentation and resources for learning, and because of the resources available to us from last year’s team to implement a signal capture device.

### Classification Algorithm

The learning algorithm that we have chosen from Table 7.3.1 is K-nearest neighbors. This algorithm has a simple implementation which translates to ease of coding and troubleshooting. This algorithm has the potential to work very well for our specific purpose as it classifies based on clustering. Data that is near each will be classified in the same way, and we believe that our data will fit well into this pattern.

Fitting the model will also be very simple as there is only one hyperparameter that needs to be optimized for our training data. This hyperparameter is a discrete parameter, making it even easier to optimize. This algorithm does not require any training, making the optimization process that much shorter.

The cons of K-nearest neighbors may not be as apparent in our implementation due to the nature of our data and feature space. With a small number of input variables, the number of training examples needed to minimize the model error is also small. This means that storage of the training data for use in predictions will not take up as much space as it may in a much larger model. Feature normalization can easily be coded for, and in our case is particularly easy since our dataset will be small. Outliers may be an issue to this model, however, and during the data pre-processing, we must either gather enough training data to mitigate any outliers or remove the known outliers.

### Hardware Optimization

Two types of the mini pc compare with the Raspberry Pi. The Intel Core i5 is slightly better than core i3 since it has lower power cost and higher clock speed which the most important factor for data analysis. The Raspberry Pi only has the advantage on those factors are not important, so it won’t be the final design solution for this project. Our design selection for the mini pc from table above is the CoreI5 Intel Next Unit of Computing Kit NUC7i5BNK because it has lower power consumption compared to the CoreI3 mini pc. This mini pc is able to handle all requirements of the software, running with a high clock speed.

The Yunlea touch screen is delivered from overseas and has larger size compare with the HDMI 7” touch screen which currently building in the device. The large size means it will reduce the mobility of the device and increase the power cost. So, the HDMI 7” with resistive touchscreen will be used as the final decision.

## Project Plan

### Gantt Chart

Figure 8.1.1 contains our plans for both semesters of our project in the form of a Gantt Chart. All major dates for the project are included in the chart for both semesters. The end dates are the due dates for each component. We have included dates and assignments that pre-date this report. The first chart shows our progress of individual tasks throughout the year as well as the specific dates and tasks shown with their respective owner.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

### Work Breakdown

With the design in place, we must move forward in planning and implementing a project plan. Below is our work breakdown schedule in Table 8.2.1 with all of our activities associated with the device. Each activity has an estimated timeline for completion as well as a person assigned to each activity.

### Major Responsibilities

Table 7.3.1 contains a breakdown of the primary responsibilities of each team member.

### Cost Analysis

Table 8.4.1 contains all of the components we plan to purchase in the near future through Dynetics. Other parts may become necessary to buy depending on performance of the existing hardware during our testing processes.

## References

## Appendix

### Objective Tree Analytical Hierarchy Process

In this section we will discuss how each category got its ranking. Each matrix was either made as a team or made individually and brought to discussion before the team. Table 9.1.1 contains the comparison between the main categories we have divided our project into: signal processing, user friendliness, safety, and cost effectiveness.

Our immediate choice was that the signal processing part of the project was the most important because it is the basis of the project. Our second choice was then that safety would take precedence over user friendliness and cost because there should be no safety concern when using our device. Third was the user friendliness of the device. We chose to give user friendliness the edge over cost because, while we do have a $2000 budget, we were supplied the previous teams hardware and we do not expect to spend very much money on the device.

Table 10.1.2 contains the comparisons for the subcategories under Signal Processing. We divided this into the ability of the device to capture signals, store the captured signals, and analyze any captured signal.

There was a very quick consensus that the most important of these factors was the ability to capture the signal because the device would be pointless if it were unable to do its one task. Second among these factors was the ability to analyze the signal due to similar reasoning to the previous factor. We chose storage as the least important because the other two factors need to happen first, so it must be last logically.

Table 10.1.3 shows the comparison of the subcategories of the Analysis category. We divided this category into the ability to extract requested data (bandwidth, modulation type, etc.), predict what the signal originates from (Bluetooth, Wi-Fi, etc.), and do the analysis in real time. This matrix was harder to choose from, but we ultimately decided that the ability to extract the requested data would take priority because it was asked for specifically from our sponsors. We choose not to make a distinction in the importance of the ability to predict and analyze in real time.

Table 10.1.4 shows the comparison between the subcategories of User Friendliness category. Upon our initial contact with our sponsors, we were told that our focus would be on the data extraction from process signals with little interest in the portability aspect of the device. Since we were told explicitly that it did not matter to our sponsors, we chose to give a slight importance to the ability to use a touch screen for device operation.

Table 10.1.5 shows the comparison between the subcategories of the Portability category. Since we were told by our sponsors that the portability of the device was mostly unimportant to them, and since the previous team had already made the device small and capable of being run off a battery pack, we chose to give no importance to one category over the other.

### Programming Language Analytical Hierarchy Process

Table 10.2.1 outlines the comparison between characteristics of interest of each programming language option: familiarity, speed, and built-in functionality.

These weights will be applied to the languages to determine which should be chosen. Both familiarity and functionality were given importance over speed because the primary goal is to analyze saved data. The speed needed to do this analysis is not nearly that which is required to perform real-time analysis, so it is given the least importance of all three. Ultimately functionality is more important than familiarity, when compared to speed, but when it comes to a straight comparison of the two there isn’t much difference.

Because of this slightly higher priority over speed, functionality is the most important category followed closely by familiarity.

Beginning with Table 9.2.2, the edge was given to MATLAB because it is the language that we are most familiar with. Many classes we have taken have included MATLAB usage at some point in the class, whereas C++ and Python have not been used as often outside of the class, or classes, they have been taught in. Python was given an edge over C++ because of my own personal experience with Python outweighing my experience with C++.

Table 10.2.3 compares the computing speed of each language. Of course, speed varies between different languages depending on the task, but in general these rankings seem to be true. The specific task we are concerned with for the languages is performing the Fast Fourier Transform (FFT) on the signal data.

From [6], we see that MATLAB has a slight edge over Python when performing this particular task. The cases where Python outcompetes MATLAB are the loading times for CSV files, which is not relevant to our project because the output file for GNU Radio is a binary file. The comparison in [7] does not involve MATLAB or the FFT, but it does give an idea of where C++ stands relative to Python. From each of the charts it is quickly observed that C++ outperforms Python by a relatively large margin, so it is given the edge over Python and a slight edge over MATLAB.

Table 10.2.4 we are comparing the built-in functionality of each language. MATLAB already has an FFT function built in, as referenced in [6], but C++ and Python do not. The reason Python is given the edge over C++ in this comparison is because Python’s FFT function comes from the Numpy package, which is required to use GNU Radio, so it is already on our device.

Each of these languages are then brought together for a holistic comparison, including the weights calculated in Table 10.2.1. This comparison is shown in Table 10.2.5

From Table 10.2.5 it can be seen that MATLAB is the clear winner for the programming language. It was the winner of each category individually, besides the speed category. However, the speed category was decided in Table 8.1 to have a very low weight in the decision of the language. There is one issue that remains with MATLAB that is not mentioned in the tables: it is expensive to get a license. Because of this issue, we chose to use Octave. Octave is an open-source alternative to MATLAB with similar syntax, so it will not impact our familiarity rating.

### Classification Algorithm Analytical Hierarchy Process

This section will discuss the analytical hierarchy process [13] used to decide which classification algorithm should be used to identify the origin of input signals. The following table contains the pairwise comparison matrix that was used to create the weights associated with each of the chosen design criteria.

The following tables are the piecewise comparison matrices that compare each of the possible design choices according to the above criteria.

The neural network algorithm and K-nearest neighbors will likely have the highest accuracy. The neural network topology allows for a wide range of decision boundaries and with appropriate training, can prove to be a very accurate model. The K-nearest neighbors algorithm works very well with clustered data. The frequency data in our case will be very heavily clustered, as frequency and modulation type are very heavily associated with the signal origin and vary greatly between signal types. The naïve Bayes’ classifier was thought to be the least efficient because of the high dependency of input features on each other. Frequency and modulation type are heavily linked with respect to the types of signals that we will be capturing. Assuming independent inputs would result in incorrect classification.

K-Nearest neighbors and the naïve Bayes’ classifier were thought to be the most understandable algorithms. The K-Nearest neighbors algorithm has a very simple premise and is a very intuitive way to go about classification. The naïve Bayes’ Classifier has a very simple implementation and requires very little complex analysis techniques, making it a simple technique to understand. Decision trees are also very intuitive, but the method of choosing a location to split the data can be difficult to understand. Neural networks can be difficult to understand due to the high interconnectedness and the advance mathematical process needed to train the model. Mistakes in model implementation can be extremely difficult to troubleshoot.

The neural network algorithm provides the most flexibility in that there are many parameters that can be adjusted to meet the needs of the project.

The K-nearest neighbors algorithm will take the least amount of time to implement due to the simple method of classifying the signal based on distances. The algorithm also requires no dedicated training period, greatly shortening the implementation time. The neural network algorithm would take the longest due to the large computation time required to forward and backpropagate along, the large amount of time needed to optimize the hyperparameters, and the time needed to code the complex structure.

The decision tree and naïve Bayes’ Classifier provide the fastest prediction time due to a low number of mathematical calculations needed. Decision trees only require a small number of comparisons to be made. Naïve Bayes’ only requires counting and simple arithmetic. K-Nearest neighbors would be the next fastest. Typically, this algorithm runs slow with very large amounts of training data, however, our data size will be fairly small relative to usual implementations of the algorithm. The neural network would take the longest time to run due to the large computational requirement of computing may weighted sums and nonlinear activation functions.

The following table shows the results of the analytical hierarchy process. From the results, it can be seen that the K-nearest neighbors algorithm best meets the selected criteria.

### Hardware Optimization Analytical Hierarchy Process

The mini pcs were evaluated on the following criteria: CPU cores, size, cost, weight and CPU clock speed.

The CPU core is the highest category, because it will be used to filter the signal data directly. The major improvement based on the CPU is how to deal with the metadata and display the data in real time. The higher number cores represent higher calculation ability. Lastly, the size is higher than weight and cost in rank because the device size will influence the user feedback.

The touch screens were evaluated on the following criteria: size, resolution, color pixel and cost.

The resolution is the highest category, because the quality of image display on the screen directly relative to this category. The size is also another main factor which will affect the mobility of the device, so it has higher weight compared with the cost and color pixel.