# 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

[Table of Contents 2](#_Toc38321378)

[List of Figures 4](#_Toc38321379)

[List of Tables 4](#_Toc38321380)

[1. Abstract 6](#_Toc38321381)

[2. Problem Statement 6](#_Toc38321382)

[2.1. Need 6](#_Toc38321383)

[2.2. Objective 6](#_Toc38321384)

[2.3. Background 6](#_Toc38321385)

[3. Requirements Specification 7](#_Toc38321386)

[3.1. Marketing Requirements 7](#_Toc38321387)

[3.2. Objective Tree 8](#_Toc38321388)

[3.3. Engineering Requirements 8](#_Toc38321389)

[4. Design Impacts 11](#_Toc38321390)

[4.1. Health 11](#_Toc38321391)

[4.2. Social 12](#_Toc38321392)

[4.3. Economic 12](#_Toc38321393)

[4.4. Manufacturability 12](#_Toc38321394)

[4.5. Standards 12](#_Toc38321395)

[5. Functional Decomposition 12](#_Toc38321396)

[5.1. Hardware Decomposition 13](#_Toc38321397)

[5.2. Software Decomposition 16](#_Toc38321398)

[6. Final System Usage 26](#_Toc38321399)

[6.1. Current Usage 26](#_Toc38321400)

[6.2. Intended Usage 27](#_Toc38321401)

[6.3. GNURadio and MATLAB Code 27](#_Toc38321402)

[7. System Integration and Validation 27](#_Toc38321403)

[7.1. System Integration 27](#_Toc38321404)

[7.2. System Validation 28](#_Toc38321405)

[8. Design 31](#_Toc38321406)

[8.1. Digital Signal Processing 32](#_Toc38321407)

[8.2. SDR Software Package 34](#_Toc38321408)

[8.3. Classification Algorithm 34](#_Toc38321409)

[8.4. Hardware Optimization 36](#_Toc38321410)

[9. Project Plan 38](#_Toc38321411)

[9.1. Gantt Chart 38](#_Toc38321412)

[9.2. Work Breakdown 39](#_Toc38321413)

[9.3. Major Responsibilities 42](#_Toc38321414)

[9.4. Cost Analysis 43](#_Toc38321415)

[10. References 44](#_Toc38321416)

[10. Appendix 45](#_Toc38321417)

[10.1. Objective Tree Analytical Hierarchy Process 45](#_Toc38321418)

[10.2. Programming Language Analytical Hierarchy Process 47](#_Toc38321419)

[10.3. Hardware Optimization Analytical Hierarchy Process 49](#_Toc38321420)

## List of Figures

[Figure 1: Project Objective Tree 7](#_Toc38321421)

[Figure 2: Level 0 - System Overview 12](#_Toc38321422)

[Figure 3: Level 1 - CPU and Touch Screen Interface 13](#_Toc38321423)

[Figure 4: Level 0 - Software Overview 15](#_Toc38321424)

[Figure 5: Level 1 - Software General Structure 16](#_Toc38321425)

[Figure 6: Level 1 - Graphical User Interface 17](#_Toc38321426)

[Figure 7: Level 2 - Bandwidth Extraction 18](#_Toc38321427)

[Figure 8: Level 2 - Modulation Classification Algorithm 19](#_Toc38321428)

[Figure 9: Level 2 - Classification Algorithm 20](#_Toc38321429)

[Figure 10: Level 3 - Classification Algorithm Loop 21](#_Toc38321430)

[Figure 11: Level 3 - Power Filter 22](#_Toc38321431)

[Figure 12: Level 3 - Bandwidth Calculator 23](#_Toc38321432)

[Figure 13:Level 3 - Center Frequency Calculation 24](#_Toc38321433)

[Figure 14: Level 4 - 3dB Frequency Calculation 25](#_Toc38321434)

[Figure 15: Application File System 26](#_Toc38321435)

[Figure 16: GUI Initial Validation 28](#_Toc38321436)

[Figure 17: System Final Validation - First Tab 29](#_Toc38321437)

[Figure 18: Final System Validation - Second Tab 29](#_Toc38321438)

[Figure 19: Final System Validation - Third Tab 30](#_Toc38321439)

[Figure 20: Final System Validation - Fourth Tab 30](#_Toc38321440)

[Figure 21: Project Gantt Chart 38](#_Toc38321441)

## List of Tables

[Table 1: Project Marketing Requirements 8](#_Toc38321442)

[Table 2: Project Engineering Requirements 11](#_Toc38321443)

[Table 3: Signal Classification Device Functionality 13](#_Toc38321444)

[Table 4: CPU Functionality 15](#_Toc38321445)

[Table 5: Touch Screen Functionality 15](#_Toc38321446)

[Table 6: Programming Language Design Choices 32](#_Toc38321447)

[Table 7: Modulation Classification Algorithm Design Choices 33](#_Toc38321448)

[Table 8: SDR Software Package Design Choices 34](#_Toc38321449)

[Table 9: Learning and Classification Algorithm Design Choices 36](#_Toc38321450)

[Table 10: Minicomputer Design Choices 37](#_Toc38321451)

[Table 11: Touch Screen Design Choices 38](#_Toc38321452)

[Table 12: Work Breakdown 42](#_Toc38321453)

[Table 13: Teamwork Breakdown 42](#_Toc38321454)

[Table 14: Cost Breakdown 43](#_Toc38321455)

[Table 15: Objective Tree Main Category AHP Table 45](#_Toc38321456)

[Table 16: Objective Tree Signal Processing AHP Table 45](#_Toc38321457)

[Table 17: Objective Tree Signal Analysis AHP Table 46](#_Toc38321458)

[Table 18: Objective Tree User-Friendliness AHP Table 46](#_Toc38321459)

[Table 19: Objective Tree Portability AHP Table 47](#_Toc38321460)

[Table 20: Programming Language Main AHP Table 47](#_Toc38321461)

[Table 21: Programming Language Familiarity AHP Table 48](#_Toc38321462)

[Table 22: Programming Language Computing Speed AHP Table 48](#_Toc38321463)

[Table 23: Programming Language Functionality AHP Table 48](#_Toc38321464)

[Table 24: Programming Language Final Weight AHP Table 49](#_Toc38321465)

[Table 25: Minicomputer AHP Table 49](#_Toc38321466)

[Table 26: Touch Screen AHP Table 50](#_Toc38321467)

## 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 it 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:

|  |  |
| --- | --- |
| # | Marketing Requirement |
| 1 | System shall detect radio signals within a specified range of frequencies |
| 2 | System shall extract important signal metadata |
| 3 | System shall predict the likely origin of the signal |
| 4 | System shall have a user-friendly interface |
| 5 | System shall comply with FCC regulations |
| 6 | System shall be safe to use |
| 7 | System shall remain within the budget specified by the sponsor |
| 8 | System should log the captured signal data |
| 9 | System should be able to run on battery power alone |
| 10 | System should be compact |

Table : Project Marketing Requirements

### Objective Tree

A screenshot of a cell phone

Description automatically generated

Figure : Project Objective Tree

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 2 contains these engineering requirements, their associated marketing requirement(s), a justification, and a verification method.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Engineering Requirement | Marketing Requirement | Justification | Verification |
| 1 | System shall capture signals with frequencies between 2.4 and 2.5 GHz | 1 | This frequency range captures many common signals that people encounter daily including Wi- Fi and Bluetooth | Generate a known signal such as a Wi-Fi signal to make sure that the device captures the data |
| 2 | System shall capture signals received with a power of at least -100 dBm | 1 | This limit allows capture of relevant signals in the nearby area | Compare the system measurements with an already existing plot of the signal |
| 3 | System shall determine the central frequency of the captured signals with a tolerance of 5% | 2 | This is an important feature of the signal that will be shown to the user and used for prediction | Compare the system measurements with the frequency of a known signal |
| 4 | System shall determine the modulation type of captured signals where applicable | 2 | This is an important feature of the signal that will be shown to the user and used for prediction | Compare the system output to classification of a known signal. |
| 5 | System shall determine the bandwidth of captured signals with a tolerance of 5% | 2 | This is an important feature of the signal that will be shown to the user and used for prediction | Compare the system measurements with the bandwidth of a known signal |
| 6 | System shall correctly identify the classification of the signal (Wi-Fi, Bluetooth, etc.) | 3 | This allows the user to know the use of the signals in the area | Compare the system output with the classification of a known signal |
| 7 | System shall provide the frequency display of the signal to the graphical user interface | 4 | This allows the user to visualize the data and to gauge the accuracy of the prediction against their own knowledge | Compare the frequency plot given by the system to the frequency plot of the known signal |
| 8 | System shall display signal metadata (central frequency, bandwidth, modulation type, and classification) to the graphical user interface | 4 | This allows the user to visualize the data and to gauge the accuracy of the prediction against their own knowledge | Compare the measurements and output of the system to known characteristics of a known signal |
| 9 | System shall remain within the given $2,000 budget | 7 | This is the budget constraint as given by our sponsors at Dynetics | Record the expenses accrued in designing and building the system and ensure the cost remains below the budgeted amount |
| 10 | System shall not transmit radio frequency signals with a power above 10 milliwatts in typical use | 5 | Transmission of radio frequency signals above this power threshold requires a license from the FCC | Ensure the transmission terminal of the SDR remains disconnected during use |
| 11 | System should be safely charged using a 120 VAC power supply | 6 | This allows the battery to be charged conveniently with no danger to the user | Record the voltage of the batteries over time to ensure that the power supply evenly charges the batteries within their ratings |
| 12 | Wiring shall be organized and contained within the device housing | 6,10 | Prevents accidental short circuiting and injury to the user | Check that all wires are secure within the housing, that they do not move greatly, that they do not touch each other unexpectedly, and that the interior cannot be accessed accidentally |
| 13 | System should respond to touch input | 7 | This allows for ease of use by the user | Ensure the touch screen works as expected by testing the different functionalities of the touch screen. |
| 14 | System should save the calculated signal data for future use | 8 | A log of past data samples and results can be useful for future analysis on the system, or for further analysis by the user | Ensure the files can be saved and loaded as expected by running the save and load commands and by checking the directory for the expected files |
| 15 | System should support running solely on battery power for thirty minutes | 9 | This allows the system to be used without the need of a power outlet | Time the system runtime after disconnecting from an AC power source |
| 16 | System should weigh less than four pounds | 10 | This allows for ease of transportation of the system | Use a scale to record the weight of the system |
| 17 | System should not exceed 14” x 9” x 3” | 10 | This allows for ease of transportation of the system | Use a tape measure to record the dimensions of the device |

Table : Project Engineering Requirements

## 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 : Level 0 - System Overview

|  |  |
| --- | --- |
| Module | Signal Classification Device |
| Inputs | * Radio frequency signals * 9V DC power * User input |
| Outputs | * Extracted signal data (frequency, bandwidth, etc.) |
| Functionality | Capture radio signals in the area whose power is over a given threshold, extract characterizing data about the signals and use the data to predict the source of the signals. |

Table : Signal Classification Device Functionality

Figure 2 shows the highest functional level of our device. As specified in Table 3, 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 : Level 1 - CPU and Touch Screen Interface

Figure 3 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 4.

|  |  |
| --- | --- |
| Module | CPU |
| Inputs | * Sample of the signals to be analyzed * User input * Power from the power supply * Signal metadata from external storage |
| Outputs | * Prediction of the signal classification * Signal metadata * Power to other hardware components |
| Functionality | Powered by the power source, the CPU takes the input signals and user input to extract the necessary signal information, classify the signal by its source, and display the information to the user and store it in the external storage for later use. |

Table : CPU Functionality

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.

|  |  |
| --- | --- |
| Module | Touch Screen |
| Inputs | * Power from the CPU * User input * Signal display and metadata |
| Outputs | * Signal display and metadata |
| Functionality | Read the user input and transfer the command to the software. Display and extract the data filtered on the screen. |

Table : Touch Screen Functionality

### Software Decomposition

#### Level 0 – Software Overview

A close up of a logo

Description automatically generated

Figure : Level 0 - Software Overview

Figure 4 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 : Level 1 - Software General Structure

Figure 5 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

Figure : Level 1 - Graphical User Interface

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 : Level 2 - Bandwidth Extraction

Figure 7 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

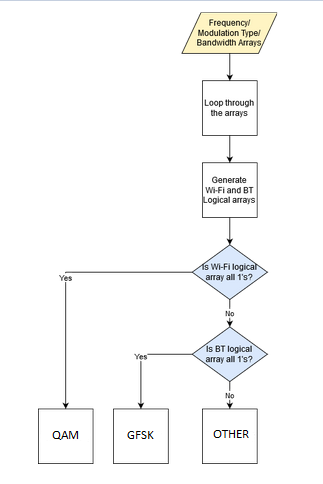


Figure : Level 2 - Modulation Classification Algorithm

Figure 8 shows the flowchart for the modulation classification algorithm (MCA). The MCA takes in the radio frequency data. It goes through the frequency and bandwidth arrays where it’ll generate logical arrays that’ll correspond to either quadrature amplitude modulation (QAM) or Gaussian frequency shift keying (GFSK) or other. After generating the arrays, if all the QAM are logical 1, it’s QAM similarly for GFSK. If it’s mixed it’ll be classified as “other”. With these three classifications, we can cover the Wi-Fi, Bluetooth, and other signals. The output is the modulation type.

##### 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 9. 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

Figure : Level 2 - Classification Algorithm

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

Figure : Level 3 - Classification Algorithm Loop

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

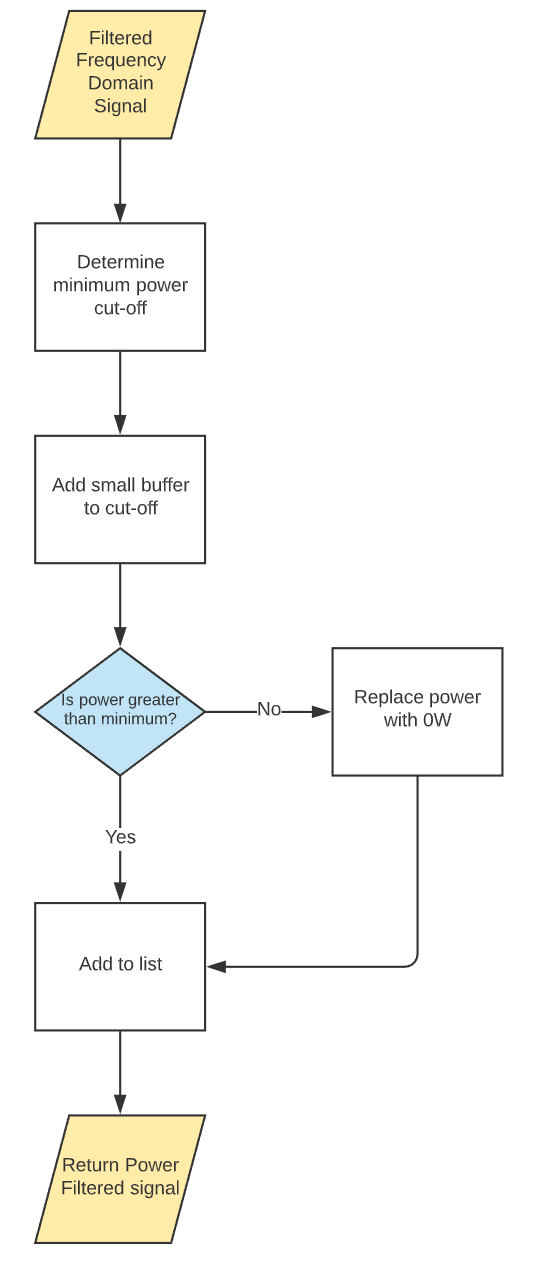


Figure : Level 3 - Power Filter

Figure 11 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 : Level 3 - Bandwidth Calculator

Figure 12 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

Figure :Level 3 - Center Frequency Calculation

The final stage of the algorithm is shown in Figure 13, 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 : Level 4 - 3dB Frequency Calculation

Figure 14 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.

## Final System Usage

### Current Usage

To use the system as we did for testing, a signal must be generated for the MATLAB code to perform analysis on. This signal sample file takes the form of a .mat file of the frequency domain containing two MATLAB variables: an array of each of the sample frequency points and an array of the magnitude of the signal at each of those frequency points. This file is then placed into the Runtime folder in the application directory under the name “sample.mat” Once this file is in the directory, the application file, “Classification.mlapp” can be run in MATLAB. GNURadio is not used as the system stands, since the SDR is not seen by the minicomputer. Once the application is open, the user can press the Run button or press the “R” key on the keyboard to start the signal analysis. The results of the analysis can be saved using the file menu or by pressing the “S” key on the keyboard. Previous analysis using the software can be loaded into the software again using the file menu or by pressing the “L” key on the keyboard.

### Intended Usage

The intended usage of the system is for GNURadio to startup automatically upon starting up the minicomputer. This means that the user does not have to manually open GNURadio for signal capture. The user should then open the MATLAB application and run the application using the Run button or the “R” key. This should signal GNURadio to supply a sample to the directory, and automatically begin signal analysis. In this implementation, there is no need for the user to generate a signal sample, as the sample is captured from our capture algorithm.

### GNURadio and MATLAB Code

The functionality of our code is outlined in the functional decomposition section of this report and can be used to write new code that performs the same basic functionality as the code written for this project. However, our written code can be found in a GitHub repository whose link is supplied in the reference section [8].

## System Integration and Validation

### System Integration

A screenshot of a cell phone

Description automatically generated

Figure : Application File System

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.

### 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 16 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 16 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 Initial Validation

A screenshot of a social media post

Description automatically generated

Figure : GUI Initial Validation

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.

#### Final System Validation

The final system validation was run by using MATLAB to generate a frequency domain sample of signals that we would expect to see as if captured by the signal capture subsystem. Since, the SDR was not being found by the minicomputer during the testing phase, we were unable to capture a test signal of Wi-Fi or Bluetooth signals to run our final validation on. The test signals generated in MATLAB included two Wi-Fi signals at different channel frequencies of 2.422 GHz and 2.462 GHz. These two signals each were given a bandwidth of 20 MHz. The test signals also included two Bluetooth signals at Bluetooth channel frequencies of 2.404 GHz and 2.480 GHz. These two signals were given a bandwidth of 1 MHz. Finally, another signal was given an arbitrary channel frequency of 2.442 GHz with a bandwidth of 4 MHz. This was to simulate an unknown signal other than Wi-Fi or Bluetooth. Finally, a small amount of random noise was added to the generated plot. In the future, more noise should be added to generated test signals in order to simulate real signals. The results of this test are shown in the following figures.

A screenshot of a computer

Description automatically generated

Figure : System Final Validation - First Tab

A screenshot of a social media post

Description automatically generated

Figure : Final System Validation - Second Tab

A screenshot of a social media post

Description automatically generated

Figure : Final System Validation - Third Tab

A screenshot of a social media post

Description automatically generated

Figure : Final System Validation - Fourth Tab

The results shown above shown that the algorithms designed were able to detect the five signals in the frequency plot supplied. The signals were correctly classified and the signal metadate was given to the user in the interface. The calculations for the signal metadata produced outputs to the user that closely matched those used to specify the signals. The application was also able to separate out each of the signals from the main sample and plot them to different plots based on their classification. Only Wi-Fi signals were present on the Wi-Fi plot, and only Bluetooth signals were present on the Bluetooth plot, etc.

## 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

|  |  |  |
| --- | --- | --- |
| Digital Signal Processing | | |
| Language | | |
| General Criteria   * Should efficiently synchronize and demodulate large amounts of signal data * Should be easily readable and writable for future updates | | |
| # | Option Description | |
| 1 | Python | |
| Pros | Cons |
| * Easy to code in and edit * Well documented | * Limited direct access to microcomputer |
| 2 | C/C++ | |
| Pros | Cons |
| * More direct access to microcomputer for data processing * A lot of code flexibility * Well documented | * Less intuitive to read and write in |
| 3 | MATLAB/Octave | |
| Pros | Cons |
| * Fast and efficient at large data processing * Many functions built in for complex data manipulation | * MATLAB is expensive to license, but Octave is similar and open-source |

Table : Programming Language Design Choices

The language we chose from Table 6 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

|  |  |  |
| --- | --- | --- |
| Automatic Modulation Classification | | |
| General Criteria   * Should efficiently synchronize and demodulate large amounts of signal data * Should be easily readable and writable for future updates | | |
| # | Option Description | |
| 1 | Likelihood-Based | |
| Pros | Cons |
| * High accuracy * Well documented | * Require a large amount of prior knowledge-requiring more memory * Difficult to code |
| 2 | Feature-Based | |
| Pros | Cons |
| * Computationally light * Faster synchronization | * Less accurate |

Table : Modulation Classification Algorithm Design Choices

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

|  |  |  |
| --- | --- | --- |
| SDR Software Package | | |
| General Criteria   * Should efficiently synchronize and demodulate large amounts of signal data * Should be easily readable and writable for future updates | | |
| # | Option Description | |
| 1 | GNURadio | |
| Pros | Cons |
| * Large number of packages available for addition * Large amount of documentation and resources to learn * Resources available from the past team * Ease of coding using blocks | * A base program where functionality needs to be coded or installed |
| 2 | GQRX | |
| Pros | Cons |
| * Specific built in functionalities such as waterfall plots and demodulation * Very nice user interface * Fast Fourier Transform capability | * Could not get help from the past team * May not be compatible with necessary GNURadio packages * Many features not applicable to our radio band |

Table : SDR Software Package Design Choices

The software package that we chose from Table 8 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

|  |  |  |
| --- | --- | --- |
| Learning and Classification Algorithm | | |
| General Criteria   * Should support multinomial classification * Should have a short algorithm runtime * Should have a low bias and varianec | | |
| # | Option Description | |
| 1 | Neural Network  A classification model that uses weighted averages in a map of several layers. | |
| Pros | Cons |
| * Large pool of possible models * Good at handling nonlinear decision boundaries * Easy to find open-source implementations | * Model has many hyperparameters that must be optimized * Model is complex meaning difficulties in troubleshooting * Forward and backpropagation are computationally expensive |
| 2 | K-Nearest Neighbors  A classification model that assigns a query the most common label of its K-nearest neighbors | |
| Pros | Cons |
| * Very intuitive; meaning simple implementation and troubleshooting * Only one hyperparameter * No training required | * Prediction is computationally expensive * Requires a normalized and balanced feature space * Very sensitive to outliers |
| 3 | Decision Trees  A classification model that recursively divides the feature space in half based on one feature and finds the best label for each division | |
| Pros | Cons |
| * Very intuitive meaning simple implementation and troubleshooting * Deals with outliers very easily * Minimal data preprocessing needed | * Prone to overfitting * Training is computationally expensive |
|  | Naïve Bayes Classifiers  A model that assumes independence and uses Bayes’ Theorem to predict the value of a query based on frequency in the training data | |
| Pros | Cons |
| * Very simple and intuitive * Fast convergence * Does not require a large training set | * Very important requirement that inputs are independent * “Bad Estimator” meaning it cannot estimate the confidence well * Cannot assign a label that is not in the training data |

Table : Learning and Classification Algorithm Design Choices

The learning algorithm that we have chosen from Table 9 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

|  |  |  |
| --- | --- | --- |
| Hardware and System Optimization: Minicomputer | | |
| General Criteria   * The power supply battery shall not damage other components under any circumstance and should supply power to the device for at least 30 minutes on a single charge * The hardware shall include one small rechargeable battery pack, a user-friendly touch screen of suitable size, and a minicomputer running the Linux operating system * The system casing should be well designed and organized so that all components are easily replacable | | |
| # | Option Description | |
| 1 | Intel Next Unit of Computing Kit NUC7i5BNK | |
| Pros | Cons |
| * Multiple USB connection ports * High clock speed at 2.2 GHz * Consumes 65 W of power | * Expensive * Heavy * Limited in connective ports |
| 2 | Intel Next Unit of Computing Kit NUC8i3CYSM | |
| Pros | Cons |
| * 8 GM RAM; higher than Core i5 * Less expensive * 1 TB hard drive | * Consumes 90 W of power * Larger size than Core i5 |
| 3 | Raspberry Pi 4 | |
| Pros | Cons |
| * Lowest price * About the size of a credit card * Low power usage | * 4 GM RAM * 1.5 GHz clock rate is too slow for data analysis |

Table : Minicomputer Design Choices

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.

|  |  |  |
| --- | --- | --- |
| Hardware and System Optimization: Touch Screen | | |
| # | Option Description | |
| 1 | Yunlea Touch Screen | |
| Pros | Cons |
| * Low price of $40 * Supports running on a variety of operating systems including Linux * Small size | * Must be delivered from overseas * No protection of connection cable * 40 x 20 cm large size |
| 2 | HDMI 7” with Resistive Touch Screen | |
| Pros | Cons |
| * Very small * Lower power consumption | * Lower resolution * Higher Price * Raw 24-bit color pixel data * High-bandwidth digital content protection is not supported |

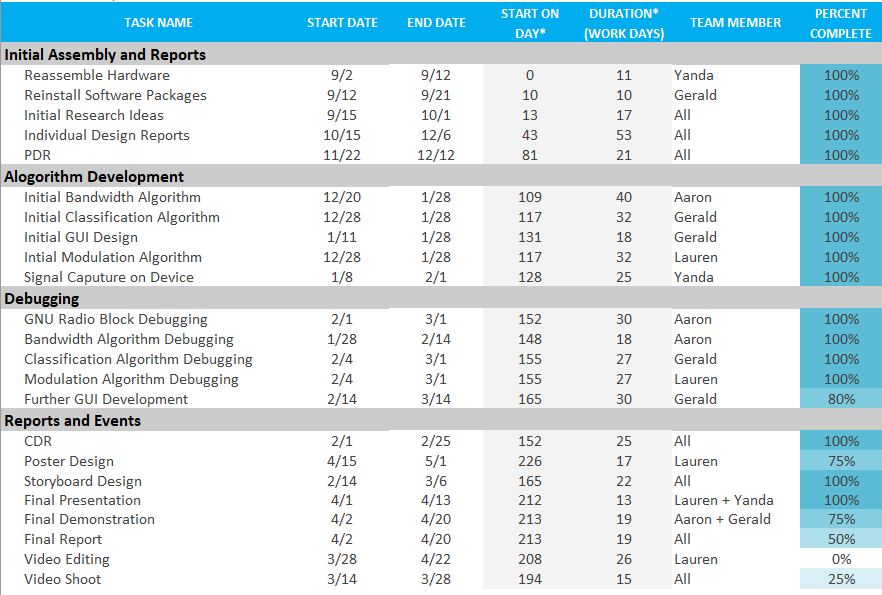
Table : Touch Screen Design Choices

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 21 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.



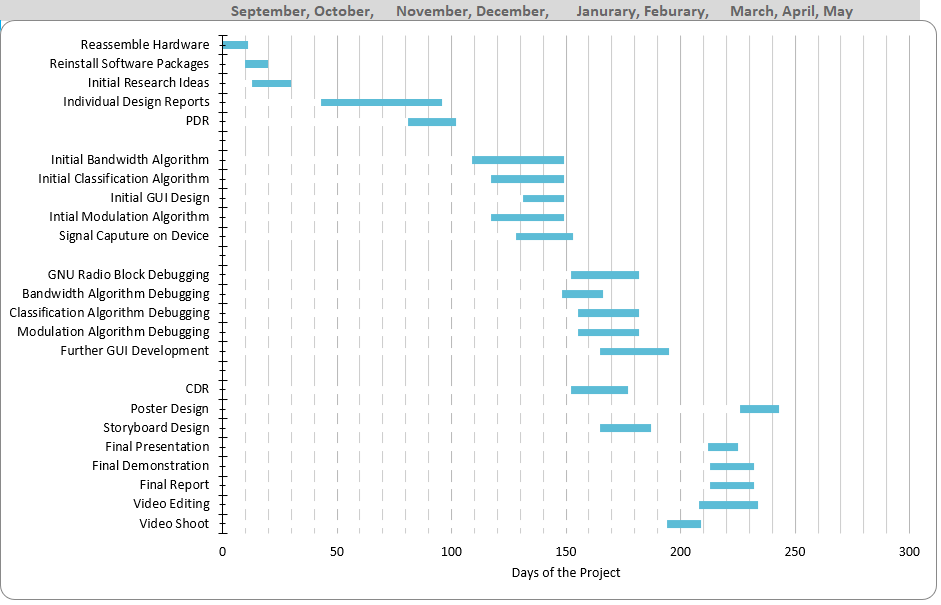


Figure : Project Gantt Chart

### 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 12 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.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Activity | Description | Deliverables | Duration (Days) | Lead | Resources |
| 1 | Install SDR Software Packages | Install all of the necessary software to run our code |  | 35 | Gerald |  |
| 1.1 | Research which Linux platform to install onto the microcomputer. | Linux platform needs to be compatible with the rest of our project and user friendly. | Linux-based platform running on machine. | 5 | Gerald |  |
| 1.2 | Implement prior team’s design. | The previous team’s software should be downloaded for reference point. |  | 5 | Gerald | Grant Cox |
| 1.3 | Research libraries to install and install GNU Radio. | GNU radio and its compatible libraries and Octave libraries used for signal classification need to be installed. |  | 8 | Gerald |  |
| 1.4 | Research GUI design languages and their implementations. | Design must be user friendly and touch screen compatible so the GUI software must reflect that. |  | 35 | Gerald |  |
| 2 | Digital Signal Processing | Install and program digital signal processing code to analyze input signals |  | 28 | Aaron, Lauren |  |
| 2.1 | Frequency and Bandwidth | Research different bandwidth qualities. |  | 21 | Aaron |  |
| 2.1.2 | Phase and Amplitude algorithms | Research algorithms to decipher phase and amplitude information from signals |  | 28 | Aaron |  |
| 2.2 | Signal Synchronization and Modulation |  | Algorithm/program to synchronize gathered signals | 21 | Lauren |  |
| 2.2.1 | Research carrier synchronization | Carrier synchronization algorithm must be picked |  | 28 | Lauren |  |
| 2.3 | Research analog and digital modulation schemes |  | Algorithm to return the modulation scheme of input signals | 28 | Lauren |  |
| 3 | Hardware Upgrades and Repairs |  | Working hardware compatible with software | 30 | Yanda |  |
| 3.1 | Antenna Upgrade | Research and purchase antenna for better range of frequencies |  | 7 | Yanda |  |
| 3.1.1 | Antenna hardware/software installation | Integrate antenna to software |  | 14 | Yanda |  |
| 3.2 | Battery and devise casing upgrade |  | Working battery pack and casing for the battery to connect to the device | 21 | Yanda |  |
| 3.2.1 | Reprint/Buy better casing for the battery | Housing needs to be reprinted for safety and quality |  | 30 | Yanda | UKY makerspace |

Table : Work Breakdown

### Major Responsibilities

Table 13 contains a breakdown of the primary responsibilities of each team member.

|  |  |
| --- | --- |
| Responsibility | Team Member |
| Install SDR Software and GUI Coding | Gerald Bankes |
| Digital Signal Processing – Frequency and Bandwidth Extraction | Aaron Smith |
| Digital Signal Processing – Modulation Type Extraction | Lauren Endicott |
| Hardware Upgrades and Installation | Yanda Cheng |

Table : Teamwork Breakdown

### Cost Analysis

Table 14 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.

|  |  |  |
| --- | --- | --- |
| Item | Price | Distributor |
| NUC i5 Barebones Kit | $499.00 | SimplyNUC |
| 7” Touch Screen | $89.95 | DigiKey |
| Wireless Keyboard/Mouse | $49.00 | Walmart |
| Total | $587.95 | |

Table : Cost Breakdown

## References

1. Keysight Technologies, Appl. Note 150, pp. 5-8
2. M. Ossmann and D. Spill, “What’s on the Wireless? Automating RF Signal Identification”, Great Scott Gadgets, Tech. Report, 2017-1, 2017.
3. General Dynamics Mission Systems, “SignalEye – AI Software for Automated Signal Classification”, gdmissionsystems.com, [Online]. Available: https://gdmissionsystems.com/en/products/electronic- warfare/signaleye[.](https://gdmissionsystems.com/en/products/electronic-warfare/signaleye) [Accessed Sept. 17, 2019].
4. R. Ford and M. Coulston 2008, “Appendix B Decision Making with AHP”, in *Design for Electrical and Computer Engineers,* New York, NY: McGraw-Hill, 2008, pp. 256-258.
5. *FCC Rules & Regulations for Title 47,* FCC Title 47 Part 15, 1989.
6. S. Hanly, “MATLAB vs Python: Speed Test for Vibration Analysis”, August 2016. [Online]. Available: https://blog.endaq.com/matlab-vs-python-speed-for-vibration-analysis-free-download. [Accessed Nov. 15, 2019].
7. J. F. Puget, “A Speed Comparison Of C, Julia, Python, Numba, and Cython on LU Factorization”, January 2016. [Online]. Available: https://[www.ibm.com/developerworks/community/blogs/jfp/entry/A\_Comparison\_Of\_C\_Julia\_Python\_N](http://www.ibm.com/developerworks/community/blogs/jfp/entry/A_Comparison_Of_C_Julia_Python_N) umba\_Cython\_Scipy\_and\_BLAS\_on\_LU\_Factorization?lang=en. [Accessed Nov. 15, 2019]
8. https://github.com/g-bankes/Senior-Design-Project

## 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 15 contains the comparison between the main categories we have divided our project into: signal processing, user friendliness, safety, and cost effectiveness.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Signal Processing | User Friendly | Safe | Cost | Geometric Mean | Weight |
| Signal Processing | 1 | 1 | 3 | 3 | 3 | 0.51 |
| User Friendly | 1 | 1 | 1/3 | 3 | 1 | 0.17 |
| Safe | 1/3 | 3 | 1 | 3 | 1.7321 | 0.29 |
| Cost | 1/3 | 1/3 | 1/3 | 1 | 0.1925 | 0.03 |

Table : Objective Tree Main Category AHP Table

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Capture | Storage | Analysis | Geometric Mean | Weight |
| Capture | 1 | 5 | 3 | 3.873 | 0.755 |
| Storage | 1/5 | 1 | 1/3 | 0.258 | 0.05 |
| Analysis | 1/3 | 3 | 1 | 1 | 0.195 |

Table : Objective Tree Signal Processing AHP Table

Table 16 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Data | Prediction | Real Time | Geometric Mean | Weight |
| Data | 1 | 3 | 3 | 3 | 0.722 |
| Prediction | 1/3 | 1 | 1 | 0.577 | 0.139 |
| Real Time | 1/3 | 1 | 1 | 0.577 | 0.139 |

Table : Objective Tree Signal Analysis AHP Table

Table 17 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Portable | Touch Screen | Geometric Mean | Weight |
| Portable | 1 | 1/3 | 0.577 | 0.25 |
| Touch Screen | 3 | 1 | 1.732 | 0.75 |

Table : Objective Tree User-Friendliness AHP Table

Table 18 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Battery | Size | Geometric Mean | Weight |
| Battery | 1 | 1 | 1 | 0.5 |
| Size | 1 | 1 | 1 | 0.5 |

Table : Objective Tree Portability AHP Table

Table 19 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 20 outlines the comparison between characteristics of interest of each programming language option: familiarity, speed, and built-in functionality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Familiarity | Speed | Functionality | Geo. Mean | Weight |
| Familiarity | 1 | 3 | 1 | 1.732 | 0.41 |
| Speed | 1/3 | 1 | 1/5 | 0.258 | 0.06 |
| Functionality | 1 | 5 | 1 | 2.236 | 0.53 |

Table : Programming Language Main AHP Table

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MATLAB | Python | C++ | Geo. Mean | Weight |
| MATLAB | 1 | 3 | 5 | 3.873 | 0.755 |
| Python | 1/3 | 1 | 3 | 1 | 0.195 |
| C++ | 1/5 | 1/3 | 1 | 0.258 | 0.05 |

Table : Programming Language Familiarity AHP Table

Beginning with Table 21, 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++.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MATLAB | Python | C++ | Geo. Mean | Weight |
| MATLAB | 1 | 3 | 1 | 1.732 | 0.41 |
| Python | 1/3 | 1 | 1/5 | 0.258 | 0.06 |
| C++ | 1 | 5 | 1 | 2.236 | 0.53 |

Table : Programming Language Computing Speed AHP Table

Table 22 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MATLAB | Python | C++ | Geo. Mean | Weight |
| MATLAB | 1 | 3 | 5 | 3.873 | 0.755 |
| Python | 1/3 | 1 | 3 | 1 | 0.195 |
| C++ | 1/5 | 1/3 | 1 | 0.258 | 0.05 |

Table : Programming Language Functionality AHP Table

Table 23 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 20. This comparison is shown in Table 24.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Weights | MATLAB | Python | C++ |
| Familiarity | 0.41 | 0.755 | 0.195 | 0.05 |
| Speed | 0.06 | 0.41 | 0.06 | 0.53 |
| Functionality | 0.53 | 0.755 | 0.195 | 0.05 |
| Score |  | 0.734 | 0.187 | 0.079 |

Table : Programming Language Final Weight AHP Table

From Table 24 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 20 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.

### Hardware Optimization Analytical Hierarchy Process

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Total | CPU cores | Clock speed | Weight | Cost | Size | Weighting |
| CPU cores | 1 | 1 | 5 | 5 | 5 | 0.385 |
| Clock speed | 1 | 1 | 5 | 5 | 5 | 0.385 |
| weight | 0.2 | 0.2 | 1 | 1 | 1 | 0.077 |
| Cost | 0.2 | 0.2 | 1 | 1 | 1 | 0.077 |
| Size | 0.2 | 0.2 | 1 | 1 | 1 | 0.077 |

Table : Minicomputer AHP Table

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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Total | Resolution | size | cost | Color pixel | Weight |
| Resolution | 1 | 2 | 4 | 3 | 0.47 |
| size | 0.5 | 1 | 3 | 2 | 0.28 |
| cost | 0.250 | 0.33 | 1 | 1 | 0.114 |
| Color pixel | 0.33 | 0.5 | 1 | 1 | 0.136 |

Table : Touch Screen AHP Table

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.