Capstone Senior Design Project Report 2
Indiana-Purdue University Fort Wayne
Department of Computer and Electrical Engineering
April 19th, 2016

Project Title
Spectrum Monitoring

Team Members
Justin Gray
Jeffrey Ballinger
Htet Khaing

Faculty Advisor
Dr. Todor Cooklev

Sponsored By
IPFW Wireless Technology Center
# Table of Contents

Section 1:
1.1 Acknowledgements........................................................................................................... 5

Section 2:
2.1 Abstract.......................................................................................................................... 7

Section 3:
3.1 Background..................................................................................................................... 9
  3.1.1 Spectrum Scanning....................................................................................................... 9
  3.1.2 DFT & FFT.................................................................................................................. 9
  3.1.3 Distributed Ground Network....................................................................................... 10

Section 4:
4.1 Problem Statement.......................................................................................................... 12
  4.1.1 Requirements and Specifications............................................................................... 12
  4.1.2 Given Parameters....................................................................................................... 12
  4.1.3 Design Variables....................................................................................................... 12
    4.1.3.1 Hardware Alternatives......................................................................................... 12
    4.1.3.2 Software Alternatives........................................................................................ 13
  4.1.4 Limitations and Constraints....................................................................................... 14

Section 5:
5.1 Digital Sweeping............................................................................................................ 16
5.2 Software-Defined Radios............................................................................................... 16
  5.2.1 RTL-SDR.................................................................................................................. 17
  5.2.2 Airspy....................................................................................................................... 17
  5.2.3 HackRF One............................................................................................................ 18
5.3 Single-board Computers................................................................................................. 18
  5.3.1 Beagleboard............................................................................................................ 18
  5.3.2 Raspberry Pi............................................................................................................. 19
5.4 Website............................................................................................................................. 19
  5.4.1 Programming Languages.......................................................................................... 19
  5.4.2 User Interfaces.......................................................................................................... 20
  5.4.3 Static vs Dynamic..................................................................................................... 21
  5.4.4 Plots and their Variables.......................................................................................... 21
5.5 Hardware Stage................................................................................................................ 21
  5.5.1 Choosing an SDR..................................................................................................... 21
  5.5.2 Choosing an SBC...................................................................................................... 23
5.6 Generating a Conceptual Design (Website)...................................................................... 25
  5.6.1 Design A................................................................................................................... 25
  5.6.2 Design B................................................................................................................... 26
  5.6.3 Design C................................................................................................................... 26
5.7 Generating a Conceptual Design (Hardware)..................................................................... 27
  5.7.1 Design Alpha: MultiPi/Airspy.................................................................................. 27
  5.7.2 Design Beta: MultiPi/RTL-SDR ............................................................................. 27
  5.7.3 Design Charlie: Single Pi/RTL-SDR.......................................................................... 28

Section 6:
6.1 Final Concept Selection (Hardware)................................................................................ 30
6.2 Final Concept Selection (Website).................................................................................. 31
Section 1
1.1 Acknowledgements

First and foremost, the team would like to thank the Wireless Department for sponsoring this project.

We also want to thank our faculty advisor, Dr. Todor Cooklev, for providing us with guidance and support throughout the entire design process.
Section 2
2.1 Abstract

The traditional approach to spectrum monitoring is based on a single spectrum analyzer and a human operator interpreting the results in real-time. The objective of this project is to implement a spectrum monitoring system using very low-cost hardware and a website. The required instantaneous bandwidth is at least 6 MHz and the range to be monitored is at least up to 1 GHz. The measurement results will be displayed on a website.

There are several low-cost, commercially-available hardware components that will be compared. These components contain an RF section, Analog-to-Digital (ADC) Converter, and provide digital I and Q time-domain samples. The commercial hardware will be connected to a computer. The website may be hosted by a different computer. The design team must compute the spectrum by performing FFTs of the IQ samples. The results may be averaged appropriately to reduce noise.

Each FFT output shall be quantized to a nearest magnitude value with a goal of 0.5 dB step size and 96 dB range. A count for the number of “hits” for each frequency-magnitude pair shall be accumulated.

The results will also be packetized appropriately. Ultimately, the results will be displayed on the website, including peak and average values over different time periods such as an hour or a day.

Several parameters will be configurable and will be specified by an appropriate GUI on the website. The design team must understand the tradeoffs related to these parameters. These parameters include receiver sensitivity, number of FFT points, sampling rate, and the amount of time to analyze the entire band (for example 1 GHz).
Section 3
3.1 Background

“Spectrum is the most sought-after resource in telecom today” (Hawn, 2015). Earlier this year, CTIA released a report urging policy makers to free up more than 350 megahertz of additional licensed spectrum by 2019 in order to meet growing demand for mobile broadband (Hawn, 2015).

Spectrum analyzers support many applications including radio frequency fingerprinting and transmission detection. Compliance agencies like Underwriter’s Laboratory use spectrum analyzers for detecting radiated emissions generated by variable frequency drives in electrically commutated motors. Interference, either intentional or resulting from overcrowding, can limit coverage and adversely affects network performance. Spectrum analyzers allow engineers and security officials to monitor and trace the source of interfering transmissions. Spectrum analysis is a fundamental RF technology with a plethora of uses.

Traditional spectrum analyzers are expensive and tend to not scale well. With the rise of inexpensive software-defined radios (SDRs), hobbyists and professionals are finding more options available to them. The primary limitations of such devices have traditionally been sensitivity and accuracy, but improved hardware and availability of higher quality crystal oscillators have lessened the impact of such limitations.

3.1.1 Spectrum Scanning

Modern analyzers work by scanning the frequency spectrum using narrowband receivers. The maximum viewable bandwidth at any one time is referred to as the instantaneous bandwidth of the device. The CPU computes the power spectral density of the signals and plots the results against frequency. The primary drawback of this approach is that transient transmissions that occur outside of the band currently being monitored will go unnoticed. Many transmission types such as Bluetooth occur in microseconds.

3.1.2 DFT & FFT

In computing the PSD of the signals, proven algorithms such as the Discrete Fourier Transform and its variation, the Fast Fourier Transform, are used by the analyzers. The DFT is used to transform a sampled time-domain signal into the frequency domain. The FFT is largely similar to the DFT, but optimized to increase processing speed. A fixed number of samples (N) are collected at a rate called the sampling frequency (f_s). The FFT is calculated for these samples
and a sequence of N complex values are generated. These values give a complete description of the input signal in the frequency domain.

There are some considerations that must be accounted for when analyzing measured signals. First, the minimum sampling frequency required to avoid aliasing is defined by Nyquist to be 2*BW, or twice the bandwidth of the signal to be measured. Second, frequency resolution is defined as \( f_s/N \). To increase resolution, one can either decrease the sampling frequency or increase the number of samples (zero-padding).

3.1.3 Distributed Ground Network

Spectral characteristics vary not only by time and frequency, but also by location. It has become desirable to utilize what is known as “crowdsourcing” to create a distributed network of monitoring stations capable of communicating and sharing information. This would allow one to not only fully characterize a larger geographical area, but determine the relative locations of stationary and mobile sources of interference. This presents a number of issues, not the least of which is cost and the need for a human observer to sit at a terminal located at each station.

By leveraging the low-cost of SDRs and developing a web-based interface, one could create an inexpensive RF spectrum analyzer where a user could log on from a remote location, initiate a scan, and view the results on their personal computer. Such a system would prove to be a huge step towards the realization of a distributed network of inexpensive base stations capable of monitoring a much larger geographical area.
Section 4
4.1 Problem Statement

4.1.1 Requirements and specifications

- Instantaneous bandwidth should be as large as possible.
- Range to be monitored is at least 1 GHz.
- Dynamic range should be at least 96 dB with a 0.5 dB step size
- Cost should be minimal.

4.1.2 Given parameters

The given, or fixed, parameters are those that will be the guidelines for the design of the system. Some of the given parameters include:

- Software-defined radio (SDR) and single-board computer (SBC) will be chosen from commercially available options – will not be built.
- All processing of IQ data will be performed on either the SDR or SBC.
- The website will utilize the data from the hardware stage to generate displays that are both professional in appearance and user-friendly.
- The website will allow the user to define certain parameters such as desired frequency range, monitoring period, and sampling frequency.

4.1.3 Design Variables

The design variables include components in hardware and software. The hardware includes the SDR and SBC. The software includes the website GUI component that needs to be developed for user interface as well as algorithms running on the hardware.

4.1.3.1 Hardware Alternatives

- Software-Defined Radio (SDR)
  - Maximum frequency range
  - Instantaneous bandwidth
  - ADC resolution
  - Cost
- Single-Board Computer (SBC)
  - Processor
  - Embedded memory
  - Interface
4.1.3.2 Software Alternatives

- Programming languages
- Available open source code
- GUI interface
  The 3 variables that need to be represented in the GUI interface are:
  - Power (dBm)
  - Time (sec)
  - Frequency (MHz)

The representation can be done in various ways. One possible alternative is the heat diagram. Heat diagrams plot power in terms of frequency (x-axis) and time (y-axis). The color represents the amount of power of the input signal. An example for the heat diagram is shown in Figure 4.1 below.

![Figure 4.1: Sample heat diagram](image)

Other representations include 2D plots of frequency versus power as shown in Figure 4.2 (left) or time versus power as shown on the right.
4.1.4 Limitations and Constraints

The design must follow limitations and constraints that are outside the scope of the predefined parameters. The main constraint includes the cost and using low power components.

- **Cost:** The project should be implemented using a minimal cost scheme. The total cost for the system should not exceed $4,000.
- **Instantaneous Bandwidth:** The goal is to implement a wideband, SDR capable of monitoring the desired spectrum in as little time as possible. Ideally, this would be accomplished all at once with a very wideband radio. In pursuing this goal, the instantaneous bandwidth of the radio should be as large as possible.
- **The system should be capable of monitoring up to at least 1 GHz.**
- **Computation speed is going to be limited by the processing power of the SBC.**
- **Data transfer rates of hardware interfaces and Internet connection speed will limit our ability to attain lossless transmission.**
- **Data storage:** There is potential to generate a considerable amount of data. Spectrum monitoring allows for lossy compression, but external storage options will need to be considered.
- **Time:** Due to a restriction of a school year to design and build the system, the design must be manageable within the time allotment.
Section 5
5.1 Digital Sweeping

As previously mentioned, spectrum analyzers work by sweeping a narrowband receiver through the desired frequency range. The information within the monitored bands are sampled, processed and stored. The center frequency of the receiver is then tuned to the next band so that the process can continue. This method allows for a narrowband receiver to be used in sweeping a much wider portion of the frequency spectrum.

There are some disadvantages to this method. First, transient events that occur outside of the monitored band will likely go unobserved. This is less of a problem for consistently occupied bands, but other bands are often characterized by random, bursty transmissions that can occur over microseconds. Time spent re-tuning the center frequency of the receiver introduces additional time where no monitoring is being performed at all.

The second disadvantage is limited real-time capabilities. The nature of sweeping the spectrum as opposed to sampling all at once introduces a degree of latency that makes it difficult to achieve real-time spectrum monitoring. These disadvantages can be partially abated by reducing the tuning time as much as possible and maximizing the instantaneous bandwidth of the receiver.

5.2 Software-Defined Radio (SDRs)

Software-defined radio is a type of communication system that uses software to implement traditional hardware components. Figure 5.1 shows a typical SDR concept. An SDR system consists of an RF front end, an analog-to-digital converter, and computer which performs all processing of received signals. Widely expected to become the de facto standard of radio communications, software-defined radio is currently used in cognitive radio, the military’s Joint Tactical Radio System (JTRS), and amateur modulation/demodulation schemes.

While it is not a new technology, SDR has seen a sharp increase in popularity recently due to advancements which have made it significantly more affordable. As can be expected, the marketplace has been flooded with new products. Among the most popular are the RTL-SDR, Airspy, and HackRF One.
5.2.1 RTL-SDR

Originally designed to serve as a cheap DVB-T tuner, the RTL-SDR is based on the Realtek RTL2832U chipset. It outputs 8-bit I/Q samples at a maximum theoretical rate of 3.2 MSPS. The frequency range of the tuner varies widely by individual model with the widest range offered by the Elonics E4000 tuner. The most popular (and cheapest) model is the Rafael Micro R820T which covers the span from 24 MHz to 1.8 GHz at a cost of around $20.

Due to their low cost, the R820T tends to suffer from inaccurate tuning and temperature drift owing to the lower quality crystal oscillators though these failings have been addressed in many tuners with some offering as low as 1 ppm.

5.2.2 Airspy

A somewhat higher-end product, the Airspy builds off the same Rafael chipset, but adds considerably more functionality. Airspy offers a continuous frequency range from 24 MHz to 1.8 GHz, a 12-bit ADC, preselects, 10 MHz instantaneous bandwidth, a 0.5 ppm high-precision
clock, and an on-board Cortex M4 processor with multi-core support. Its I/Q output is scalable down to 2.5 MSPS for low power devices like Raspberry Pi. All of these extras come at a price: the Airspy has a per unit cost of $199, roughly 10 times that of the standard RTL-SDR.

### 5.2.3 HackRF One

Another improvement over the basic RTL-SDR, the HackRF One is produced by Great Scott Gadgets. It is an SDR capable of transmitting and receiving from 10 MHz to 6 GHz. It also has an 8-bit ADC and an instantaneous bandwidth of 20MHz. The market price of the HackRF One is currently $299, one hundred dollars more than the Airspy. Its bandwidth and overall range exceed that of the Airspy, but it lacks the bit resolution and preselects. As a spectrum analyzer, the TX is of no benefit either.

### 5.3 Single-Board Computers (SBCs)

A single-board computer is a complete computer built on a single circuit board, which comes with microprocessor(s), graphics processor, memory, input/output (I/O) and other features to function as a computer. Single Board Computers allow users to create complex systems using PC based technology and offers a powerful and exciting alternative to Microcontrollers and are ideal for processor intensive applications.

The SBC will act as the central processing HUB, running all signal processing blocks as well as running compression algorithms and transmitting data to the server.

Single board computers (SBC), originally intended for engineers, are now appearing in university, secondary, and even elementary school curriculums around the world. As it can be expected, the marketplace has been flooded with new products. Among the most popular are Raspberry Pi and Beagle board.

#### 5.3.1 Beagleboard

The Beagle board is a relative newcomer to the world of easy to use microprocessor breakouts. The board’s based on low-power Texas Instruments processors featuring the ARM Cortex-A series core. The X-15 model comes at a price of $239 and has Dual ARM Cortex-A15 and Dual ARM M4 (212 MHz) processor multicore, 2048 MB of RAM, and on board 4 GB storage. It has 157 GPIO pins and some of its peripherals include 1 USB host, 1 Mini-USB client, 110/100 Mbps Ethernet.
5.3.2 Raspberry Pi 2

The Raspberry Pi hardware has evolved through several versions that feature variations in hardware performance, memory capacity, and peripheral device support. The Raspberry Pi primarily uses Linux kernel-based operating systems. The current release of Ubuntu supports the Raspberry Pi 2 while Ubuntu, and several popular versions of Linux do not support the older Raspberry Pi 1 that runs on the ARM11. The Raspberry Pi 2 currently supports Raspbian, OpenELEC, and RISC OS.

Raspberry Pi 2 uses a 900 MHz quad-core ARM Cortex-A7 which is the biggest improvement from its predecessor which uses a single 700 MHz core. It comes at a price of $35 and has 1GB RAM with 46 GPIO pins. It draws 150-350mA at 5V. However, the additional price comes from SD card, micro-USB cable, HDMI cable, and a keyboard. The Raspberry Pi offers a low-cost and widely popular option, but at the cost of processing power.

5.4 Website

There is an infinite amount of possibilities for what can be done for the website. The possibilities all depend on the programming languages used, what user interface is optimal for the website, and whether the webpages are static or dynamic. The website will also depend on what information will be sent to and from the cloud.

5.4.1 Programming Languages

The programming languages for the website are split into 2 different sides. The first side is client-side, and the second is server-side.

1. Client-side languages
   a. These languages use the client’s station to perform actions and process information that will be viewed on the website. They are used to create visually engaging webpages as well as user interfaces for web applications. Due to the World Wide Web being so large, there are only a few languages that are used.
   b. HTML – HyperText Markup Language is the standard markup language that is used to create webpages.
   c. CSS – Cascade style sheets are used to organize attributes for HTML tags. It makes the HTML documents more organized and readable.
d. Scripting Languages – The main scripting language that is widely used is JavaScript. A new client-side scripting language that is becoming popular is TypeScript. They are used in order to make the website dynamic and interactive.

2. Server-side Languages
a. These languages use the server to perform actions and process information in the background. They can be used to keep a constant connection between server and client. They also can be used to process form data.

b. Server-Scripting Languages – these are the main programming languages used to do scripting on the server. Some popular Object-Oriented languages that are used for this are Java, C#, Python, ASP.NET, PHP, and Server-side JavaScript. Only 2 or 3 of these languages will be used on one website.

c. SQL – Structured Query Language is a language used to create and run databases. The language that is widely used with big data is NoSQL, which stands for Not Only SQL. It uses Java along with a Hadoop distributed file system. Some databases have their own SQL server.

All of these Languages combine together to allow a website to run on a web browser. They also allow the website to be viewed in a different location by sending it across networks to a destination requested by the client.

5.4.2 User Interface

The user interface is what will allow the user to navigate around the website. A user interface with too much information can be overwhelming, but a user interface with too little can make it hard to navigate around the website without knowledge of where links take you. That said, the user interface must be easy to:

- Use – If a user interface is not easy to use, it can bring many questions and people will not want to use it.
- Navigate – All of the main pages should have a link to them on every page. Making simple pages that notify that your station was or was not registered after the user attempts to register it will make the user aware of these situations.
- Read – If the text is too small it will be hard to read. If the text is too big, it will take up too much space, leaving it to look very unprofessional.

There are software packages that will help make the website run better and smoother. This package is the jQuery JavaScript package. Within this package there is a jQuery user interface that can be applied.
5.4.3 Static vs Dynamic

Both static and dynamic webpages will be needed if multiple webpages are implemented in a hierarchy of interconnected webpages. If one dynamic page is implemented, all of the webpages will be dynamic with no refreshing the page every time you click on a link to go to the next dynamic page. Static webpages are pages that load information once. The only time it loads information is when the webpage first opens. A dynamic, or interactive, webpage is a page that will load information to the webpage after the opening the webpage. An example of a dynamic webpage is a webpage that implements a calculator. When numbers are calculated, the webpage will be loading new information for the page to display the result of the calculation.

5.4.4 Plots and their Variables

There are many ways to display the information gathered by the spectrum monitoring stations. A near real-time waterfall plot, Min/Max/Average Plot, near real-time Power Vs Frequency, Power Vs Frequency Vs Time (3D plot), and a passive waterfall plot are some examples of plots that can be used. There are 3 variables that must be transmitted to the website for each plot. These three variables are frequency, power (magnitude), and time. With just these three, almost any plot will be able to display an output.

All other variables that are optional will be determined by the station when the website first connects and registers the station to the cloud. The website will generate textboxes and other user interface tools depending on which variables are selected. To do the near real-time plots, a constant connection will need to be established. This can be done by using the software package SignalR, which uses ASP.NET language as well as JavaScript and C#. This package will add near real-time capabilities to the webpages.

5.5 Hardware Stage

Given the large amount of independence that exists between the hardware stage and the website stage, the two were developed separately. Once a primary conceptual design is chosen, the interface between the two stages will be addressed in the detailed design.

5.5.1 Choosing an SDR

Previously, it was mentioned that when the frequency spectrum is sequentially swept, the probability of missing transient events increases as the size of the instantaneous bandwidth decreases. Given this, it becomes desirable to construct a system with the largest possible
bandwidth. In order to accomplish this, our system will implement multiple SDRs and SBCs. **Figure 5.2** shows an illustration of how each node will work together to scan the spectrum in as few sweeps as possible. An appropriate SDR will need to be selected that can sweep a wide band at one time, has a low per-unit cost, and won’t suffer from significant performance issues.

![Figure 5.2](image)

**Figure 5.2**: Example of two synchronized wideband receivers being used to sweep a much wider band of the RF spectrum.

It should be noted that an alternate strategy would be to employ a similarity-based frequency hopping algorithm. Such an algorithm would hop to different bands semi-randomly, but would visit bands of particular interest more frequently than others. This method has proven effective in similar applications for capturing bursty activity when compared to sequential sweeping (Giustinano et al, 2015).

We compared the three SDRs previously mentioned: RTL-SDR, Airspy, and HackRF One. Each one was evaluated based off of 6 chosen criteria:

1. Cost
   a. This is the per unit cost of the device.
2. Frequency Range
   a. The full range of scannable frequencies available.
3. Instantaneous Bandwidth
   a. The real-time RF chunk that can be monitored at one time.
4. ADC Resolution
   a. The bit resolution of the on-board quantizer. This parameter defines much of the radio’s functionality including dynamic range, sensitivity, reduced imaging, and a lower noise floor.
5. Preselects
   a. The inclusion of front-end analog filters.
6. Extras
   a. The inclusion of extra goods like additional microprocessors.

In constructing our decision matrix, we assigned the following weights:

1. Frequency range = 0.1
   a. All designs have been previously vetted to fit our range requirements. Extra range is considered as a bonus.

2. Cost = 0.2
   a. Cost is the primary limiting factor. As it is likely that every SDR will require its own SBC, cost should be kept low to anticipate this.

3. Resolution = 0.25
   a. Signal analysis systems are largely defined by their ADC resolution.

4. Instantaneous bandwidth = 0.2
   a. A larger bandwidth will reduce the number of SDRs required.

5. Preselects = 0.15
   a. Front-end filters help reduce interference and imaging and will provide a boost to our system’s performance.

6. Extras = 0.1
   a. Considered as a bonus, but only if the extras prove useful to our design.

Our SDR decision matrix can be seen in table T5.1. With a rating of 2.85, the Airspy is chosen to serve as the SDR. It is significantly more expensive than the RTL-SDR, but it covers roughly 4 times the bandwidth at one time and comes with a host of extras. The HackRF One covers a much wider portion of the frequency spectrum than Airspy, but its low resolution, higher price, and exclusion of any preselects or extras were the determining factor.

**Table T5.1: SDR decision matrix**

<table>
<thead>
<tr>
<th>Device</th>
<th>Frequency Range (0.1)</th>
<th>Cost (0.2)</th>
<th>Resolution (0.25)</th>
<th>Bandwidth (0.2)</th>
<th>Preselects (0.15)</th>
<th>Extras (0.1)</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL-SDR</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2.25</td>
</tr>
<tr>
<td>Airspy</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2.85</td>
</tr>
<tr>
<td>HackRF One</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2.35</td>
</tr>
</tbody>
</table>

**5.5.2 Choosing an SBC**

We compared two SBCs previously mentioned: Raspberry Pi 2, and Beagleboard-X-15. Each one was evaluated based off of 6 chosen criteria:

1. Cost
1. Cost of one single board computer.
2. Processor
   a. Defines processing power of CPU
3. Interface Options
4. Embedded Memory
5. Power Draw
6. Community
   a. Added as a tie-breaker.
   b. Strong community and homebrew support can be beneficial in troubleshooting potential issues.

In constructing our decision matrix, we assigned the following weights:

1. Cost = 0.15
   a. Cost is the primary limiting factor. As it is likely that every SDR will require its own SBC, cost should be kept low to anticipate this.
2. Processor = 0.25
   a. It will affect actual performance enhancements depending on the most power-efficient application processor ARM and clock speed.
3. Interface Options = 0.15
   a. Having much more pins/ports and being able to interface for multiple board much easier.
4. Embedded Memory = 0.2
   a. It would greatly help the system operating system.
5. Power Draw = 0.15
   a. It need to be considered to process with a huge load without heating up or operating at its full power.
6. Community = 0.1
   a. If there is a project that will in some way rely on the community for proper operation, one should choose the very active community.

Our SDR decision matrix can be seen in **table T5.2**. With a rating of 2.65, the Raspberry Pi is chosen to serve as the SBC. It is significantly cheaper than the Beagleboard and draws less power which benefits our vision of a distributed network. The Beagleboard’s 1.5 GHz processor makes it a tempting option, but Pi 2’s new quad-core lends it a significant power boost.
Table T5.2: SBC decision matrix

<table>
<thead>
<tr>
<th>Device</th>
<th>Cost (0.15)</th>
<th>Processor (0.25)</th>
<th>Interface Options (0.15)</th>
<th>Embedded Memory (0.2)</th>
<th>Power Draw (0.15)</th>
<th>Community (0.1)</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry pi</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2.65</td>
</tr>
<tr>
<td>Beagle board</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2.45</td>
</tr>
</tbody>
</table>

5.6 Generating a Conceptual Design (Website)

Before generating the designs, there are a few pieces that need to be put together. What programming languages will be used? All of the client-side programming languages stated in section 5.4.1 will be used. For the server-side languages, NoSQL, Java, ASP.NET, and C# will be used. There will be multiple software packages that will be used. The package that will be used the most will be jQuery. What plots will be used? The plots that will be used are both the near real-time and passive waterfall plots, the Min/Max/Average plot, and near real-time Power vs Frequency plot.

5.6.1 Design A

For design A, all pages that have nothing to do with data from the cloud will be static webpages. All plots will be on separate webpages. The Navigation menu will be a bar near the top of the page. The navigation bar will contain links to all of the main pages. All of the main pages are:

- Home
- Plot Selection
- Recorded data
- More Information
- Register Station
- Source Code

Advantages:
- Most user friendly and least complex. It is similar to the way that most websites are formatted on the World Wide Web.
- Favors client-side languages. Server-side will only be used on loading the page with little to no load on the server and for the plots to use the near real-time plots.
- Layout is mostly static pages and favors space for the plots.
- Favors time. There will be enough time to make the website with no bugs, and this allows for more dynamic and better features to

Disadvantages:
• May prove to be unprofessional if the attributes and style sheets for the user interface are not done with precision.
• Having plots on separate pages means that it will take longer to switch from plot to plot.

5.6.2 Design B

For design B, all pages that have nothing to do with data from the cloud will be static webpages. All plots will be one dynamic page. The Navigation menu will be an accordion menu. The navigation bar will contain links to all of the main pages. All of the main pages are:
• Home
• Plot Selection
• Recorded data
• More Information
• Register Station
• Source Code

Advantages:
• The submenus allow for a better navigation through the website. It also allows for the website to have the plots listed within the menu.
• No refresh time when switching plots. If the plot is passive and the user switches to another passive plot, there would be no need for changing settings.

Disadvantages:
• Layout does not favor space. With the accordion menu on the side, the layout will have less room horizontally for plots and their variables.
• The dynamic plot page will prove to be difficult with the amount of complexity that would be needed to make it. This means more time debugging and less time adding features.

5.6.3 Design C

One dynamic webpage. There will be multiple pages, but there will be no refresh going from page to page, making it seem that it is all one page. This is made possible by using AJAX. This design can use either the accordion or bar navigation menus. The selection of the plots will act the same as they do in design B.

Advantages:
• The connection between the pages will be more professional.
• When a link is clicked, it will take you to that link with no refresh time.
• Allows for more formatting decisions. It can support either menu.

Disadvantages:
• Most complex of the three designs. This means that most of the time will be spent debugging instead of implementing.
• Since it is one dynamic page, it will favor server-side processes.
• Most time consuming due to the amount of debugging that will be necessary to make this website work with no bugs.

5.7 Generating a Conceptual Design (Hardware)

With our hardware and requirements outlined, the next item is to determine the optimum implementation. In an effort to increase the instantaneous bandwidth, multiple SDRs will be used to scan the frequency spectrum. There is concern that the chosen SBC will not be up to the task of processing the amount of data that the SDR will output. In anticipation of this, two additional conceptual designs have been generated and will be evaluated along with the primary design.

5.7.1 Design Alpha: Multipi/Airspy

The first conceptual design (henceforth known as Alpha), utilizes multiple Airspy modules and multiple Raspberry Pis connected to the server in a master/slave configuration. Each SDR/SBC will constitute a slave node being controlled by commands sent by the server. They will be synchronized and each one will scan a uniquely defined band. Coordinated scanning will allow the system to sequentially move through the spectrum until the full range has been scanned. The I/Q samples of each SDR will be sent to its respective SBC for processing. Once completed, the samples will be compressed, packetized, and sent to the server for storing. Alpha offers the highest performance when the Airspy module is set for its maximum sampling frequency. However, there is concern that Raspberry Pi 2 will be unable to handle the output when this setting is used. In this case, Airspy does offer a lower 2.5 MHz sampling rate for low power devices. This option would nullify the bandwidth advantage that Airspy holds over RTL-SDR.

Advantages:
• Minimum risk of missing transient events
• High bit resolution system

Disadvantages:
• Most complex design
• Output load from SDR may overwhelm SBC

5.7.2 Design Beta: Multipi/RTL-SDR

The second conceptual design (Beta) differs from Alpha in that the RTL-SDR dongle will be used in place of the Airspy. This will come at the cost of the instantaneous bandwidth and bit resolution. The configuration will otherwise be identical. This design capitalizes on the reduced I/Q output and cheaper price of the RTL-SDR. Throughput for each node is reduced by roughly 84% which greatly reduces the likelihood of data loss and overloading of the SBC processors.
The reason that HackRF One is not considered for alternative designs is that it is anticipated that Airspy’s primary challenge will be excessive throughput. HackRF One uses a slightly lower bit resolution, but its sampling frequency is twice that of Airspy. We will likely encounter the same problems as before.

Advantages:
- Less complex than Alpha design
- Cheaper than Alpha

Disadvantages:
- Increased risk of missing transient events
- Reduced bit resolution

5.7.3 Design Charlie: Single Pi/RTL-SDR

The third design (Charlie) further simplifies the design by reducing the number of nodes to one. A single SDR connected to a single SBC will serve as the monitoring slave node. As the least complex design, Charlie removes the need to synchronize multiple slave nodes and simplifies pipelining of data to the server. It is the cheapest, but it also offers the least amount of real-time coverage.

Advantages:
- Least complex design
- Cheaper than Beta

Disadvantages:
- Highest risk of missing transient events
- Longest time taken to sample frequency range
Section 6
6.1 Final Concept Selection (Hardware)

After analyzing the advantages and disadvantages of each design, the team came up with a decision matrix to choose the best design. The decision matrix includes some important criteria that need to be considered in the final design. These criteria include:

- **Time of implementation**: The time taken for the team to assemble all the components and build the design is the Time of implementation. It is given a 30% weightage in the decision matrix. This is because the team has a time constraint of 1 semester to complete the task.
- **Complexity**: The hardware complexity is another important issue that needs to be considered before choosing a design. Also, the programming expertise demanded by the design, and the prototyping issues need to be evaluated. Thus, complexity of programming and hardware complexity are given a 20% weightage in the decision matrix.
- **Functionality**: While all three designs satisfy our overall design requirements, it is important to gauge the performance and consider which one fits our goals the most. We will evaluate how each fares in the areas of spectrum coverage and bit resolution. This category will be assigned a weight of 40%.
- **Cost**: Cost is an important constraint in any design development. Thus, costs of the designs are compared to each other to come up with the best design. Cost is given a 10% weightage in the matrix. It is given the least importance because the official budget of the project is $4000, and the team believes that the estimated cost of most of the designs is lower than the official budget.

The table **T6.1** shows the decision matrix. The weightage of each criterion is shown in the table 1. Each design is given points ranging from 1 to 3. A value of 3 is given to a design when the team considers it to be good.

1. **Poor**
   a. Design meets bare minimum requirements
   b. Design introduces high level of complexity
   c. Fabrication of design could introduce long lead time
2. **Good**
   a. Design performs well over bare requirements
   b. Design complexity is not excessive
   c. Fabrication of design is not excessively time consuming
3. **Best**
a. Design far exceeds requirements
b. Highly simplistic design
c. Design can be quickly implemented

**T6.1: Design decision matrix**

<table>
<thead>
<tr>
<th>Designs</th>
<th>Time of implementation (30%)</th>
<th>Functionality (40%)</th>
<th>Complexity (20%)</th>
<th>Cost (10%)</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept Alpha</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2.3</td>
</tr>
<tr>
<td>Concept Beta</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2.0</td>
</tr>
<tr>
<td>Concept Charlie</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1.9</td>
</tr>
</tbody>
</table>

**6.2 Final Concept Selection (Website)**

After analyzing the designs for the websites, a decision matrix was formed. The criteria that proved critical to the success of the designs are:

- **User-Friendly** – Will the user be able to use it without having any previous knowledge of how it works? Will they have to read a long list of instructions? These are the questions that were asked to determine how user-friendly each design will be. This is the most important piece of criteria with a weight of 30%.
- **Time** – Will the design be implemented in time? Will there be some time left over for optimizing the website to be better? These are the questions that were asked to determine how time will be a factor for each design. This piece of criteria has a weight of 20%.
- **Interconnectivity** – Is the website navigate fluently with no misleading links? Will there be a lot of refreshing going to another page? These are the questions that were asked to determine how interconnectivity of webpages will be a factor for each design. This piece of criteria has a weight of 20%.
- **Professionalism** – Will there be notifications for errors? Is it too decorative? These are some of the questions that were asked to determine how professional the website will be for each design. This piece of criteria has a weight of 15%.
- **Overall Layout** – Does the website use the space it is given efficiently? Is it too complex? These are some of the questions that were asked to determine how the overall layout will be a factor for each design. This piece of criteria has a weight of 15%.

Table **T6.2** shows the website’s decision matrix. The weightage of each criterion is shown in the table. Each design is given points ranging from 1 to 4. A value of 1 is given to a design that will meet the bare requirements, but will prove to be a problem when implementing. A value of 2 is given to a design that will meet the requirements, but is not the most efficient. A value of 3 is
given to a design that will meet the requirements and is efficient. A value of 4 is given to a design that will meet the requirements, and proves to be the most efficient of the designs.

**T6.2: Website Decision Matrix**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Importance Weight (%)</th>
<th>Concept A Rating</th>
<th>Weighted Rating</th>
<th>Concept B Rating</th>
<th>Weighted Rating</th>
<th>Concept C Rating</th>
<th>Weighted Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Friendly</td>
<td>30</td>
<td>4</td>
<td>1.20</td>
<td>3</td>
<td>0.90</td>
<td>2</td>
<td>0.60</td>
</tr>
<tr>
<td>Time</td>
<td>20</td>
<td>4</td>
<td>0.80</td>
<td>3</td>
<td>0.60</td>
<td>1</td>
<td>0.20</td>
</tr>
<tr>
<td>Interconnectivity</td>
<td>20</td>
<td>3</td>
<td>0.60</td>
<td>3</td>
<td>0.60</td>
<td>4</td>
<td>0.80</td>
</tr>
<tr>
<td>Professionalism</td>
<td>15</td>
<td>2</td>
<td>0.30</td>
<td>3</td>
<td>0.45</td>
<td>4</td>
<td>0.60</td>
</tr>
<tr>
<td>Overall Layout</td>
<td>15</td>
<td>4</td>
<td>0.60</td>
<td>2</td>
<td>0.30</td>
<td>3</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>3.5</td>
<td>2.85</td>
<td></td>
<td></td>
<td></td>
<td>2.65</td>
</tr>
</tbody>
</table>
Section 7
7.1 Detailed Design

![Diagram of final conceptual design for Spectrum Monitor]

**Figure 7.1:** Final conceptual design for Spectrum Monitor

Our final conceptual design will utilize 4 sensor nodes which will act together to scan the full frequency range of the radios. Each sensor will have a dedicated frequency range which will be a subset of the overall range. Each node will utilize a single SBC, SDR, and omni-directional, monopole antenna. The SDRs will pass all raw data to the SBC for processing. Each SBC will run the open-source library *librtlsdr* to initiate scans and process data. The functions of the library will be detailed further in this report, but it currently works to divide work into signal processing blocks such as sampling/segmentation, windowing, FFT, averaging, and compression. An automated script will initiate scans and transmit the output files to the server via FTP. Note that the design team had originally opted to use Vita-49, but later decided to use FTP since Vita-49 seemed excessive for the desired function. A user operating the website would send a request for a certain time span and the server would be queried for the relevant data. Said data would be retrieved and sent to the website where an output plot would be generated.

7.2 Antenna

The antenna used will be a simple omnidirectional “whip”-style antenna. These tend to be hardy, cheap, and easy to work with. An omnidirectional dis-cone antenna would provide better reception quality, but these tend to be bigger and more expensive. Neither of these qualities are desirable for the envisioned distributed network. Due to their low cost, it will be possible for each sensor to have its own antenna. It will be terminated with an SMA connector which is more common than the MCX connector and offers lower insertion losses.
7.3 Sensor

The spectrum monitor will implement features such as sampling, segmentation, windowing, DC removal, FFT, averaging, compression, etc. It will process the I/Q data received from the RF front end, compress it, and transmit it to the server for storage. A sensor manager will initiate scans and transmit them to the server upon completion.

7.3.1 Software-Defined Radio

The original design called for use of the Airspy brand SDR dongle. While it overpowered the alternative dongle in performance, the Airspy proved to have extremely limited software support. As a result, the use of Airspy would have required extensive re-writing of the source code. The design team did not feel confident that we could accomplish such a task in the limited amount of time available. After much deliberation, it was decided to move to the alternative design which utilizes the RTL2832U dongle. This particular dongle enjoys a significantly larger amount of community support and is a more proven product.

Due to time constraints, it was decided to scale the design down to Charlie (single node system).

7.3.2 RTL-Power

Open-source library *librtlsdr* possesses a wide range of functions specifically suited for software-defined radio. One of which is RTL_POWER.C. This function was originally written by Kyle Keen (Keenerd) in August of 2014 and officially included in the *librtlsdr* library in that same year. RTL POWER allows a user to scan a defined range within RTL SDR’s allowable bandwidth. It also allows the user to set a number of constraints such as desired resolution, integration interval, scan duration, gain, and windowing function. The process may be better explained via demonstration.

![Figure 7.2: CLI input calling RTL_POWER; FM band scan with 10 kHz resolution, integration interval of 1 minute, single-shot mode, output to FMScan.csv](image)
Here is an example of a scan request for the FM band between 88 MHz and 108 MHz. The requested resolution is 10 kHz. The integration interval is set for 1 minute and the system is set for single-shot mode (scan for a single integration interval and exit). The scan is set to output all data to the file FMScan.csv.

### 7.3.3 Frequency Range

Once the command is sent, the program begins by initializing all global variables. A switch case is used to accept all user inputs and assign them to their relevant variables. The first internal function to be called is `frequency_range()`. This function accepts the maximum and minimum range as well as the desired resolution. The algorithm iteratively determine the bandwidth of the dongle by taking the difference between the upper and lower frequency and dividing by an integer between 1 and 1,500. It repeats this process until it reaches the largest possible number that is less than 2.8 MHz. The integer becomes the number of re-tunes or “hops” that the dongle will make. The resulting frequency is the bandwidth of the dongle. In this example, the range is 20 MHz. The result is that the dongle will hop 8 times with a bandwidth of 2.5 MHz. If the full range had been requested, then the bandwidth would be 2.793 MHz and it would hop 600 times. However, if the range is less than 1 MHz, then the number of hops is set to 1, downsample is set to 2.8 MHz divided by the range, and the range is then set to 2.8 MHz. Note that the order of these actions are important for obvious reasons.
7.3.4 Resolution

The function then sets out to determine bin size and the number of bins, which directly relates to the resolution seen. Bin numbers are always a power of two and the maximum number of bins allowed is 2²¹. It iteratively determines the bin size by dividing the dongle bandwidth by increasing powers of 2 and multiplying by the downsample value until the bin size is below the requested resolution. In our example, our requested resolution was 10 kHz. The algorithm determines that the resolution should be 9,765.62 Hz which corresponds to a 256-point FFT. Note that in the output seen in figure 7.3, the number of bins indicated is the number of bins per hop multiplied by the number of hops.

If the requested resolution is set higher than 1 MHz, then the resolution and dongle bandwidth are set equal to the requested resolution, the tune count is recalculated, and the number of bins per hop is set to one. Crop function is also disabled in this mode.

If the tune count exceeds 3,000, then the program will return an error since the range requested will be too large for the dongle. It then calculates the proper buffer size based off the product of downsample and the number of bins. This buffer space is used to eliminate or at least reduce the number of dropped samples during operation.

7.3.5 Scanning Intervals

Upon returning from the frequency range function, the program will proceed to set the following:

1. File name
2. Enforce a minimum interval of 1 second
3. Index the desired device if multiple dongles are present
4. Open the device
5. Set gain
6. Set PPM error correction

It then sets the value of next_tick and exit_time. Next_tick is the moment in time when the scanner will exit to report its results. This is calculated by adding the interval to the current time similar to how a person might set an alarm letting them know when something is due. Exit_time is the total amount of time the user would like the scanner to operate. For example, if one were to set the dongle with an interval of 5 minutes and an exit time of 24 hours then the
device will scan and report in ~5 minute intervals continuously over a 24 hour period before closing down for good. It is important to either set the scanner in single-shot mode (perform one integration interval and exit) or set the exit timer otherwise the scanner will operate on an infinite loop until the user manually shuts down the scan. It is also important to note that the time taken to scan a given range will depend on the size of the range. If either the interval or exit timer expire mid-scan, the algorithm will not exit the scan until it has finished its current sweep.

### 7.3.6 Scanning

Once the timers are set, it begins scanning by calling scanner(). The center frequency is set to the desired frequency and raw samples are read to the buffer. It then removes any DC content from the signal and performs the desired windowing function. Windowing is an important step as performing FFT on a non-periodic signal can result in skewed results. Windowing works by attenuating the signal in a way that depends on the chosen function. RTL_POWER defaults to a rectangular window function which attenuates the signal at the beginning and end of the window while leaving the center section unchanged. Other options such as Harris and Hanning are available, but will not be explored in this design.

Finally it performs the FFT and records the results to the memory. The algorithm used is a variation of the Cooley-Tukey FFT algorithm known as the radix-2 decimation-in-time (DIT) FFT. It works by dividing an N-point DFT into two N/2-point DFTs. See Appendix 12.2 for more information on radix-2 algorithm.

Once a scan is completed, it will check if either the integration interval or exit timer has been exceeded. If not, then it will repeat the process. Since the program records a single point for each bin, having an interval longer than the time taken to sweep the range will result in more samples taken and a stronger average. If the time has been exceeded, it will exit the loop and record all values to the output file. The current date and time are stored as well. It then calls csv_dbm to open the file and write the values. The output file is formatted to show date/time of scan, lower bound of the frequency bin, and power value in dBm. Lossy compression has been added to the original program by the design team. All unnecessary spaces are removed and all power values are resolved to single decimal precision. The function then performs any necessary cleanup before exiting.

If the exit timer has been exceeded or if the device is operating in single-shot mode, then it permanently exits the loop and performs any necessary cleanup such as closing the output file and deallocating memory before exiting.
7.3.7 Transmission

Originally, it was determined to use VITA-49 as the transport protocol between the sensors and the server, but subsequently we decided to go with FTP for simplicity. Filezilla Server is installed on the server while FTP client is installed on the sensor nodes.

An automated script is executed on the sensor which launches a series of scans using predetermined specifications such as range and resolution. Once each scan has finished, a sub-script is executed which contains all relevant FTP commands for connecting to the server, sending the file, and disconnecting from the server.

Currently, the script is set to run a scan from 24 MHz to 1.7 GHz with an effective resolution of ~5.5 kHz. The resulting output file is 11MB and takes ~40 seconds to complete. The script immediately executes the FTP commands which take ~4.5 seconds to transmit. The file size, sweep time, and transfer time may be adjusted by changing the desired resolution. FM scans take about ~11 seconds with a resolution ~315 Hz. The file size ended up being around 2.27MB for each FM scan.

7.4 Router

The router will act as the interface to connect the sensors, cloud server, and clients into a small LAN. The sensors will not communicate with the website directly. The website will also not communicate with the sensors directly. In order for information to be transmitted between the website and the nodes, the information must be sent through the server first. This allows the server to notify if the station that is being requested is online or not. The station in this case is the 4 nodes. The router will use static IP addresses that are assigned manually to each node, the cloud server, and the client PC.

7.5 Cloud Server

The cloud server will act as the central storage database for the information collected by the sensors. New scans will be stored on the server prior to being sent to the website. It will consist of a server PC and a large amount of storage space. The PC will be an inexpensive barebones running Windows Server 2016 Technical Preview with a virtual machine handling all Hadoop tools. Ubuntu Server OS and Hadoop Distributed File System were originally what the server was supposed to contain as main software components, but there were compatibility issues with the .Net framework and the Mono Server. That also changed what the solid-state
drive’s intentions since it was too small for the Windows Server OS. The Website and its temporary data is stored on the solid-state drive. For storage and the OS, we use a 5 terabyte internal HDD. It is the belief of the design team that 5 TB will suffice for preliminary testing purposes.

The website will then be sent to the client pc after the client pc sends a HTTP request for it. Having the website on the SSD allows for faster file transfer for the temporary files that are stored and deleted. The plan was to utilize the 5TB drive for storage, but it was used to store many features that will help develop more of the website in the future. The server will only be heavily utilized when the client is using the near real-time plots because of transferring data through a SignalR2 hub, but it will not be enough to slow down the process of collecting information from the nodes through the router.

7.6 Website

The website is an ASP.NET Web Application which displays the requested data that is available from the cloud through graphical plots. Only the cloud server will communicate with the website, which means that it will not connect to the sensor nodes directly. Currently, the user may only change the viewable frequency range, as well as changing to preset ranges through a drop box, but future iterations could allow the user to select windowing functions and FIR options.

Figure 7.4 shows the current version of the website which includes the home page and the 2D Power Vs Frequency Plot. Time would not allow for functionality to be given to the Waterfall plot. All pages and paths through the website have been included so that developing can be done in the future for each page. Many of the paths lead to a partial page that notifies the user that the page has not yet been developed or finished. The views are formatted using Razor Syntax, which uses C#. This allows one to balance the usage of client-side and server-side programming for the website.
The Web application is accessible under the domain name IPFW.spectrummonitoring.net purchased through GoDaddy.com, which will share the domain name with other domain name servers. The server is hosting the website with IIS server. Currently, the Web Application can only be accessed from within IPFW’s network. One could access it from the outside, but the user would have to log into an external VPN.

### 7.7 Output Plot

The Power Vs Frequency plot is the sole functioning plot at this time. It has been formatted such that the user will restrict the frequency range that they wish to view and the plot will scale to fit that range. The plot is capable of displaying the scans stored in its database sequentially, creating a dynamic plot which updates when a new scan is detected. The units for frequency are in Hertz and power is in dBm.

The client-side programming is done in JavaScript. The programming is put inside of a jQuery function that will call when the document is done loading. SignalR2 hubs were used to transfer the data. These hubs are written in C# and are dynamically created through the assemblies and JavaScript support files.
Section 8
8.1 Testing

The following test plan was formulated in order to gauge the effectiveness and robustness of the design. The only test that is considered Critical to Pass (CTP) is the Core Functionality (CF) test. Additional testing will only be conducted if time and resources are permitting. They are not CTP and, therefore, not required.

8.1.1 Core Functionality

Core functionality will be determined as follows:

1. Initiate a scan using an automated script
2. Transmit the finished scan to the server upon completion
3. Detect the presence of the new file and generate a plot on the website

No real-time or near-real-time functionality is required for CF testing. Frequency range should be at least 1 GHz.

8.1.2 Multiple Sensors

Same as CF, but multiple sensors will be used. Hypothesis is that increasing the number of sensors will increase the amount of instantaneous coverage allowing for more bursty transmission to be captured. Frequency range will be divided into equally-sized sub-intervals and delegated to each sensor. Output plot will be compared to that of CF and analyzed for improvements to resolution. Metric will be difference in number of bursty transmission seen between the two. If it is possible, design team may generate the transmission from a noise source.

Due to the fact that we were unable to properly implement multiple dongle functionality, this test will not be performed.

8.1.3 Proximity Distortion

It is known that analog circuitry in low-cost dongles can be prone to distortion when placed in close proximity to strong transmission sources. Sensor will be tested at varying distances from a known source transmitting a tone. If distortion occurs, it will present itself as harmonics of the tones fundamental frequency. These harmonics should become worse as distance decreases.

Time and resources did not permit this test to occur.
8.1.4 Near-Real-Time

NRT functionality will be gauged by attempting to generate a waterfall plot. This differs from the previous heatmap plot in that the display will stream downwards, displaying new information as it is received. To reduce network congestion, the NRT plot will be constrained to the instantaneous bandwidth of a single dongle (2.5 MHz). Design team will measure latency by generating a signal and measuring the time taken for the signal to appear on the waterfall.

NRT capability was not finalized for this design.

8.1.5 FM Analysis

It is known that one can discern between whether an FM station is broadcasting music or speech by observing the frequency spectra of that particular station. If there is activity across the band, then it is likely they are playing music. However, if the frequency components are constrained to the lower portion of the band then it is likely that they are broadcasting speech. Such a test will gauge how useful the system is at performing spectrum analytics. The sensors will be tuned to a particular station and a narrowband analysis will be conducted. The design team will attempt to discern speech transmission from music transmissions.

8.2 Results

8.2.1 Core Functionality

To gauge core functionality, a script was run to automatically generate scans over the bandwidth of the dongle with a resolution of 10 kHz. Owing to the resolution algorithm, the actual resolution ends up being ~5.7 kHz. Upon completion the scans are automatically sent to the server for storage. Once the website detects the presence of a new scan, it sets out updating the display. This behavior is monitored via monitors attached to the sensor, the server, and the website.

The script is executed and the scan commences. It takes the sensor ~40 seconds to complete the scan and ~0.5 seconds to transmit it. Shortly after sending has completed on the sensor side, the server monitor shows that the file has been received. Within seconds, the website display updates to show the newly acquired data. The process continues until the scanning process is manually halted.

The website defaults to show the full frequency range when the user first navigates to the plotter. Slider bars on the side of the screen allow the user to scale the display to view whichever portion of the spectrum is important to them. Due to issues with the scaler and limited time, the X and Y-axis resolution cannot be displayed properly leaving the user unable to view the axis tick labels, however the user is able to view the max and min frequency, giving them some indication of the range and scale.
Figure 8.1: Full range scan over dongles maximum bandwidth

In figure 8.1, a sampled scan over the dongle’s full range can be seen. The user can see various peaks throughout the range corresponding to various transmission types. The slider bar seen on the left can be used to alter the maximum and minimum bandwidth seen. The display will automatically rescale to fit the desired range to the window. Figures 8.2 – 8.4 were generated by selectively scaling the frequency range to view particular bands such as the cellular band and the FM band.
Figure 8.2: Full range scan; display has been slightly altered via the slider bars on the left of the window.

Figure 8.3: Scan showing a portion of the cellular band from 800 MHz to 969.5 MHz
Figure 8.4: Scan showing entire FM band from 88 MHz to 108 MHz

The current output allows the user to get an idea of how the spectrum is being used in their area, but without information pertaining to resolution or tick labeling, the usefulness is limited. Obviously, this is not ideal, but the website and sensors have displayed the ability to generate, transmit, store, and display scans of the RF spectrum. Overall, the system passes the CF test.

8.2.2 FM Analysis

The radio station 98.9 was chosen as it is known to be relatively close and produces a strong signal to our testing location. One hundred scans over the FM band (88 MHz to 108 MHz) were performed over a 20 minute period with a resolution of 320 Hz. During that period of time, the station made the following broadcasts:

- 9:43 A.M. - "You've Got Another Thing Coming" - Judas Priest
- 9:48 A.M. - Short Promo
- 9:49 A.M. - "Under the Bridge" - Red Hot Chili Peppers
- 9:55 A.M. - 10:01 A.M. - Commercial Break
Figure 8.5: FM scan of radio station broadcasting music

Figure 8.6: FM scan of radio station broadcasting commercial
It can be seen that during periods where the station was playing music, the spectrum appears as a soft "hump" extending over the station's bandwidth. This is due to the energy of the signal being spread out...
across a wide range of frequencies. However, commercials tend to have more voice than music components. This is evident by the presence of large spikes in the spectrum occurring approximately at the station’s center frequency.

While certainly not definitive, the plots obtained suggest that it is possible to surmise, with limited accuracy, whether a station is broadcasting voice or music. The system passes the FMA test, but could use further development.
Section 9
### 9.1 Bill of Materials

<table>
<thead>
<tr>
<th>Description</th>
<th>Supplier</th>
<th>Part Number</th>
<th>Quantity</th>
<th>Cost</th>
<th>Extended</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTLSDR RTL2832U DVB-T Tuner Dongle w/ 2x Telescopic Antennas</td>
<td>RTL-SDR.com</td>
<td>RTL2832U</td>
<td>1</td>
<td>$24.95</td>
<td>$24.95</td>
</tr>
<tr>
<td>Raspberry Pi 2, Model B Basic Starter Kit (Pi 2, Clear case, Power Supply)</td>
<td>Prestige Milano (Amazon.com)</td>
<td>none</td>
<td>1</td>
<td>$49.99</td>
<td>$49.99</td>
</tr>
<tr>
<td>Samsung 32GB EVO Class 10 Micro SDHC</td>
<td>Amazon.com LLC</td>
<td>MB-MP32DA/AM</td>
<td>1</td>
<td>$11.99</td>
<td>$35.96</td>
</tr>
<tr>
<td>CAT 6 Ethernet Cable, 5 foot</td>
<td>Monoprice</td>
<td>3427</td>
<td>10</td>
<td>$2.10</td>
<td>$21.00</td>
</tr>
<tr>
<td>10/100/1000Mbps, 5-port Gigabit Switch</td>
<td>TP-LINK</td>
<td>TL-SG1000SD</td>
<td>1</td>
<td>$19.99</td>
<td>$19.99</td>
</tr>
<tr>
<td>Wireless 750Mbps Gigabit Router</td>
<td>D-Link</td>
<td>DIR-836L</td>
<td>1</td>
<td>$29.98</td>
<td>$29.98</td>
</tr>
<tr>
<td>ThinkServer Server Computer</td>
<td>Lenovo</td>
<td>TS140</td>
<td>1</td>
<td>$244.99</td>
<td>$244.99</td>
</tr>
<tr>
<td>5TB SATA HDD</td>
<td>Toshiba</td>
<td>PH3500U-1I72</td>
<td>1</td>
<td>$144.99</td>
<td>$144.99</td>
</tr>
<tr>
<td>SATA III Cable</td>
<td>Monoprice</td>
<td>8782</td>
<td>1</td>
<td>$3.51</td>
<td>$3.51</td>
</tr>
</tbody>
</table>

Total Cost: $575.36
Section 10
10.1 Conclusions

In August 2015, the design team set out to create a spectrum monitor which utilizes commercially-available off-the-shelf parts and open-source software. The hardware was to generate scans and display the results to a website for remote viewing.

During the preliminary design phase, many hardware and software options were evaluated and, ultimately, three design options were generated. The primary design would utilize multiple high-end SDRs along with multiple SBCs to increase the size of the real-time spectrum that will be monitored at any one instance. This design offered a reduced probability of missed transient events and a higher bit resolution. This design ended up being shelved due to incompatibility with our chosen software. Airspy is still a viable option for future iterations given that the design team has sufficient time and possesses adequate software coding skills.

Our next alternative utilized multiple Realtek dongles. Though the source code was compatible with the dongles, in the end the design team ran out of time to pull this design together.

Ultimately, it fell to design Charlie which utilizes a single Realtek dongle. The design team was able to successfully generate scans and plot the results using a single dongle connected to a wired network and accomplished this while using only slightly more than 1/8th of the allotted budget.

Future designs may improve on this design in the following ways:

1. Fix scaling issues in Power vs Frequency plot
2. Implement Waterfall and Max/Min plots
3. Implement user input functionality
4. Improve source code to enable Airspy
5. Develop distributed ground network protocols
Section 11
11.1 References


<https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm>


Section 12
12.1 Appendix

12.1.1 MATLAB Implementation of the FFT

Computing the discrete Fourier transform according to the formal definition is not efficient as it requires $O(N^2)$ arithmetic operations to be performed. The most commonly used fast Fourier transform algorithm, the Cooley-Tukey FFT algorithm, can reduce the required arithmetic operations to $O(N \log N)$ providing a significant performance advantage while producing the same results as the DFT.

% Implementation of a Fast Fourier Transform (FFT)

f1=20; % frequency 1 (Hz)
f2=70; % frequency 2 (Hz)
f3=35; % frequency 3 (Hz)

fs=1000; % sampling frequency (samples/second)
L=1; % length of signal (seconds)
Ts=1/fs; % sampling period (seconds)
t=[0:Ts:L]; % discrete time domain

x=0.2*cos(2*pi*f1*t)+0.15*cos(2*pi*f2*t)+0.3*cos(2*pi*f3*t); % sampled signal

N=length(x); % number of samples
sigma=0.1;
noise=sigma*randn(1,N);

x=x+noise;

plot(t,x)
title('Time Domain Signal Representation');
xlabel('t (s)');
ylabel('x(t)');

X=fft(x);
f=[0:N-1]/L; % discrete frequency domain

Xm=((2*abs(X)/N).^2)./2; % power spectrum
Xm=10*log10(1000*Xm); % convert to dBm

figure(2)
plot(f(1:(N-1)/2), Xm(1:(N-1)/2)) % plot half of the power spectrum

title('Fast Fourier Transform (FFT)');
xlabel('f (Hz)');
ylabel('Power (dBm)');
Figure 12.1: Time Domain Signal Representation and FFT of 3 sinusoids of frequencies 20, 35, and 70 Hz with amplitudes of 0.2, 0.15, 0.3 respectively in the presence of white Gaussian noise.

12.1.2 Radix-2 DIT

A radix-2 decimation-in-time (DIT) FFT is the simplest and most common form of the Cooley–Tukey algorithm, although highly optimized Cooley–Tukey implementations typically use other forms of the algorithm as described below. Radix-2 DIT divides a DFT of size N into two interleaved DFTs (hence the name "radix-2") of size N/2 with each recursive stage.

The discrete Fourier transform (DFT) is defined by the formula:

\[ X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i nk}{N}}, \]

where k is an integer ranging from 0 to N-1.

Radix-2 DIT first computes the DFTs of the even-indexed inputs (x_{2m}=x_0, x_2, \ldots, x_{N-2}) and of the odd-indexed inputs (x_{2m+1}=x_1, x_3, \ldots, x_{N-1}), and then combines those two results to produce the DFT of the whole sequence. This idea can then be performed recursively to reduce the overall runtime to O(N log N). This simplified form assumes that N is a
power of two; since the number of sample points N can usually be chosen freely by the application, this is often not an important restriction.

The Radix-2 DIT algorithm rearranges the DFT of the function \( x_n \) into two parts: a sum over the even-numbered indices \( n=2m \) and a sum over the odd-numbered indices \( n=2m+1 \):

\[
X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}
\]

One can factor a common multiplier \( e^{-\frac{2\pi i}{N}k} \) out of the second sum, as shown in the equation below. It is then clear that the two sums are the DFT of the even-indexed part \( x_{2m} \) and the DFT of odd-indexed part \( x_{2m+1} \) of the function \( x_n \). Denote the DFT of the Even-indexed inputs \( x_{2m} \) by \( E_k \) and the DFT of the Odd-indexed inputs \( x_{2m+1} \) by \( O_k \) and we obtain:

\[
X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{2N} mk} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}
\]

This result, expressing the DFT of length \( N \) recursively in terms of two DFTs of size \( N/2 \), is the core of the radix-2 DIT fast Fourier transform. The algorithm gains its speed by re-using the results of intermediate computations to compute multiple DFT outputs.

**Figure 12.2:** Data flow diagram for \( N=8 \): a DIT radix-2 FFT breaks an 8-point DFT into two 4-point DFTs followed by a combining stage consisting of many 2-point DFTs called “butterfly” operations.
[All information taken from
https://en.wikipedia.org/wiki/Cooley%E2%80%A9TukeyFFT_algorithm]