2011-04-27

A Practical Distributed Spectrum Sensing System

Devin WW Kelly
Worcester Polytechnic Institute

A Practical Distributed Spectrum Sensing System

by

Devin Kelly

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Electrical and Computer Engineering
by

May 2011

APPROVED:

Professor Alexander M. Wyglinski, Research Advisor

Dr. Andrew P. Worthen

Professor William R. Michalson

This work was sponsored by the Department of Defense Research and Engineering under contract number FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Abstract

As the demand for wireless communication systems grows, the need for spectrum grows accordingly. However, a large portion of the usable spectrum has already been exclusively licensed to various entities. This exclusive allocation method encourages spectrum to be left unused if the licensee has no need for that spectrum. In order to better utilize spectrum and formulate new approaches for greater spectrum use efficiency, it is imperative to possess a thorough understanding about how wireless spectrum behaves over time, frequency, and space. In this thesis, a practical, scalable, and low-cost wideband distributed spectrum sensing system is designed, implemented, and tested. The proposed system is made up of a collection of nodes that use general purpose, off-the-shelf computer hardware as well as a collection of inexpensive software-defined radio (SDR) equipment in order to collect and analyze spectrum data that varies across time, frequency, and space. The spectrum data the proposed system collects is the power present at a given frequency. The tools needed to analyze the gathered data are also created, including a periodogram and spectrogram function, which visualize average spectrum use over a period of time and as spectrum use varies with time, respectively. The proposed system also facilitates the testing of a spatio-spectrum characterization method using real data. This method has only been simulated up to this point. The characterization technique allows for spatially varying spectrum measurements to be visualized using heat maps.

Acknowledgements

I would like to acknowledge my advisors Professor Alexander M. Wyglinski and Dr. Andrew P. Worthen for their guidance and support. I also want to thank Professor William R. Michalson for serving on my committee.

Finally, I would like to acknowledge the financial support of MIT Lincoln Laboratory.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As demand for wireless communications increases, the strain on the availability of the electromagnetic spectrum progresses. This growing demand for wireless communications systems comes from various entities including: commercial, consumer, medical, public safety, industrial, and military. All of these organizations require spectrum, resulting in a strain on the current spectrum allocation system. With the current spectrum allocation rules, nearly all the usable electromagnetic spectrum is allocated. However, before one can address the problem of finding available spectrum for new communications systems the electromagnetic spectrum must be better understood.

Spectrum is neither completely used nor inappropriately allocated; this is a reflection of a misunderstanding of the relationship between spectrum allocation and spectrum use. To enable more efficient use of the spectrum, the allocation rules must change. The type and magnitude of these changes are unknown. Before one can change the rules regarding spectrum allocation, one must first have a complete understanding of the way the spectrum is used.

Understanding spectrum is a complex task; the spectrum is constantly changing by time and location. The tools to gather the data needed to understand this complex entity require special knowledge to operate. If not considering the difficulty of acquiring the data,

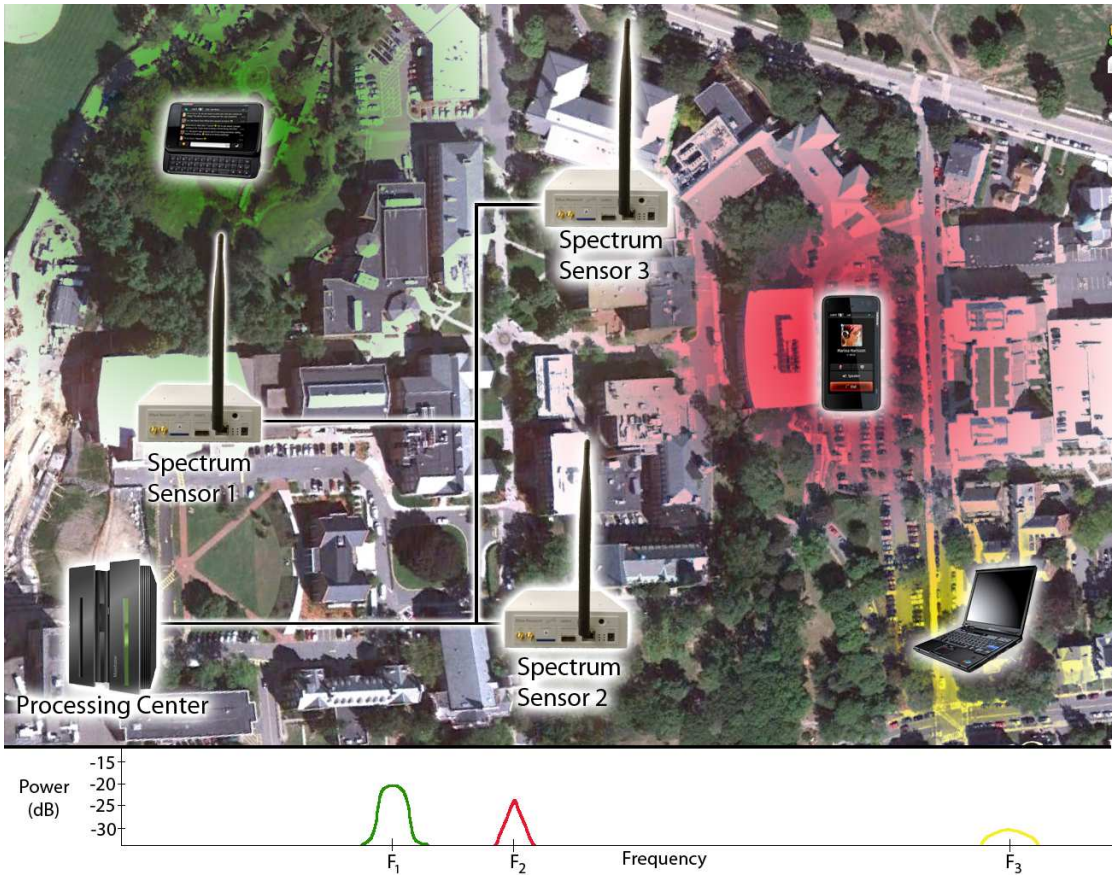


Figure 1.1: Concept Diagram

spectrum is still difficult to understand.

A necessity for a spectrum measurement system is a simple technique for gathering spectrum data. This technique must build on the shortcomings of the systems created and data that was gathered previously. Several attributes possessed by a spectrum sensing system include:

- **Practicality** Currently, spectrum sensors can be prohibitively expensive for research groups. A collection of inexpensive sensors can keep the total cost of a spectrum sensing system down. This system should be affordable for any research group.

- **Long Term Measurements** Spectrum sensing involves large amounts of data collected quickly, thus making long term measurements difficult. This system should provide the capability to measure spectrum over longer periods of time, such as weeks,

months, or years.

- **Wideband** The need for spectrum covers its entire usable range, thus data collection over as much of the spectrum as possible is necessary.
- **Scalable** Multiple sensors provide advantages over a single sensor, so this system should scale to as many sensors as the designer desires.
- **Data Sharing Mechanism** With a method to share data the spectrum sensors will be able to cooperate to form one conclusion about the state of the spectrum.
- **Consistent Data Storage Format** The system shall provide data in a consistent, compact format and provide the user with tools to analyze the data.

Such a system will enable a better understanding of the electromagnetic spectrum that will allow researchers to associate spectrum allocation with actual spectrum use. Figure 1.1 shows a concept diagram of a distributed spectrum sensing network. In this concept diagram there are three spectrum users, all of them operate on different bands. In the diagram there are also three spectrum sensors measuring the spectrum. The spectrum sensors all report their measurements to the processing center in the lower left hand corner. Finally, the processing center creates the spectrum measurement that is shown at the bottom of the figure.

1.2 Current State-of-the-Art

Software defined radios (SDRs), defined as the intersection between hardware radios and computer software [4], are the technological foundation for future spectrum sensing radios. An SDR is a radio that is software controllable, or a radio that can make decisions based on its environment. A spectrum sensor is a special type of SDR that attempts to characterize the radio frequency (RF) environment around it and makes decisions based on those characterizations. Depending on the system design, a spectrum sensor does not always need to be a coherent receiver, therefore a spectrum sensor does not necessarily need to decode transmissions into messages. The goal of the spectrum sensor is to characterize

the RF environment around it.

Currently, spectrum sensing is necessary for a variety of applications. These applications drive the need for spectrum sensing in the research, military, commercial and industrial communities. The applications that rely on spectrum sensing as a fundamental technology include dynamic spectrum access, interference avoidance, and spectrum characterization [5].

For dynamic spectrum access systems there are primary spectrum users and secondary spectrum users. The difference between the two is that primary users always have priority to spectrum access over the secondary users. Secondary users may access the spectrum but only when the primary users are idle. This limits primary user interference to accidental interference [6]. Spectrum that is unused by primary users is called vacant spectrum or a spectrum opportunity. A secondary user scans the spectrum looking for spectrum opportunities to communicate with other secondary users. Secondary users are not allowed to interfere with primary users, so they use spectrum sensing technologies to identify the presence or absence of primary users. Depending on the state of the primary user, the spectrum sensing radio has two options for communications. If a primary user is present, the communication can take place using spread spectrum technologies, where the secondary user transmits so that the power gets spread out over a large bandwidth [6]. The primary user will not be noticeably interfered with, since the power per unit frequency is low. The other option, called opportunistic spectrum access [7] occurs when a secondary user communicates normally in the absence of a primary user. Spectrum sensing in opportunistic spectrum access systems is necessary not only to identify unused spectrum, but also to identify when a primary user wants to reacquire that spectrum.

As part of the research applications, the measurements taken that characterized the RF environment were short-term and characterized a central geographical area [8, 9, 10]. The objective of the studies performed was to gain a more general understanding of how spectrum is used and occupied. The longest work measured different bands of spectrum over a period of several days, these studies all quantified short term spectrum use [8]. All of the studies also used instrument grade spectrum analyzers for data capture. Additionally, all of these studies measured spectrum at single points in space; meaning that geographic

dependent impairments can degrade some signals resulting in data that may not represent the nearby area well. The results of the spectrum survey studies all agree that spectrum is under utilized. For example, these studies show that some bands have a near zero percent occupancy rate [11].

All of the above studies measure spectrum at a single point in space. There is another study which implements a wideband distributed spectrum sensing system. Microsoft Research India has proposed a wideband distributed spectrum sensing system called SpecNet [12]. This system proposes a network of spectrum analyzers using Extensible Markup Language Remote Procedure Calls (XML-RPC). The proposed architecture is one where a master server sends out a command to slave servers (a PC networked with a spectrum analyzer) using the XML-RPC format, then waits for the measurement from each server. This spectrum sensing network is dependent on using instruments as a part of their network. Acquisition of multiple spectrum analyzers is expensive and the authors admit any practical system depends on volunteers.

The spectrum survey studies mentioned above had the common goal of characterizing spectrum in their respective environments. Spectrum has two basic states, which are used or unused. When characterizing spectrum the task of the sensor is to identify the current state of that spectrum. There are techniques to identify the state of observed spectrum, some of these technologies are energy detectors, matched filters, cyclostationary analysis, and waveform analysis [13]. An energy detector simply takes the frequency domain spectrum data and tests frequencies against a given power level [14]. A bank of matched filters associates the signal with one of the filters in the filter bank therefore deciding the state of the spectrum [15]. Cyclostationary detectors analyze the statistics of the signal to identify the state of the spectrum [16]. Waveform analysis attempts to identify certain predefined characteristics of a signal to identify spectrum state [17]. Spectrum sensors use any of the above methods to identify the state of the spectrum. However, matched filters, cyclostationary analysis, and waveform analysis can all make decisions regarding the signals occupying the spectrum. For example, matched filters and cyclostationary detectors can identify modulation scheme and waveform analysis can identify protocols [13].

One sensor or a collection of sensors working together can operate as a spectrum sensing

network. A single spectrum sensor has lower costs and a more simple implementation, but can only sense one propagation channel per transmitter. Sensing a singular propagation channel allows fading to impact the data that each sensor collects unequally compared to all transmitters. With distributed sensing, each sensor sees a different propagation channel. If the propagation channels are independent, then distributed sensing can overcome the fades that affect each propagation channel differently [18]. Distributed sensing also helps solve the hidden terminal problem [19], where the relative position the transmitter from the spectrum sensor.

When different spectrum sensors perform distributed sensing each sensor collects different data. This presents the problem of drawing one conclusion from the datasets that each node collects. One approach is gathering all data together to one centralized location and drawing a single conclusion from all of this different data respectively. Another approach is nodes drawing a conclusion then informing neighboring nodes of that conclusion, when there are no neighboring nodes left, the final conclusion is made [13].

A sensor-aware control channel can solve the problem of data collection or the control channel can be a wired data transmission medium. When using a wireless control channel, the amount of control data becomes an issue. Too much data is impractical and too little data will not allow the sensors to make proper conclusions about the spectrum environment. This trade-off is best illustrated by soft decisions and hard decisions respectively. A soft decision is when a decision device utilizes raw or slightly processed data and a hard decisions is when a decision device utilizes only a true or false statement about spectrum use at a particular frequency.

Finally, when considering distributed spectrum sensing there is data fusion. Data fusion a is technique in which multiple sensors combine their data to form a single conclusion. For soft decisions the techniques include voting algorithms, weighted averaging and Bayesian filtering. For hard decisions, `AND` and `OR` techniques have been proposed [20], which require a unanimous and a singular consensus respectively.

1.3 Thesis Contributions

This thesis contributes the follow to the wireless communications research community:

- A practical implementation of a distributed spectrum sensing system. This is a system that can characterize spectrum use. The characterization is far reaching in that a wideband of spectrum is characterized at once and that wideband is characterized for a long duration of time. This is all done with a low cost per node.
- Analysis tools for interpreting the data collected by this spectrum sensing system, including the periodogram, the spectrogram, and the PSD map visualizations. These tools allow researchers to to draw conclusions about spectrum. The periodogram allows researchers to make conclusions about spectrum based on a particular frequency they are interested in. The spectrogram allows researchers to see how spectrum use in that band changes with time. And finally, the PSD map allows researchers to visualize how the band they are interested in changes as a function of location. These three tools allow researchers to understand spectrum in three dimensions.

1.4 Thesis Organization

This thesis follows the following organization. Chapter 2 presents the background information needed to understand the topic of distributed spectrum sensing. Chapter 3 proposes the design of the spectrum sensing system. Chapter 4 describes the testing of the design proposed in Chapter 3. Finally, in chapter 5 research accomplishments and future work are discussed.

Chapter 2

Background

This chapter explains the background topics needed to understand the remaining chapters of this thesis. First, spectrum and the spectrum space are thoroughly defined. Then there is an examination of spectrum policy. Following this is a description of the requirements of a spectrum sensor. Afterwards, there is an explanation of spectrum sensing and spectrum sensing technique. Finally, there is a description of available spectrum sensing software and hardware.

2.1 Spectrum Use

Preceding an analysis of spectrum use, spectrum itself must be thoroughly defined. A definition of spectrum is not straightforward. The first definition of spectrum is in [21], where the first three domains are proposed. Following this was the addition of two more domains to the definition of spectrum [22]. These five domains, plus another, together make up the spectrum space. The spectrum space can also be referred to as the *electrospace* [23]. These domains are:

1. Time

   The temporal characteristics of spectrum use.

2. Frequency

   The frequency bands that spectrum users occupy.

3. Location (or geography)

   The physical space that a electromagnetic radiation propagates through.

4. Angle

   The beam angle (*e.g.* the angle of arrival) that a electromagnetic radiation propagates through.

5. Code

   The code space that a transmission occupies.

6. Antenna Polarization

   Antenna polarization is either horizontal, vertical or circular, *i.e.* a combination of the two.

These are the six domains of the electrospace that a spectrum user occupies. Figure 2.1 shows a digram of several signals occupying the electrospace in the frequency, time and spatial domains. A transmitter occupies all of these spaces simultaneously. An occupation of this space denies the use of the spectrum to other users. The domains that make up the spectrum space are also the domains that multiple access systems exploit. If two transmissions are orthogonal in any one of these spaces, then two different transmitter/receiver pairs can both access their respective transmissions without significant interference.

The Federal Communications Commission (FCC), as well as other governmental organizations around the world, have divided the spectrum space by frequency and geography. The FCC then issues a license to the desired licensee providing exclusive use for that spectrum. For example, two licensees in the same geographical region may be assigned licenses for two different frequency bands (*i.e.*, orthogonal in frequency) if they are to use spectrum simultaneously. Two licensees that are issued licenses to transmit at the same frequency may be in two different geographical regions (*i.e.* orthogonal in space). Presently, the division of the spectrum space is static. Spectrum space has only a single license allocated to it at a time. A licensee wastes their assigned spectrum if the licensee decides not to use the spectrum at that particular time. Wasted spectrum is called unused or vacant spectrum.

Spectrum is considered unused if either of the following criteria are met:

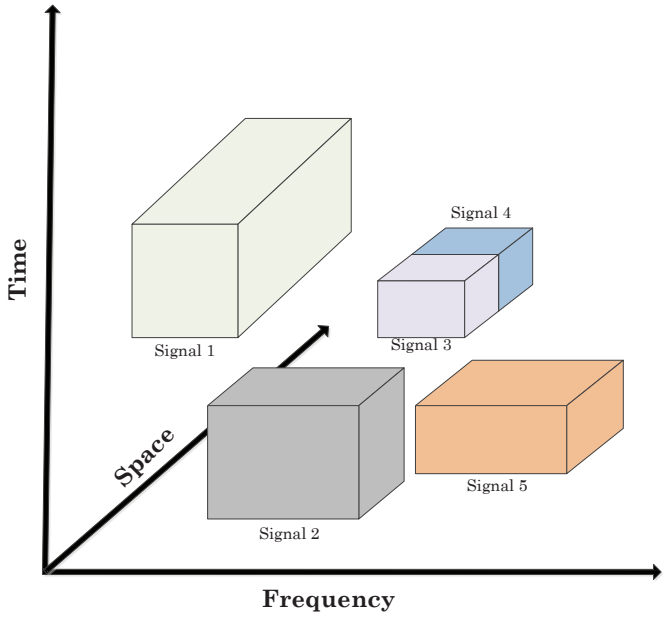


Figure 2.1: The Electrospace.

1. A transmitter/receiver pair can operate without being interfered with.

2. A transmitter can use spectrum without interfering with an unrelated receiver.

The differences between (1) and (2) above are subtle, but they become clear when considering the receivers. In (1), spectrum is unused when the intended receiver can receive a transmitted signal without interference. In (2), spectrum is unused when unintended receivers can receive their signals without being interfered with.

2.1.1 Spectrum Measurement

The combination of static allocation along with the licensees exclusive rights result in an underutilized spectrum space [8, 10, 9, 24, 25]. These studies all have similar characteristics: they are all wideband, conducted for short periods of time, and require expensive equipment. The spectrum captures in all but one of these studies lasted for short periods of time in that each spectrum capture lasted from tens of microseconds to tens of milliseconds. The study with the longest time lasted in hundreds of minutes. This measurement is a collection of

narrowband spectrum captures.

The conclusions of these studies are also similar. They all show an underutilization of spectrum at their respective locations. The sites where these studies were conducted include Chicago, Illinois, Singapore, Chicago, IL, USA, Los Angeles, CA, USA, and Paris, France. Measurements made in one study ranged from DC to 2.5 GHz. This study found that the highest occupancy rate being 31% [9]. In another two studies, measurements ranged from DC to 3 GHz and there was a 17.4% spectrum occupancy rate [8]. These two studies were both conducted in Chicago, Illinois. The longest measurement was on the scale of the length of a day, while all the other measurements lasted less than a second.

2.1.2 Increasing Spectrum Efficiency

All of these studies have concluded that the spectrum is inefficiently utilized. As a result of this, the wireless research community has proposed many techniques to improve spectrum efficiency. These techniques are collectively known as dynamic spectrum access (DSA) [13]. There are two subgroups of dynamic spectrum access: the *Dynamic Exclusive Use Model* and the *Hierarchical Use Model*. The dynamic exclusive use model involves temporarily reassigning spectrum to a secondary user, where as the hierarchical use model involves changing the regulations in which the secondary user may transmit in unlicensed spectrum.

Dynamic Exclusive Use Model

The dynamic exclusive use model has two subgroups of its own: spectrum property rights and dynamic spectrum allocation. The spectrum rights model would allow primary users to trade a certain amount of spectrum for a certain duration to secondary user [26]. Dynamic spectrum allocation allows for a secondary user to exploit gaps in spectrum use. The technique that allows for this exploitation to occur depends on the nature of the technology that the primary user uses [27]. For example, if the primary user transmits for 10 seconds once every 30 seconds, then the secondary user can reallocate that spectrum for the remaining 20 seconds.

Hierarchical Use Model

The hierarchical use model also has two of its own subgroups: spectrum underlay and

spectrum overlay. Spectrum underlay techniques involve using low power ultra wide-band signals to avoid interfering with a primary user [28]. A frequency domain plot of a spectrum underlay system is shown in Figure 2.2. Using this technique, a secondary users transmission remains under the primary receiver noise floor, thereby not interfering with the primary user. There are also spectrum overlay techniques for dynamic spectrum access. Using this technique, a secondary user scans spectrum looking for unused spectrum. When unused spectrum is found the secondary user may transmit while intermittently scanning the spectrum for the primary user [29]. A frequency domain plot of a spectrum overlay system is shown in Figure 2.3.

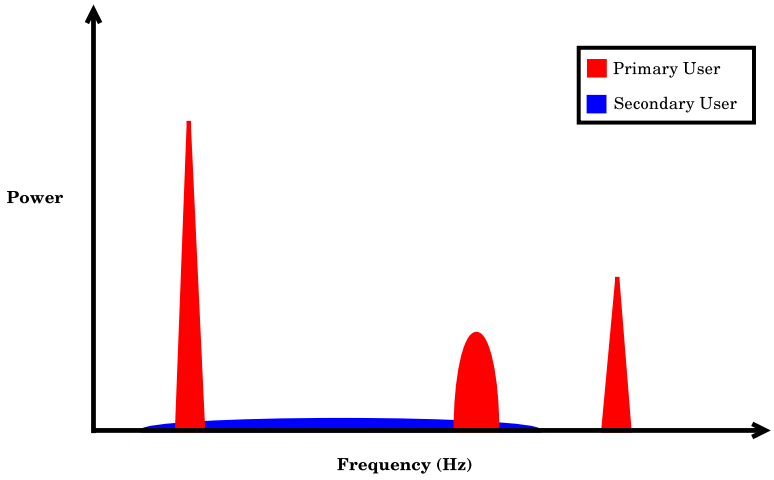


Figure 2.2: A Frequency Domain Representation of a Spectrum Underlay System.

When spectrum is accurately measured, the secondary user can choose the most appropriate secondary access system. For example, measurements in Amherst, MA, USA found that the availability of Wi-Fi and 3G networks are negatively correlated. That is, one communication system tends to be available when the other is not [30]. These measurements show that in this situation there are spectral holes that can be exploited by secondary access systems that vary by geography.

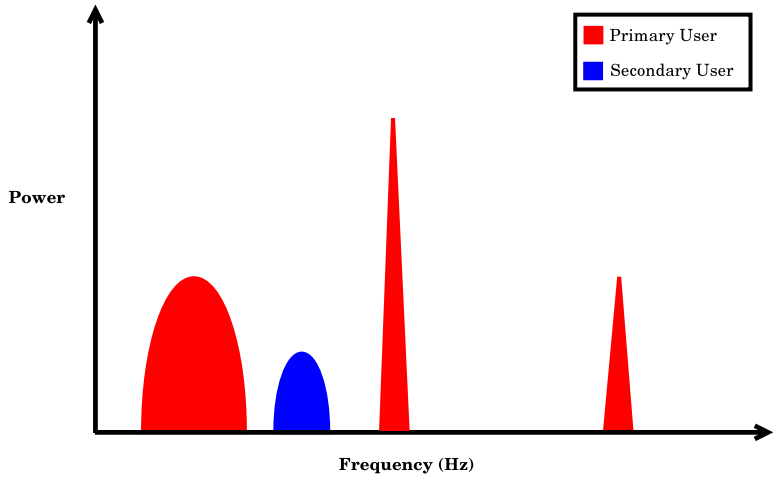


Figure 2.3: A Frequency Domain Representation of a Spectrum Overlay System.

2.2 Spectrum Policy

In order to increase spectrum efficiency, the proper regulations must be in effect. In the United States there are two bodies that regulate spectrum use: the Federal Communications Commission (FCC) and the National Telecommunications and Information Administration (NTIA). The NTIA is responsible for spectrum that is allocated for users that are members of the Federal government. The FCC is responsible for how the rest of the spectrum is allocated in the United States. The FCC and NTIA have developed the following model that applies to most allocated spectrum, which will be referred to as the *traditional model* [5]:

- Spectrum is divided by frequency.
- Each frequency band has a single entity that may use it. Examples of these entities include a wireless communications firm, public safety services, or the military.
- Generally, this entity has exclusive right to that spectrum. There are some non-exclusive licenses, these are in the minority.

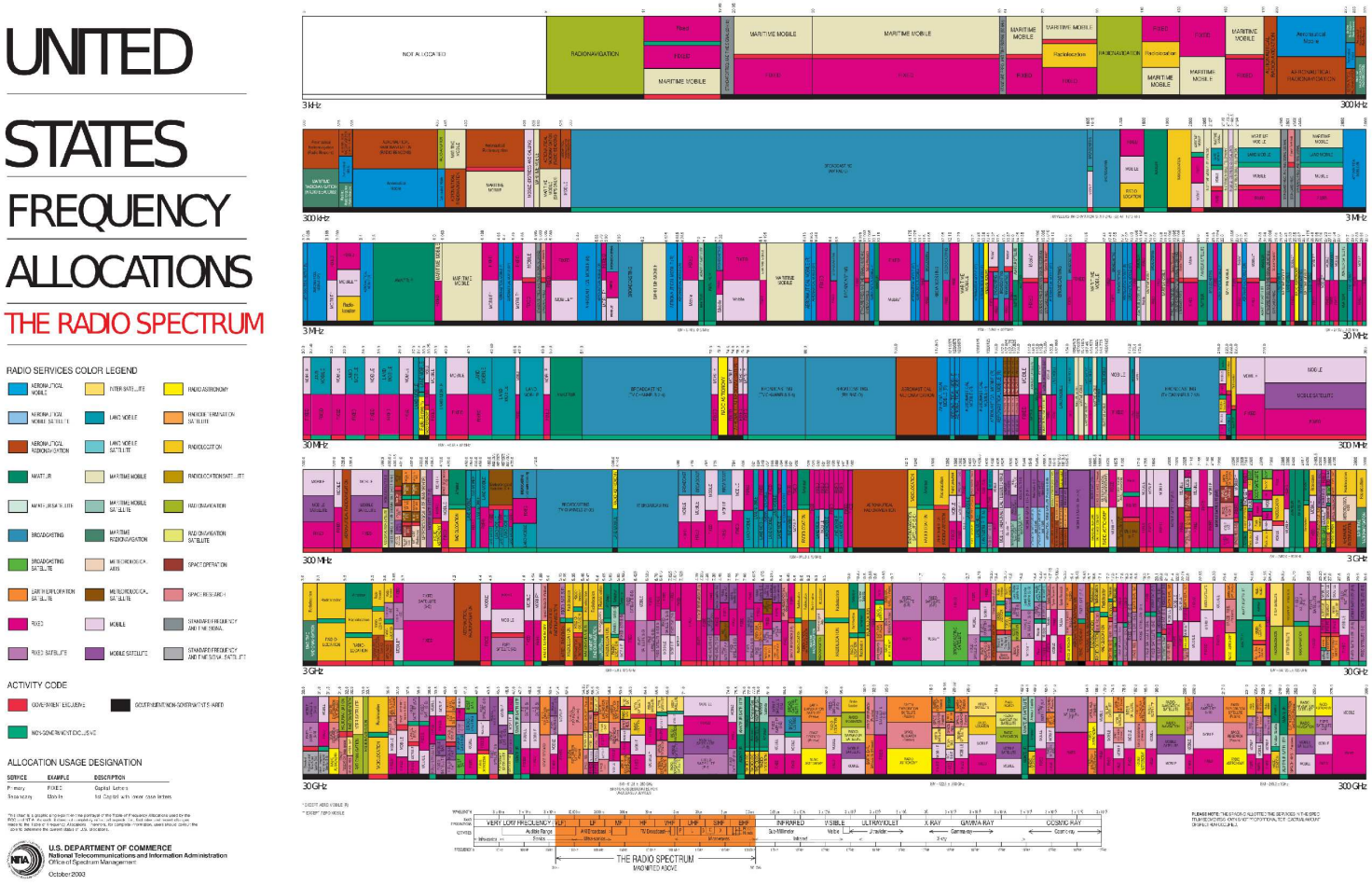


Figure 2.4: The United States Spectrum Allocation Chart [1].

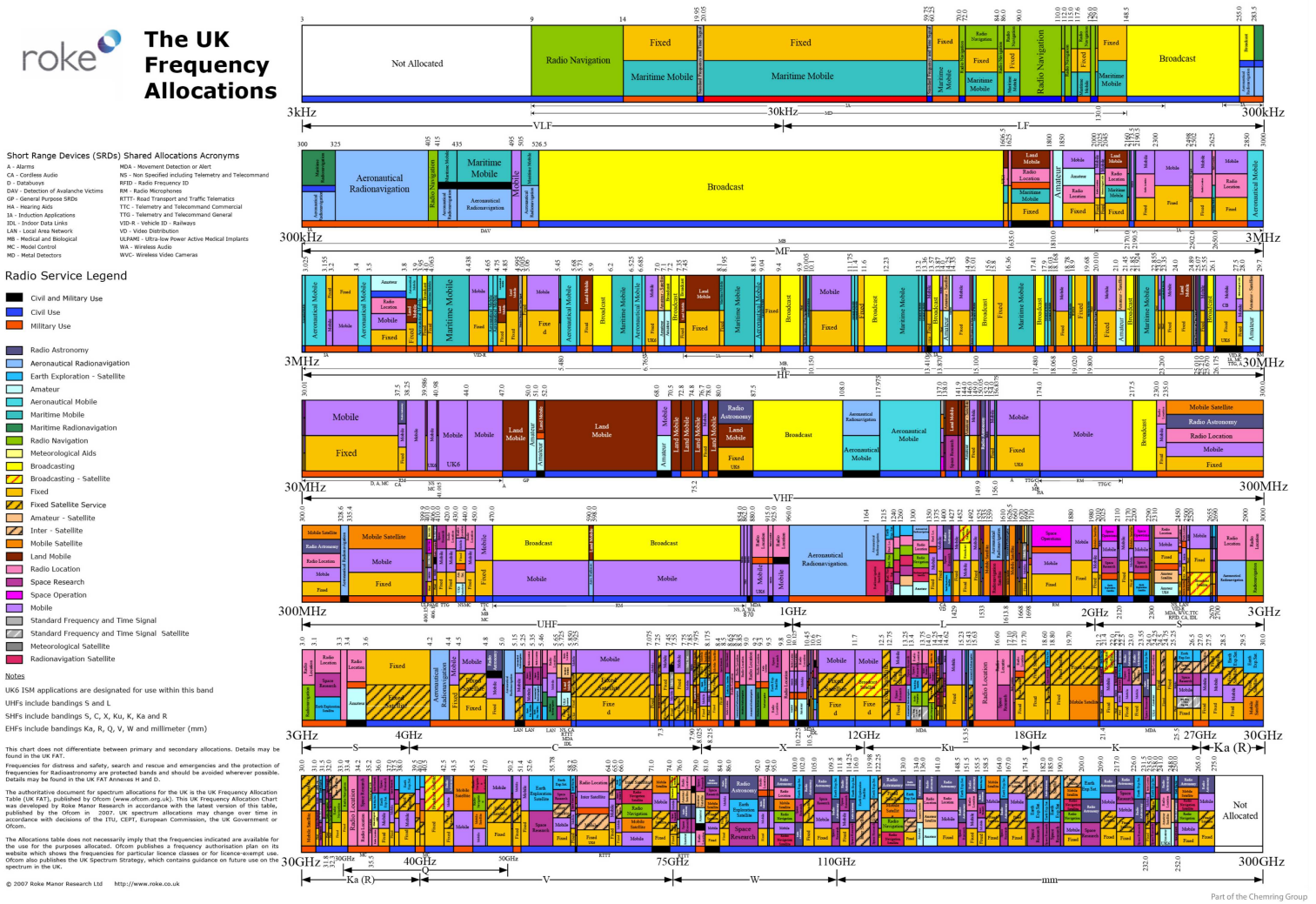


Figure 2.5: The United Kingdom Spectrum Allocation Chart [2].

The spectrum allocation chart for the United States is shown in Figure 2.4. The spectrum allocation chart for the United Kingdom is shown in Figure 2.5. As one can see both charts show an allocation of the electromagnetic spectrum that allows spectrum use for either single purpose or a small number of related purposes. A single purpose allocation is one where each band is allocation for only one use. In bands that are allocated for several purposes, they are related. Also notice that, although the entire spectrum is not allocated, the entire band that is of interest to this project is completely allocated.

As demand for more ubiquitous and higher bandwidth wireless communications grows the traditional model is becoming increasingly strained. As discussed in the previous section, the source of the strain on the spectrum is not spectrum use, but outdated spectrum allocation policies. The FCC Spectrum Policy Task Force (SPTF) found the spectrum to be drastically underutilized [31]. The realization that the spectrum is underutilized by the SPTF represent one of the first steps to changing the spectrum allocation system.

The FCC continues to take the steps needed in order to better utilize spectrum. These steps resulted in creating regulations that enable new technologies but also provide primary spectrum users with the exclusivity they are guaranteed. With the following rulings, the FCC has shown that it is willing to continue taking these steps:

- 1985 - The FCC modifies ISM bands to allow frequency hopping wireless communications [32].
- 2002 - The FCC allows ultra wide band (UWB) underlays [33].
- 2005 - The FCC allows cognitive underlays [34].
- 2010 - The FCC allows unlicensed operations in the TV broadcast bands [35].

Despite these steps forward, there are still many possible regulations to consider that will increase spectrum utilization. The regulations that are left to implement are significantly more complex than filling whitespace or allowing spectrum underlay systems. One such regulation is the prioritization of military spectrum access over commercial spectrum access. Prioritizing military spectrum access over commercial spectrum access is needed, but the details of the regulations that enable this prioritization are not clear. For example, the

military may not need as much spectrum access in areas that are distant from military bases. The military may also need expanded spectrum access during times of emergency. There are also other regulatory challenges such as radio propagation crossing international borders where the spectrum allocation conflicts. This means that countries with conflicting regulations would need to be addressed individually. There is also the problem of assuring that legacy systems can continue to operate unimpeded. The four techniques listed in above all assure this, but this set of regulations does not enable complete spectrum utilization.

Researchers have identified techniques that, if adopted, will allow for some of these regulatory concerns to be properly addressed [36]. These techniques are called "rules of entry." These techniques are all for cognitive radios (CRs), which is a radio that can make decisions based on its surrounding RF environment. These techniques, which are listed below, all allow for CRs to check the state of the spectrum before using it. When using these techniques CRs can reduce interference to primary users.

1. Spectrum Sensing

   The regulator must set the CR sensitivity requirement to determine if the band is empty.

   Concerns:

   - Vulnerable to the hidden terminal problem.

2. Geographical Databases

   The CR finds its location and compares to a database to see if it can transmit.

   Concerns:

   - Location accuracy.
   - Updating the database.
   - Database maintenance.
   - Database availability.

3. Beacon Reception

   A CR will find a beacon that indicates what frequencies are permissible to transmit

on.

Concerns:

- Beacon transmitter.
- Speed of updating the beacon.
- The spectrum the beacon uses.
- Receiving the beacon out of desired range.
- Beacon failures.

Each rule of entry has its own problems and concerns, and there is no obvious choice for a rule that allows secondary users to access spectrum.

2.3 Spectrum Sensing

Spectrum sensing is a technique used to characterize the occupancy state of the spectrum. Spectrum sensing is usually performed in the frequency domain, that is a spectrum sensor will hop across multiple frequency bands in some predetermined order testing for occupancy. Two measures of the performance of a spectrum sensor are the sensor's ability to locate unoccupied spectrum correctly without missing a primary user and the sensor's ability to not characterize noise as a user.

Spectrum sensing is needed to better utilize the electromagnetic spectrum and therefore facilitate the emergence of new wireless technologies. Nearly all spectrum is allocated already, but utilization remains low. Emerging wireless technologies are able to utilize vacant spectrum, but before this is possible a common set of rules regarding secondary spectrum use must be developed.

Before delving further into spectrum sensing technology there must first be a definition of some terms and mathematical frameworks. The goal of spectrum sensing is to determine the presence or absence of a primary user. In additive white Gaussian noise (AWGN) channels, a signal with a primary user transmitting (*i.e.* an occupied channel) is modeled as $y(t) = x(t) + n(t)$, where $x(t)$ is the primary users signal, $n(t)$ is AWGN, and $y(t)$ is

Table 2.1: A Table of the four possible decisions from 2.1.

Statement	Decision	Comment
$P(\mathcal{H}_0\|\mathcal{H}_0)$	Correct	Noise Identified
$P(\mathcal{H}_1\|\mathcal{H}_1)$	Correct	Signal Identified
$P(\mathcal{H}_1\|\mathcal{H}_0)$	Incorrect	False Alarm
$P(\mathcal{H}_0\|\mathcal{H}_1)$	Incorrect	Missed Detection

signal received by the spectrum sensor. This presents the spectrum sensor with a binary hypothesis as in Equation (2.1):

$$\begin{aligned} \mathcal{H}_0 &: y(t) = n(t) \\ \mathcal{H}_1 &: y(t) = n(t) + x(t) \end{aligned} \tag{2.1}$$

Where $\mathcal{H}_0$ represents an unoccupied channel and $\mathcal{H}_1$ represents an occupied channel. This binary hypothesis presents four possibilities, which are listed in Table 2.1. In this document, the probability of a missed detection will be denoted as $P(\mathcal{H}_0|\mathcal{H}_1) = P_{MD}$ and the probability of a false alarm will be denoted as $P(\mathcal{H}_1|\mathcal{H}_0) = P_{FA}$,

2.3.1 Spectrum Sensing Architectures

Spectrum sensing can provide an understanding of spectrum in the short term and the long term. The short term understanding of the spectrum can provide dynamic spectrum access users with the knowledge needed to operate these systems. A dynamic spectrum access system needs to be able to identify vacant spectrum in order to function. Once the spectrum sensor identifies vacant spectrum, the spectrum sensor can then pass that information on to other systems. For example, parameters such as center frequency and bandwidth are passed from the spectrum sensor to other parts of the dynamic spectrum access system.

A long term understanding of the spectrum can provide policy makers with the necessary knowledge to develop new policies that enable greater spectrum utilization. Long term spectrum survey not only helps identify spectrum that is used and unused, but also how it is used. Spectrum utilization varies in all the domains defined as part of spectrum: space, time, frequency, angle, polarization and code. Of these domains, time, frequency, and space

are the domains that are easily measured and easily exploited to gain efficiency. With long term spectrum studies that characterize these domains, policy makers can re-design policy to exploit spectrum use as defined by these measurements. For example, spectrum utilization could vary during different parts of the day, week, or year.

However, gathering this data on spectrum use is not straightforward, especially when gathering long term data. This is because sensing spectrum, even for a short period of time, requires handling large quantities of data at high processing rates. This requirement drives up the cost for spectrum sensors as well as the complexity of implementation. To assist with these challenges, one might consider two spectrum sensing architectures: centralized and distributed [37]. With centralized spectrum sensing, there is only one sensor per propagation channel whereas with distributed spectrum sensing many spectrum sensors may cooperate together.

2.3.2 Centralized Spectrum Sensing

Centralized spectrum sensing requires only a single sensor. The quantity of data the spectrum sensor sensed and stored is limited only by the hardware of the sensor. This makes implementing a centralized spectrum sensor comparatively easy, as the sensor's only task is to sense the spectrum and then act on that data appropriately.

However, a centralized spectrum sensor does have some drawbacks. A centralized sensor only senses a single propagation channel per transmission, so any local channel impairment could make the data inaccurate, therefore compromising that data. The centralized spectrum sensing technique is vulnerable to the hidden terminal problem. For example, the spectrum could be significantly different in two different locations (but still in the same region) because of a localized channel impairment.

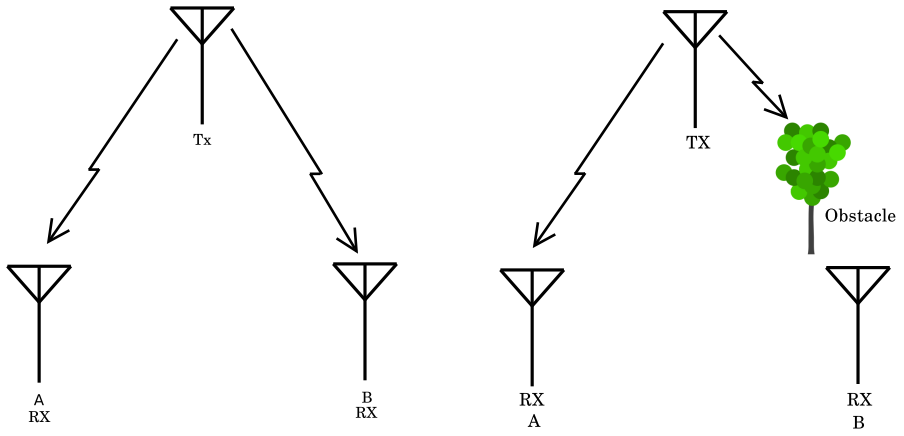
2.3.3 Distributed Spectrum Sensing

Centralized spectrum sensing systems have only a single sensor, while distributed spectrum sensing systems have multiple sensors. Systems with multiple sensors present interesting opportunities for spectrum sensing, increasing the implementation challenges. Distributed spectrum sensing can enable the collection data that varies by location. This

compensates for the primary disadvantage of having only a single sensor: localized impairments. However, there is a disadvantage to distributed spectrum sensing as well: sensor communication and data fusion.

With distributed sensing, the data that each sensor collects varies by geography and may vary by time depending on the accuracy of the sensors clocks. In Figure 2.6, sensors A and B data is only affected by the geography surrounding it, since the two sensors and the transmitter. In Figure 2.6(a) this is of little consequence, the two sensors are roughly equal distances away from the transmitter and there are no obstacles. However, in Figure 2.6(b) there is an obstacle between the transmitter and sensor B which causes shadowing to occur, therefore attenuating the signal and the data from that sensor.

Distributed spectrum sensing enables spectrum to be sensed from multiple channels.



(a) Spectrum Sensors with similar paths (b) Spectrum Sensors, one with an obstacle

Figure 2.6: Environment Affecting Spectrum Sensors.

Sensing from multiple channels allows the spectrum sensors to network together to circumvent channel impairments that are local to one sensor. One can keep the rate of missed detections, P_{MD}, arbitrarily low only by adding sensors while keeping the rate of false alarm, P_{FA}, constant [38].

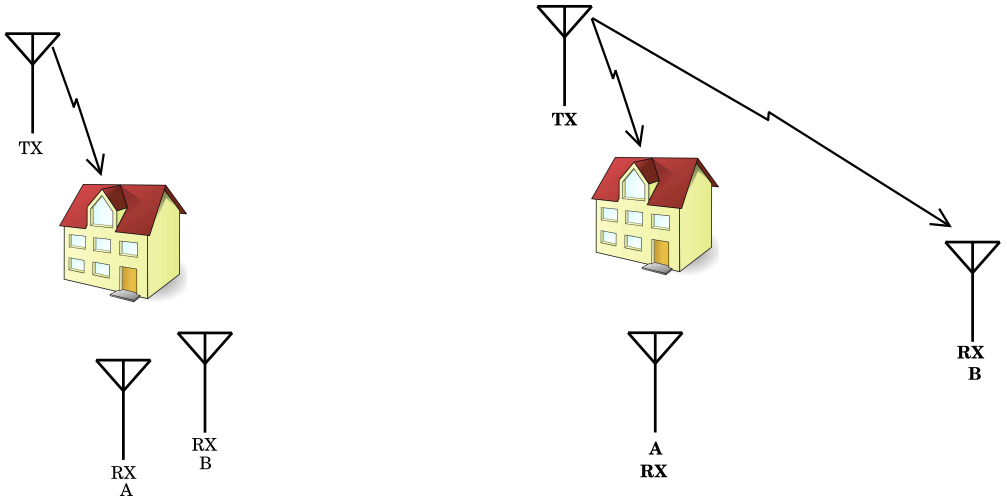
Improving performance is not always as straightforward as adding sensors. More sensors

does not necessarily mean that spectrum sensing capabilities will increase. When additional sensors experience correlated shadowing or fading, spectrum sensing performance does not improve [39]. For spectrum sensing performance to improve, the channel that each sensor senses must be independent from all the other channels. Sensors that are separated by a greater distance are more likely to observe independent channels. Fewer sensors that are more spread out in a geographical space are more effective than a larger number of sensors condensed into a small space [18].

The environment that the sensor network is deployed in determines the density of the sensor distributed. For example, in urban environments there are more obstacles between transmitters and spectrum sensors (*e.g.* cars, buildings) than rural environments. The number of obstacles affects the independence of a channel when the position of two sensors remains constant. One can see this in Figure 2.7(a), where the spectrum sensors experience the same shadowing effect. The signals these sensors received have correlated shadowing. This means that the channel for each sensor, with respect to the transmitter, are dependent and therefore having multiple sensors provides no significant advantage. In Figure 2.7(b), sensor A is shadowed and sensor B has a direct path to the transmitter. In this scenario, the channel for each sensor is independent and having multiple sensors provides a significant advantage for the spectrum sensing system.

Adding sensors to a distributed spectrum sensing network can improve performance by increasing sensitivity as well as negating the effects of shadowing and fading. As discussed above, spectrum sensors must be more sensitive than primary users. In order to avoid the hidden terminal problem, spectrum sensors must be able to detect primary user transmissions from a distance further than a primary receiver is able to receive. However, in distributed spectrum sensing networks, individual spectrum sensors do not necessarily need to have better sensitivity than a primary receiver. As spectrum sensors are added to a network, the sensitivity requirements of individual spectrum sensors becomes increasingly relaxed [38].

Distributed spectrum sensing systems present challenges that centralized systems do not. The foremost of these additional challenges is combining all the data that these spectrum sensors receive into one conclusion. One can break down this problem into two categories:



(a) Each sensor experiencing similar shadowing

(b) Only one sensor experiencing shadowing

Figure 2.7: Two Different Spectrum Sensing Environments.

the first being how to share the data collected and the second being how to combine the data such that the correct conclusion is made about the spectrum.

Data Sharing

Any distributed system can have a variety of architectures:

1. A Single Master Node:

   All spectrum sensors report their data directly to this node.

2. Several Master Nodes:

   Each master node receives a copy of all the data.

3. All Nodes Equal:

   Every node receives a copy the data from every other node at some processing stage. The data may have some or all the processing done before it is shared.

Each one of these data sharing architectures provides different advantages and disadvantages. These system architectures are of increasing robustness, that is, systems with fewer points of failure are more robust. These systems also require increasing volume of data

sharing. That is, these systems require more communication overhead.

Each data sharing technique requires an overhead channel. An overhead channel is any channel in which spectrum sensors can communicate with one another. There are several basic overhead channels: offline, wired and wireless channels. An offline overhead channel is when each sensor stores the data it collects locally and then the data is collected after the experiment takes place. A wired overhead channel uses an existing wired network to share data. A wireless overhead channel is when the nodes share data using wireless networks. When offline or wired channels are available, they are ideal as they do not interfere with spectrum sensing and they allow sensors to share more information. Wireless channels present some additional challenges, mainly that of handling the spectrum sensors transmitting. This can be handled in several ways: first, the system designer can assume that the spectrum sensors are as much part of the environment as all the other receivers and count them all the same. The system designer can also decide that spectrum sensor overhead should not count as part of the spectrum measurement and somehow avoid including the overhead. This becomes complicated because data needs to be exchanged between sensors while avoiding interference with the spectrum users that are being sensed. If the spectrum sensors are synchronized, they can avoid interfering by sensing other parts of the spectrum when overhead communications take place.

When one examines the data that is shared between nodes, this trade-off becomes less straightforward. The data that nodes share can be data ranges in complexity from directly sampled spectrum to conclusions about a particularly bands occupancy. Overhead channel capacity is usually limited, so sharing directly sampled spectrum is unrealistic: the data must be first be processed before it can be shared. There are two types of decisions that a spectrum sensor can render: *soft decisions* and *hard decisions*. Soft decisions are intermediate data, such as samples of an FFT, that a sensor will pass on to a decision making device. Hard decisions are the conclusion of that spectrum sensor, *e.g.* "occupied" or "not occupied". There is another trade-off with hard and soft decisions: soft decisions lead to more accurate conclusions, but require more overhead channel capacity, and hard decisions lead to less accurate conclusions, but require less overhead channel capacity [40].

There is another complication to this set of trade-offs, where hard decisions can provide

conclusions with equal confidence of soft decisions. For hard decisions to be as accurate as soft decisions, the right conditions must exist. First, the sensors must be spaced far enough apart so that they sense independent channels. Second, the more sensors that work together, the more reliable a conclusion based on hard decisions becomes for sensors in the similar locations [41].

In order to develop the most appropriate spectrum sensing system the system designer must know the overhead channel capacity, the needed system robustness and level of system accuracy. Only with all of these specifications set can data be shared effectively.

Data Fusion

Data fusion is an approach used by the decision device to combine the decisions of the spectrum sensors into one conclusion. The specific method used to combine information varies depending on whether hard decision or soft decisions are being combined. For hard decisions there are two decision rules, which are measured by the probability of detection and the probability of false alarm. These probabilities correspond to $P_{MD} = P(\mathcal{H}_0|\mathcal{H}_1)$ and $P_{FA} = P(\mathcal{H}_1|\mathcal{H}_0)$ from Table 2.1. The two rules are:

1. AND Rule:

   The decision device concludes that a primary user is present if all the spectrum sensors make that decision. The probability of detection for the AND Rule is:

   $$P_D = \prod_{n=1}^{N} P_{D,n} \tag{2.2}$$

   The probability of false alarm is:

   $$P_{FA} = \prod_{n=1}^{N} P_{FA,n} \tag{2.3}$$

2. OR Rule:

   The decision device concludes that a primary user is present if any of the spectrum sensors make that decision. The probability of detection for the OR Rule is:

   $$P_D = 1 - \prod_{n=1}^{N} 1 - P_{d,n} \tag{2.4}$$

The probability of false alarm is:

$$P_{FA} = 1 - \prod_{n=1}^{N} 1 - P_{FA,n} \tag{2.5}$$

Here, P_D is the probability of detection, P_{FA} is the probability of false alarm, n is each individual sensor, N is the total number of sensors, $P_{D,n}$ is the probability of detection for each sensor, and the $P_{FA,n}$ is the probability of false alarm for an individual spectrum sensor. In all cases, the probability of missed detection is $P_{MD} = 1 - P_D$. The AND Rule provides better performance when measuring by the false alarm rate and the OR Rule provides better performance when measuring by detection rate [42]. The data fusion algorithms for soft decisions are more complex and depend upon the specific detection algorithm implemented.

Hard decisions yield a binary conclusion, where one hypothesis is correct and the other is incorrect. However, soft decisions do not come to a binary conclusion. A common soft decision test is the likelihood ratio test, where the ratio of the probability densities of each hypothesis are compared. For example, if one wishes to test the distribution of a random variable one may use the likelihood ratio test as shown in Equation (2.6) [43]. A central node may collect the likelihood ratio, $L(y)$, from all nodes then combine then, by finding the mean for example, and then comparing to a threshold to find a final decision.

$$\begin{aligned} &\mathcal{H}_0 : Y \sim f_{y1}(y) \\ &\mathcal{H}_1 : Y \sim f_{y2}(y) \\ &L(y) = \tfrac{f_{y1}(y)}{f_{y2}(y)} \end{aligned} \tag{2.6}$$

2.3.4 A Comparison of Distributed and Centralized Spectrum Sensing

Centralized and distributed spectrum sensing have a set of trade-offs that will be compared in this section. In Table 2.2, the trade-offs between centralized and distributed spectrum sensing are listed.

A distributed spectrum sensing system can detect and correct environments with fading or shadowing. Each sensor senses a different channel if the sensors are placed correctly in the environment. However, there is an increased complexity of implementation with a distributed spectrum sensing network. An overhead channel is needed to coordinate all the

Table 2.2: A comparison of centralized and distributed spectrum sensing techniques.

Topic	Centralized	Decentralized
Fading and Shadowing	Cannot detect or correct	Can detect or correct
Implementation Complexity	Low	High
Overhead Channel	Unneeded	Needed
Sensitivity Requirements	High	Low
Scalability	None	Can Scale

sensors, thus adding complexity to an implementation. Additionally, a data fusion system must be implemented with a distributed spectrum sensing system. Another advantage of a distributed spectrum sensing system is that when multiple sensors are used together, the sensitivity needed for each sensor is reduced. Finally, with a distributed spectrum sensing system researchers can add as many sensors as they like.

The only trade-off not addressed in preceding sections is the scalability trade-off. For a centralized spectrum sensing system, the system inherently cannot scale as there is only one node by definition. Distributed systems are scalable as they can be made up of as many nodes as the overhead channel allows.

2.4 Spectrum Sensor Requirements

In order to detect primary users transmitting at particular frequencies at particular times a spectrum sensor must meet certain requirements. Detecting users also means finding any user that is receiving a transmission as well. Detecting spectrum users in the primary goal of the spectrum sensor. Classification of spectrum users is a secondary goal of spectrum sensors. Classification of spectrum users means to identify a particular characteristic beyond what is given by detection. For example, identifying physical layer parameters, such as modulation scheme, is classification.

In order for any spectrum sensing system to properly detect and classify primary users, the system must meet certain requirements. These requirements are dictated by the FCC and must be met in order to ensure that a primary user is detected and classified. In dynamic spectrum access scenarios, this is especially important as a missed detection of

a primary user and resulting transmission by a secondary user could lead to interference. The heightened motivation to detect and classify primary users in dynamic spectrum access systems displays the requirements needed to confidently identify primary users without missed detection for any spectrum sensing system.

The difficulty in identifying primary users is explained by the hidden terminal problem.

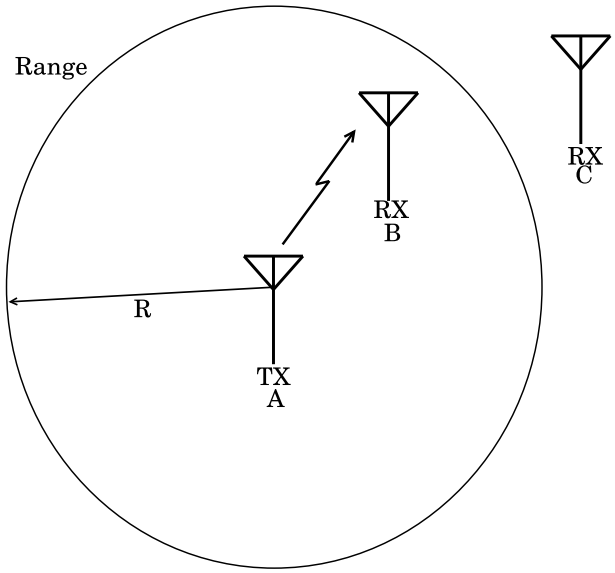


Figure 2.8: The Hidden Terminal Problem.

As shown in Figure 2.8, the hidden terminal problem is when a receiver is out of the range of a transmitter. In Figure 2.8, Radio A is transmitting to Radio B. Both Radios A and B are primary users. The circle of radius R indicates the area in which a primary receiver that meets the sensitivity specification can successfully receive the transmission from Radio A. Radio C is a spectrum sensor that, if it only meets the sensitivity requirements for the primary user, cannot detect the transmission. In dynamic spectrum access systems, the spectrum sensor can determine that the spectrum is vacant and then transmit. If this were to happen, then the reception from Radio B would be interfered with. This is the hidden terminal problem, when a spectrum sensor falsely comes to the conclusion that the spectrum is vacant when the spectrum is occupied [44].

In order for a spectrum sensing system to be effective, it must account for the hidden terminal problem. If a spectrum sensing system is made more sensitive than a primary receiver it will be more effective at detecting and classifying primary transmitters. In Figure 2.8, increasing the sensitivity of the spectrum sensor (Radio C) increases the radius R from the perspective of Radio C.

2.5 Spectrum Sensing Techniques

This section will discuss the RF and signal processing techniques used to perform spectrum sensing. RF and signal processing techniques are a collection of subsystems and algorithms that facilitate data gathering.

The RF subsystem in any software defined radio is static, such that once it is constructed it remains in a static state. A generic RF subsystem is shown in Figure 2.9 [45]. In this figure there are several components, the first of which is the antenna. The band pass filter is used to attenuate undesired frequencies and prevent the low noise amplifier (LNA) from saturating. The LNA is then used to amplify the signal so it above the noise floor. Then there is a mixer which is used to downconvert the signal from passband directly to baseband. The automatic gain control is then used to utilize the full dynamic range of the analog to digital converter. The analog to digital converter samples and quantizes the signal.

Following the RF subsystem is the signal processing chain, which can be implemented

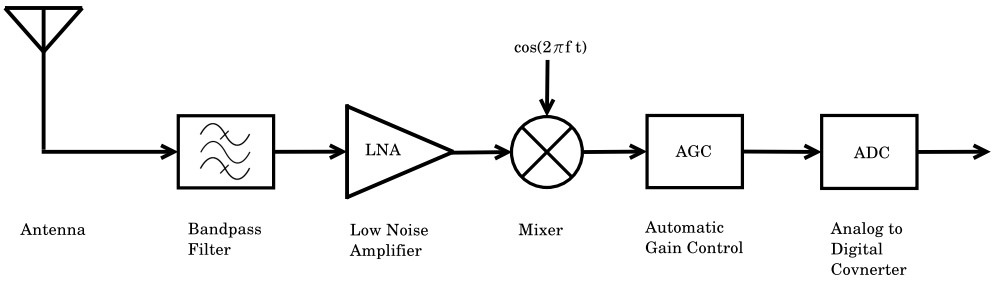


Figure 2.9: A Generic RF Receive Chain.

in software. Since the signal processing chain can be implemented in software, there are a variety of spectrum sensing techniques that can easily be implemented and tested. The

function of the signal processing chain is to detect and sometimes identify signals. The three most popular techniques are energy detection, matched filtering, and cyclostationary analysis.

These three algorithms are measured by several metrics. The first metric is the information needed to detect and/or classify a signal, as some algorithms need prior information about a signal before it can be detected or classified. There are also the metrics of probability of missed detection, P_{MD}, and probability of false alarm P_{FA}. These two probabilities compliment one another and can be graphed against one another to form a receiver operating characteristic (ROC). Another metric is computational complexity, or how efficient the algorithm is. Finally, there is implementation complexity, or how difficult it is to understand and implement the algorithm.

2.5.1 Spectral Estimation Techniques

Before performing spectrum sensing, one must find the spectrum of a signal. To find the spectrum of a signal the power spectral density (PSD) is needed. The power spectral density is a representation of a signal that shows the signal power present as a function of the frequency of the signal. For stochastic signals, the PSD is a purely theoretical entity. In other words one cannot calculate the PSD of a stochastic signal that is measured. To calculate the PSD of a signal one must have a complete representation of the time domain information of that signal, that is one must have knowledge of the signal for all time [15]. Since one cannot record any signal for infinite time one cannot find the PSD. However, one can estimate the PSD using a finite record of a signal, called the periodogram.

Periodogram

To estimate the PSD of a signal, one shall use the Khintchine-Weiner Theorem (or the sometimes Einstein-Khintchine-Weiner Theorem) as shown in Equation (2.7) [15].

$$S_{xx}(f) = \sum_{k=-\infty}^{\infty} r_{xx}(k) \cdot e^{-j2\pi kf} \tag{2.7}$$

The Khintchine-Weiner theorem states that the Fourier Transform of the autocorrelation function, $r_{xx}(k)$, equals the PSD, $S_{xx}(f)$. Note that the autocorrelation function, $r_{xx}(k)$, only depends on the delay, k, between the signal with itself. When the autocorrelation

function only depends on a delay, k, the signal $x[n]$ must be wide sense stationary. Wide sense stationarity occurs when the mean and variance are constant with respect to time. The autocorrelation function is also an infinite length sequence. As a result, one must find an estimate of the autocorrelation function, as shown in Equation (2.8). In Equation (2.8), $\hat{r}_{xx}(m)$ is the autocorrelation estimate, N is the signal length (in samples), and * is the complex conjugate operator:

$$\hat{r}_{xx}(m) = \frac{1}{N} \sum_{n=0}^{N-m-1} x^*(n) \cdot x(n+m) \tag{2.8}$$

Using Equations (2.8) and (2.7) one is now ready to estimate the PSD of a finite length signal of length N, with an autocorrelation function of length $2N-1$. This estimate, $P_{xx}(f)$ is called the periodogram [15]:

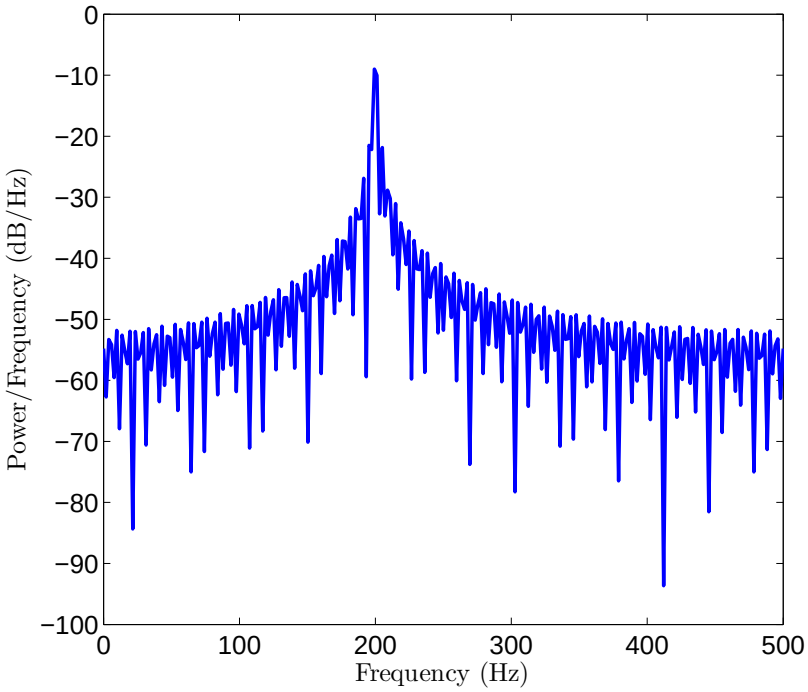
$$P_{xx}(f) = \sum_{k=-(N-1)}^{N-1} \hat{r}_{xx}(k) \cdot e^{-j2\pi kf} \tag{2.9}$$

$$= \sum_{k=-(N-1)}^{N-1} \left(\frac{1}{N} \sum_{n=0}^{N-k-1} x^*(n)x(n+k) \right) e^{-j2\pi kf} \tag{2.10}$$

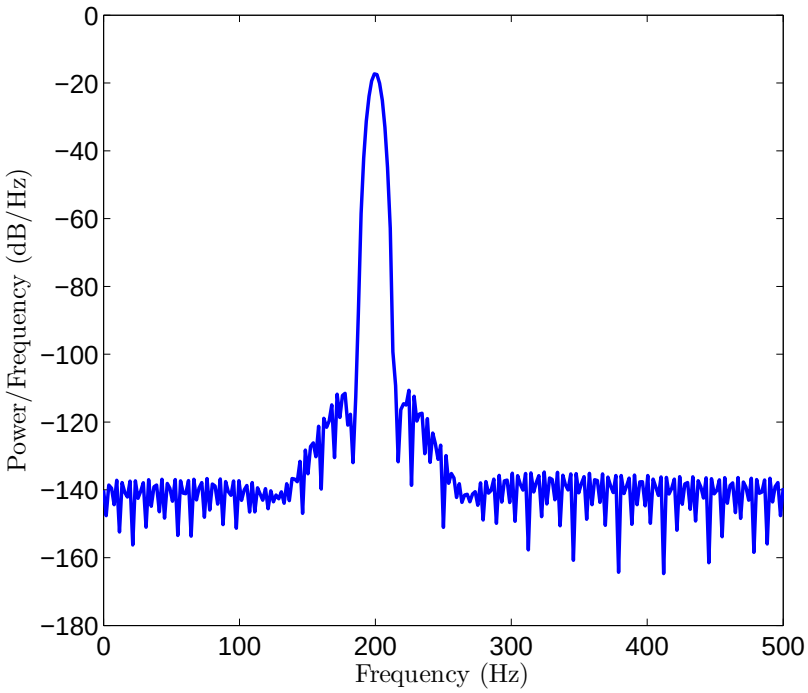
The periodogram is a flawed estimate of the PSD for several reasons:

- The periodogram does not approach the PSD as the number of samples approach infinity [46].
- The periodogram misrepresents the PSD by 'leaking' signal energy from one frequency into neighboring frequencies.

The misrepresentation of the PSD by leaking energy into neighboring bins is called 'spectral leakage,' and is the result of windowing a signal. A window is a function in which a signal is multiplied by that has a non-zero value for the duration of the signal and zero valued everywhere else. Thus, when the PSD is estimated using a finite length signal it is equivalent to multiplying that finite length signal by a rectangular window. A rectangular window is a window that equals one for the signal duration and zero everywhere else as shown in Equation (2.11). The frequency domain representation of the rectangular window is infinite in length as shown in Equation (2.12). Since the window function is multiplied



(a) An Example Periodogram of a Cosine Signal with a Rectangular Window



(b) An Example Periodogram of a Cosine Signal with a Blackman-Harris Window

Figure 2.10: Periodogram Examples.

by the signal in the time domain, the two representations of the signals are convolved in the frequency domain. This convolution is responsible for smearing the spectrum into neighboring frequencies, which results in spectral leakage. Two example periodograms of a 200 Hz cosine wave are shown in Figure 2.10. In Figure 2.10(a) a rectangular window was applied to the signal. In Figure 2.10(b) a Blackman-Harris window is applied to the signal. Observe that in Figure 2.10(a) the cosine signal is more narrow than in Figure 2.10(b), but the side-lobes are much greater.

$$w(n) = \begin{cases} 1 & 0 \leq \text{n} \leq \text{N-1} \\ 0 & \text{Otherwise} \end{cases} \tag{2.11}$$

$$W(\omega) = \frac{\sin((\omega/2)N)}{\omega/2} e^{-j\omega(N-1)/2} \tag{2.12}$$

The rectangular window, as defined in Equations (2.11) and (2.12), is not the only window one can choose for their spectral analysis. There are several other windows designed to reduce spectral leakage. However, the windows that reduce spectral leakage often also reduce spectral resolution, resulting in less distinct spectra. For example, two separate signals that are close to each other in frequency can be merged together into one signal if the window has large side lobes, such as the rectangular window.

Spectrogram

The spectrogram is a representation of the frequency content of a signal changing with time. The spectrogram is a function of both frequency and time; so therefore it is simply a collection of periodograms that are arranged to form an image. For spectrograms, the horizontal axis is for frequency, the vertical axis is for time, and the color represents the power of the signal for that duration of time and frequency. An example spectrogram of a linear chirp signal is shown in Figure 2.11. A linear chirp signal is a sine wave that increases its frequency linearly as a function of time, for example $x(t) = sin(2\pi(f_0 + t)t)$. It is important to note that the spectrogram shows the power of a signal at a certain frequency for a certain duration of time, that is the spectrogram cannot be used to find signal power at a given time instant. The spectrogram is useful for observing how spectrum changes as a function of time.

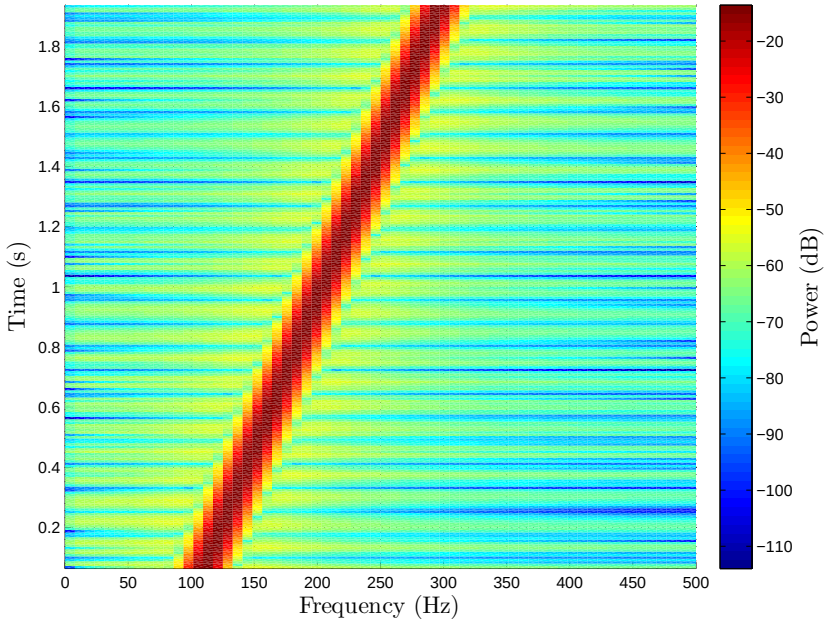


Figure 2.11: An Example Spectrogram of a Chirp Signal.

2.5.2 Energy Detection

An energy detector is a signal processing algorithm that detects spectrum use by measuring energy of a signal for a certain period of time. An energy detector needs no prior information above the signal it tries to find. The strategy of an energy detector is to develop a test statistic and compare the test statistic to a threshold. If the statistic is greater than the threshold, it is concluded that a signal is present. If the test statistic is less than the threshold, then it is concluded that a signal is not present.

Energy detectors use a three step process to find the test statistic. First, there is a band pass filter to eliminate the uninteresting frequencies. Second, there is a square law device, to make the entire signal positive. Third there is an integrator, to sum the signal for a predetermined period of time, T. Figure 2.12 shows the signal processing chain for an energy detector. There are several factors that influence the performance of an energy detector the bandwidth of the band pass filter, the threshold that the test statistic is compared to, and the time of integration, T. Manipulating the bandwidth introduces the trade-off that

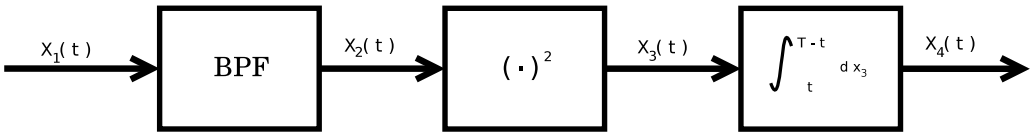


Figure 2.12: An Energy Detector.

a larger pass band makes the energy detector more susceptible to fading, but a smaller passband allows for smaller bandwidth signals to be detected. Manipulating the threshold changes the false alarm and missed detection rates. A threshold too low increases the false alarm rate and decreases the missed detection rate. A threshold too high decreases the false alarm rate and increases the missed detection rate. Manipulating the time of integration also effects the false alarm and missed detection rates. A longer time of integration yields a lower false alarm and lower missed detection rates.

Energy detectors are most effective at detecting signals in AWGN channels with only flat fading across the band of interest. If there is frequency selective fading on the band of interest, the effects are similar to having the band pass filters bandwidth being too large. In these channels P_{MD} and P_{FA} are lowest. Energy detectors can also be effective in environments with Rayleigh and Rician fading [47], although environments with this type of fading are not ideal. Energy detectors do not need to have knowledge of the symbol timing of transmitted signals and thus are noncoherent detectors. A noncoherent detector allows for low implementation and computational complexities. The trade-off for this is that energy detectors can only detect signals, they cannot classify signals.

2.5.3 Matched Filter

The matched filtering technique involves creating a bank of matched filters and then choosing the best filter from the filter bank. Matched filters require prior information about the signal that is being tested. This information includes modulation scheme and symbol timing, thus a matched filter detector is a coherent detector. A matched filter bank is a collection of receivers where the signal is passed through every receiver. Every signal is filtered, decoded, and demodulated as if the signal belongs to each filter. Since the signal only belongs to one filter, each filter develops a test statistic. Each test statistic is then

compared to find which filter can correctly receive the signal. The test statistic, z_k, is described by Equation (2.13). Each matched filter generates its own test statistic, so if there are k filters then there are k test statistics. In Equation (2.13), h is the unknown gain of channel, $s(n)$ is a known pilot sequence, $n(n)$ is additive white Gaussian noise, $y(n) = h \cdot s(n) + n(n)$ is received signal, and n is the index [36]. In Equation (2.13), * is the complex conjugate operator:

$$z_k = \frac{1}{N} \sum_{n=1}^{N} y(n) \cdot s^*(n) \tag{2.13}$$

When this information is known, matched filtering is the optimal technique for receiving signals in AWGN channels [15]. If the information is not known, then the matched filter may not be able to detect or classify the signal at all. If the matched filter has prior information about the signal it can detect and classify that signal. Due to the need for coherency, matched filters require full reception of the received signals for each filter chain thus increasing the computational and implementation complexities.

2.5.4 Cyclostationary Detector

Cyclostationary detectors analyze certain statistical properties of signals in order to detect and classify them. The signal processing techniques used to make communications signals (such as sampling, modulation, and coding) introduce periodicities into these signals, making the statistics of these signals (such as the mean or variance) periodic. A signal with periodic statistics is defined as a cyclostationary signal. Noise is a stationary random process, where the mean and variance are constant.

A signal is considered cyclostationary in the strict sense if the Nth order distribution function is periodic with period T, as shown in Equation (2.14). A signal is cyclostationary in the wide sense if only the mean and autocorrelation are periodic with period T, as in Equation (2.15). Where E[$\cdot$] is the expectation operator and $\mathrm{R}_x(\cdot)$ is the autocorrelation of $x(\cdot)$ [48].

$$F_{x(t+\tau_1+T)\ldots x(t+\tau_{N-1}+T)}(\xi_1, \ldots, \xi_N) = F_{x(t+\tau_1)\ldots x(t+\tau_{N-1})}(\xi_1, \ldots, \xi_N) \tag{2.14}$$
$$\forall t \in \mathbb{R}, \forall (\tau_1, \ldots, \tau_N) \in \mathbb{R}^{N-1}, \forall (\xi_1, \ldots, \xi_N) \mathbb{R}^N$$

$$\mathrm{E}[x(t)] = \mathrm{E}[x(t+T)] \tag{2.15}$$
$$\mathrm{R}_x(\tau) = \mathrm{R}_x(\tau+T) \forall t, T$$

Cyclostationary detectors exploit these differences in statistical properties by developing the Spectral Correlation Function (SCF) [16]:

$$S_x^\alpha(f) = \int_{-\infty}^{\infty} R_x^\alpha(\tau) e^{-j2\pi f\tau} d\tau \tag{2.16}$$

Where $S_x^\alpha(f)$ is the SCF, α is the cyclic frequency, f is the frequency, and R_x^α is the cyclic autocorrelation defined as:

$$R_x^\alpha(\tau) = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} R_x(\tau) e^{-2j\pi\alpha t} d\tau \tag{2.17}$$

The cyclic coherence function (SOF) is given as in Equation (2.18) [49].

$$C_X^\alpha(f_0) = \frac{S_X^\alpha(f)}{[(S_X^0(f+\frac{\alpha}{2}))^*(S_X^0(f-\frac{\alpha}{2}))]^{(1/2)}} \tag{2.18}$$

Where $R_x(\tau)$ is the received signal autocorrelation and T_0 is the period.

Cyclostationary detectors detect and classify signals first by using the cyclic autocorrelation function to develop an estimate for the SCF. The SOF for multiple values of α is then computed resulting a three dimensional plot. When data is modulated with different modulation schemes the characteristics of the SOF changes as well. Each modulation scheme has a distinct SOF which is the property that cyclic detectors exploit. However, since the SOF must be computed for multiple values of α, resulting in large amount of computation. In addition, a secondary system that classifies the signal is needed as well further increasing the computational complexity.

Cyclostationary detectors can detect and classify signals, however some information regarding the signal such as modulation scheme but not symbol timing, is needed prior to classifying the signal. Cyclostationary detectors have high implementation and computational complexities but do not need to coherently receive signals.

Table 2.3: A Comparison of the USRP1 and USRP2 [3].

	USRP1	USRP2
Data Bus	USB 2.0	Gigabit Ethernet
FPGA	Altera Cyclone EP1C 12	Xilinx Spartan III 2000
Sampling Rate	64 MS/s	100 MS/s
RF Bandwidth	8 MHz	25 MHz
Daughterboard Capacity	2	1

2.6 Spectrum Sensing Hardware and Software

Just as critical to spectrum sensing as the algorithms is the hardware and software. This section will provide the background for the hardware and software that will be used for spectrum sensing in this project.

2.6.1 The USRP1 and USRP2

The USRP1 and USRP2 are both designed by Ettus Research LLC [50]. The hardware design, including schematics and Verilog code, is all open source. Both the USRP1 and USRP2 share the same basic architecture. In both architectures the USRP is the radio frontend, the only responsibility of the USRP is to capture RF data, downconvert, and package the data. Once packaged the data is transferred to a host via a USB 2.0 bus for the USRP1 or a gigabit Ethernet interface for the USRP2. Further processing is then performed on a host machine, typically a PC with x86 architecture. Both the USRP1 and USRP2 have a permanent motherboard and a removable daughterboard. The motherboard houses all the common hardware such as the FPGA, ADCs, DACs, and the Ethernet controller. The removable daughterboard (or interchangeably, the daughtercard) houses the hardware that is needed to change frequency bands, such as the RF frontends.

As far as the user is concerned the system architecture for the USRP1 and USRP2 are similar but their specifications vary as listed in Table 2.3.

2.6.2 GNU Radio

The signal processing that is performed on the host is done within the GNU Radio software framework. GNU Radio provides the reconfigurable signal processing blocks that are necessary for software defined radios. GNU Radio is an open source project allowing for SDR developers to develop unique signal processing blocks and SDR systems. The architecture of GNU Radio is that the signal processing block are written in C++ and they are then connected together in Python using a SWIG [51] interface, as shown in Figure 2.13.

This software architecture allows the speed of C++ to be exploited while keeping the

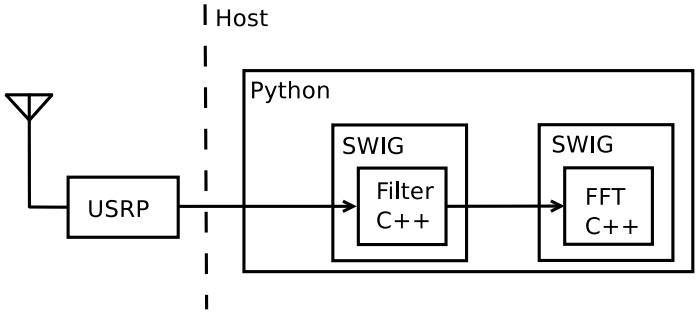


Figure 2.13: A Trivial GNU Radio Signal Processing Chain.

simplicity of implementation by using Python to tie the blocks together.

2.7 Chapter Summary

This chapter explains what spectrum is, how it is used, and what it means for spectrum to be used. Spectrum use is when a transmitter transmits a signal that can interfere with other transmissions in any one of the five domains that make up spectrum. This chapter also explains what spectrum sensing is, what characteristics a spectrum sensor needs, and how to form a spectrum sensing network. Spectrum sensing is when any one of the five domains that makes up spectrum is characterized. A spectrum sensor needs a combination of hardware and software to allow for that data to be collected. Finally, a spectrum sensing network, composed of multiple spectrum sensors, needs some type of communication channel

and technique for fusing data together.

Chapter 3

Proposed Design of Spectrum Sensors and the Spectrum Sensor Network

In this chapter, the design of a low cost distributed spectrum sensing network is proposed. The chapter begins with an overview of current spectrum sensing technology. Discussion then moves on to the requirements for a spectrum sensor, a distributed spectrum sensing network and then how this work meets those requirements in a novel way. This chapter then proceeds with a spectrum sensor design proposal that includes the design of the hardware and software systems. The hardware design focuses on the software radio, the daughtercard and the antenna. The software section shows the design of the software radio system, the data storage format, the command and control system and the data fusion system.

3.1 Spectrum Sensor Overview

This section outlines a list of requirement for a spectrum sensor and spectrum sensing network. Following this is a description of the objectives of this work and then compares those objectives with the current state of the art. The system described in this work pos-

sesses the following abilities. The system should be able to make measurements that are wideband, spatially varying, and time varying. The system should also have the following attributes practicality and ease of use. The spectrum sensing network proposed as a part of this system possesses the following abilities data sharing and data fusion.

Various current works possess a subset of these abilities or attributes. For example, some studies conducted perform wideband and time varying measurements [8, 9, 10]. Another study outlines the design of a wideband distributed spectrum sensing network, but the design may be difficult to realize in practice [12]. In this design, the spectrum sensing system is dependent on a network on spectrum analyzers. This system can easily exceed a cost of hundreds of thousands of dollars, depending on the number of sensors. The authors of this spectrum sensing network architecture even admit that any practical implementation of this network would depend on a collection of volunteers. Consequently, this system can be prohibitively expensive for a single research group to realize. Additionally, there are obvious problems with depending on volunteers. One problem may be that the volunteer spectrum analyzer may not be available for a specific duration. Other problems with volunteer spectrum analyzers is data verification, where a volunteer spectrum analyzer could be calibrated incorrectly. Another problem is that the spectrum analyzer configuration may not be able to be employed uniformly across all sensors.

This project puts an emphasis on practicality by assigning an estimated node cost of $ 2000 USD. With a lower cost, a single research group may afford to purchase multiple sensors and configure them to a specific scenario. This spectrum sensing design allows one to design their own spectrum sensing experiments without depending on other research groups.

3.2 Spectrum Sensor Design

This section details the design of each individual sensor, from the sensor hardware to the sensor software. The selection, configuration and design of the hardware and software components is determined by the following feature set:

- Each individual sensor must have a low cost. The target price is $ 2000 USD.

- Each sensor must have wideband sensing capabilities, from 50 MHz to 2.2 GHz.
- Adding sensors to the spectrum sensing network must be simple.
- The sensor should be mostly independent, that is the sensor should not need outside assistance after the initial setup or before the end of the scan. Each sensor should be able to function for an extended period of time on its own.
- The system should provide a mechanism for the sensors to share data, whether via a computer network or off-line.
- Tools for collecting and analyzing the data should be provided.

All of these features are flexible. For example, the cost feature may be over $2000 USD as long as the price remains low enough for a research group to afford multiple sensors.

3.2.1 Sensor Hardware

This subsection discusses the selected sensor hardware. Topics include motivation for selecting each piece of hardware, the function that hardware performs and a brief discussion on how the hardware component works.

The USRP2

The USRP2 is the main hardware component for this system, which is shown in Figure 3.1. The basic function of the USRP2 is to capture samples at high (passband) frequencies and convert them into a form (baseband) where they can be processed on the host Linux PC. The USRP2 does this by interfacing with an RF daughtercard, then preparing the samples to be transferred from the USRP2 to the host Linux PC via a gigabit Ethernet interface. The USRP2 to Linux PC interface is shown in Figure 3.2.

The USRP2 was selected as a hardware platform for several reasons. The first reason is the known low cost of $ 1400 USD [52]. The second is that the USRP2 can meet performance requirements in that it is capable of performing wide-band scans. Third, since the USRP2 requires a host Linux PC, the system as a whole is capable of networking with other sensors using already existing network infrastructure to form a spectrum sensing network.

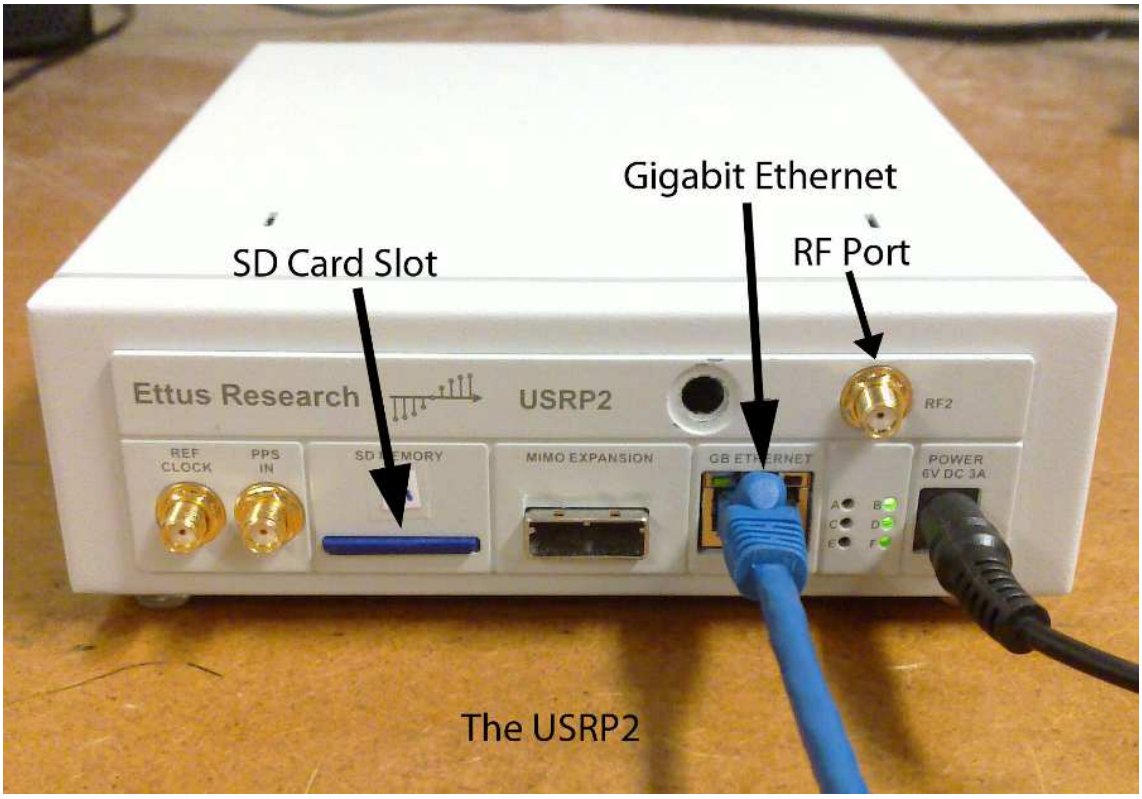


Figure 3.1: The USRP2.

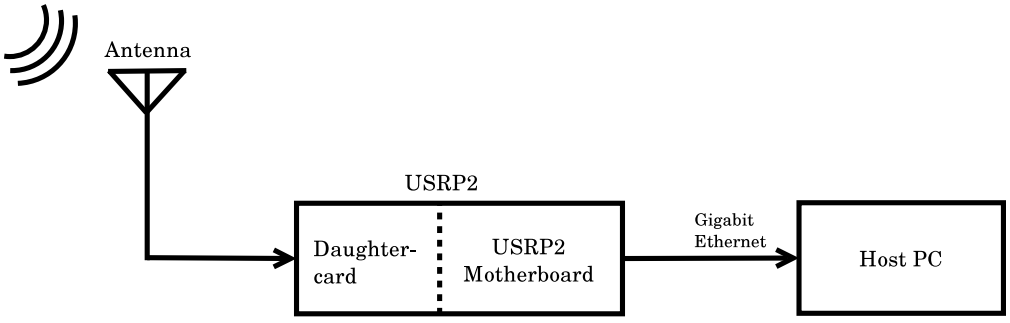


Figure 3.2: A Block Diagram of the Hardware System.

The USRP2 is part of a family of software defined radio products that Ettus Research sells. Of the other products the USRP E100 is particularly interesting to this project. The E100, the embedded software defined radio platform offered by Ettus Research, sells for $1300 USD [52]. The USRP E100 is also a stand-alone system, that is no host Linux PC is needed. This lowers the cost further. A host Linux PC is not needed because the USRP E100 comes with a 720 MHz OMAP3 (ARM Cortex A8 processor & TI C64x+ DSP) and a Xilinx Spartan 3A-DSP1800 FPGA [50]. The USRP E100 is capable of running a full Linux distribution, so any development that is done for the USRP2 can be ported to the USRP E100 easily provided that the deployed system is not too computationally complex.

The USRP2 houses the daughtercard, which is responsible for receiving signals and mixing them from passband directly to baseband. The daughtercard then passes the data to the USRP2 motherboard. Upon receiving the baseband signal from the daughtercard the USRP2 first samples the signal with two analog to digital converters that sample at a rate of 100 Msps at 14 bits per sample. The two ADCs perform quadrature sampling. One ADC samples the inphase component of the signal while the other samples the quadrature component of the signal. Next, the samples go to the FPGA, a Xilinx Spartan III. The FPGA is responsible for filtering and decimating the sampled baseband signal. This is done on the FPGA with a digital down converter (DDC). The DDC is made up of three filters: a cascaded integrated comb (CIC) filter, a high rate half band filter, and a low rate half band filter. All three filters are used for decimation. The CIC can decimate at a rate from 1 to 128 and each of the half band filters can decimate with a rate of 2 [53]. Using a combination of the CIC and half band filters the USRP2 can decimate the signal with a rate from 4 to 512.

The USRP2 requires a minimum decimation rate of 4. This is due to the maximum throughput of the gigabit Ethernet interface, which is the bottleneck in this system. The USRP2 can provide data at a higher throughput than gigabit Ethernet can provide. A gigabit Ethernet interface can handle a sustained throughput of 1 gigabit per second. The USRP2 however can produce a sustained throughput nearly three times greater than this.

The USRP2 performs quadrature sampling, meaning that the USRP2 has two ADCs that sample 90° out of phase, both at 100 Msps. With 14 bit samples the total bit rate

needed 2.8 gigabits per second, as shown in Equation (3.1). Equation (3.2) shows that with a decimation rate of 4, only 700 megabits per second is needed throughput.

$$(2) \cdot \left(100 \ \frac{\text{megasamples}}{\text{second}}\right) \cdot \left(14 \ \frac{\text{bits}}{\text{sample}}\right) \ = \ 2.8 \ \frac{\text{gigabits}}{\text{second}} \tag{3.1}$$

$$(2) \cdot \left(\frac{1}{4}\right) \cdot \left(100 \ \frac{\text{megasamples}}{\text{second}}\right) \cdot \left(14 \ \frac{\text{bits}}{\text{sample}}\right) \ = \ 700 \ \frac{\text{megabits}}{\text{second}} \tag{3.2}$$

When the decimation rate is set to 4, there is throughput left for network overhead, which is also required. In Equations (3.1) and (3.2), the required throughput is the product of the number of ADCs, the decimation rate, the sampling rate, and number of bits per sample. Equation (3.1) has a decimation rate set to 1 and Equation (3.2) has a decimation rate set to $\frac{1}{4}$.

After decimating the signal the FPGA in the USRP2 then encapsulates the data such that it can be transmitted to the host Linux PC. The USRP2 uses the UDP over IP so the IP address and port must be assigned. There are also headers for network and data link layers of the network stack. The data encapsulation and protocol implementation for the data link, network (IP), transport (UDP), and application layers, all performed on the FPGA, while the physical layer (Ethernet) is performed on a National Semiconductor DP83865 [54].

The WBX Daughtercard

The daughtercard is responsible for all the signal processing before the signal is sampled. That is, the daughterboard must be able to tune to a specific frequency, amplify, separate the in-phase and quadrature components and mix the signal down to baseband. The first item in the receiver on the WBX is a digitally controlled attenuator, which is used to properly set the gain for the system. Next, is a low noise amplifier. Then, there is the quadrature demodulator, which mixes the signals down to baseband and separates the in-phase (real valued) and quadrature (complex valued) signal components. Following the quadrature demodulator is a low pass Chebyshev filter that is used for anti-aliasing. After the filter is the ADC driver, the purpose of the ADC driver is to amplify the signal so the signal voltage matches that of the ADC. When the voltage levels match, the full dynamic range of the ADC can be utilized [55]. The last component in the WBX before handing the signal off to the USRP2 motherboard is a low pass Butterworth filter. This signal processing chain

is shown in Figure 3.3.

The WBX board has one more feature that makes it particularly flexible. The WBX board has a granddaughterboard that rests on top of it. The WBX is the only daughtercard that currently has this feature. The granddaughterboard is there so users can easily add other components to their systems, such as power amplifiers for transmitters. The WBX transceiver is the daughtercard selected for the spectrum sensor. The selection of the WBX is due to it being a wideband transceiver. The WBX covers from 50 MHz to 2.2 GHz, most of the desired range. The noise figure for the WBX ranges from 5 to 7 dB, depending on the operating frequency. In keeping with the low cost feature, the cost of the WBX daughter card is $ 450 USD [52].

Antenna Selection

The antenna is an important part of any communications system, this system is no different. The most important feature of this system from the perspective of the antenna, is the wideband feature. Since the WBX daughtercard can tune from 50 MHz to 2.2 GHz, an antenna that covers as much as the corresponding frequency response is necessary. A class of antennas with this wideband characteristic is the discone antenna. The wire discone antennas are ideal for this application since they are wideband, low cost and omnidirectional. A suitable antenna for this spectrum sensing system is the Diamond D220 wire discone antenna, which has frequency coverage from 100 MHz to 1.6 GHz [56]. The antenna use in this project is located on the roof of the Atwater Kent Electrical and Computer Engineering building, 42° 16.511 N, 71° 48.427 W with an elevation of 164 meters above sea level.

Another antenna is the VERT 900 sold by Ettus Research LLC. The VERT 900 is inexpensive at $ 35 USD [57] and is a wideband antenna as shown by the antenna characterization in Figure 3.4. Figure 3.4 shows the S Parameters as measured by the author. The measurement was conducted by attaching the VERT 900 antenna to a network analyzer and then sweeping a signal from 50 MHz to 3 GHz. The S Parameters of an antenna is the fraction of power lost for each frequency in dB. The horizontal red lines in Figure 3.4 shows the -3 dB of loss and the vertical lines shows where the S Parameter and -3 dB of loss lines intersect. The frequency in between the vertical red lines is the operating frequency of the antenna.

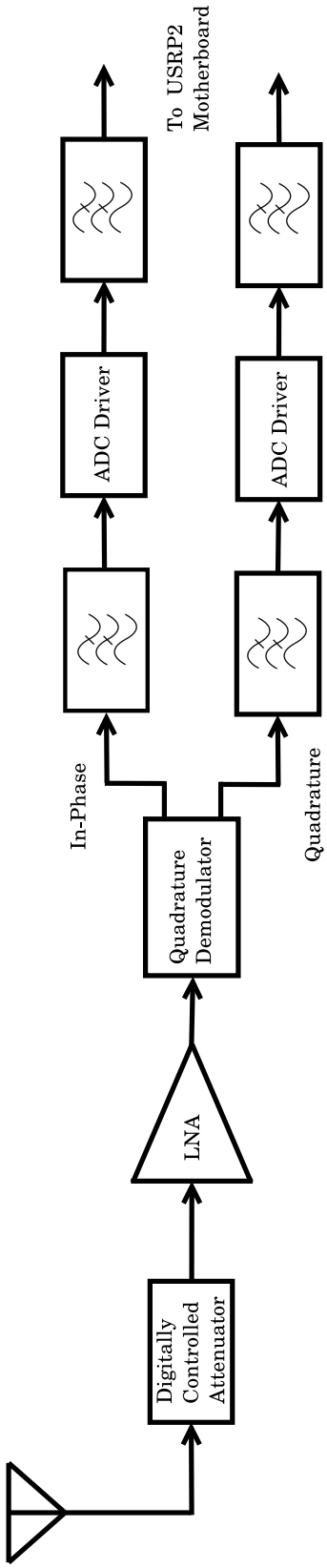


Figure 3.3: A Block Diagram of the WBX Signal Processing Chain.

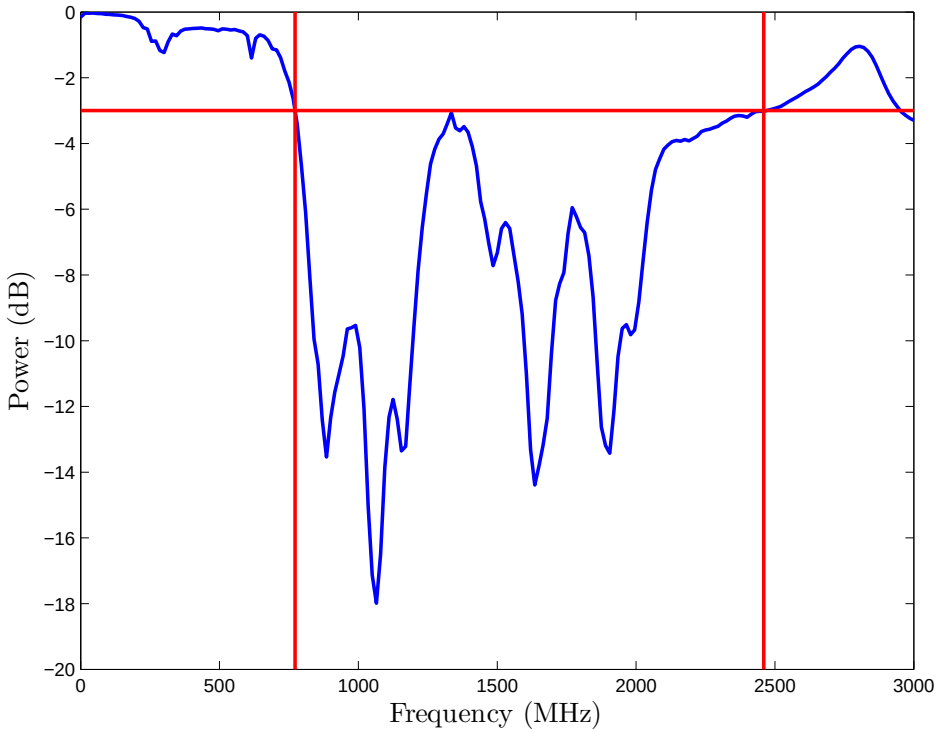


Figure 3.4: The S Parameters of the VERT 900 Antenna.

3.2.2 Sensor Software

This section describes the design of the various software modules that make up the distributed spectrum sensing system. Since this is a distributed system, the software operates on various machines. Throughout this section, the software described operates on either a spectrum sensor on the master node, which is the system where data is collected and processed such that a decision is made regarding the state of the spectrum.

The Universal Hardware Driver

The first software module that is discussed is the universal hardware driver (UHD), the software that resides on the host PC and is responsible for the communications with the USRP2. The UHD is written in C++ and is distributed by Ettus Research, LLC. It is

possible to use the UHD independently (that is, without GNU Radio) if one wishes to use another signal processing software framework.

The UHD was created by Ettus Research for several reasons:

1. To simplify the process of porting GNU Radio applications for different radio hardware.

2. To make a software driver that can communicate with USRP hardware family that is not licensed under the GPL. This makes the creation of propriety software radio applications more simple from a licensing point of view. GNU Radio provides a GPL licensed driver.

3. To make a driver that supports a full network stack. Previously, the GNU Radio driver used only the physical and data link layers of the network stack. With the previous configuration, only root users could communicate with the USRP2 on Unix systems and using Microsoft Windows systems there are no methods to communicate with the USRP2.

The software modules in this project exploit these benefits, especially the simplification of porting GNU Radio applications. Before the UHD switching from the USRP1 to the USRP2 was tedious. With the UHD, the application programming interface (API) to communicate with the USRP device became simple. To switch from one USRP product to another, all that is needed is a change in the initial call to the UHD software.

GNU Radio Signal Processing Chain

The UHD fetches the data from the USRP2 and then provides the data to GNU Radio. Once the data is provided by the UHD, it is passed to the stream to vector signal processing block. The UHD produces samples serially, where they are produced one at a time. Due to the use of the FFT in this system, vectors of samples must be formed. The length of the vector is the FFT size. For example, if the desired FFT size is 512, then the stream to vector converter will accumulate 512 samples, then package them into a single vector and pass that vector to the next signal processing block.

The next signal processing block is the FFT block. The FFT block is responsible for taking a vector of time domain samples and finding the frequency domain information of those samples. The FFT block uses a fast Fourier transform (FFT) technique implemented in the Fastest Fourier Transform in the West (FFTW) [1] C library. The GNU Radio FFT block also uses windowing to reduce the affects of spectral leakage. Spectral leakage occurs when power from a signal "leaks" into neighboring frequency bins in an FFT. Spectral leakage occurs because every signal is multiplied by a window in the time domain. The window is always equal to the duration of the signal, making it finite length. All windows are finite length in the time domain, the window is wide in the frequency domain. Since the window and signal are multiplied in the time domain, they are convolved in the frequency domain. This convolution produces spectral leakage. The window used in this project is a Blackman-Harris window.

After the FFT block there is a block for finding the magnitude of the FFT output. The output of the FFT is complex valued, and the magnitude response is what is desired. So what is needed is a block to convert from complex to magnitude, which is next block in the signal processing chain. The complex to magnitude converter takes the square root of the sum of the squares of the real and imaginary parts of each element in the vector passed from the FFT block. The next step in this signal processing chain is to record each one of these magnitude vectors.

The block that performs this recording is the bin statistics block. This block performs several important functions. The first and most obvious of these function is to take the output of the complex to magnitude block and direct that data to the recording software. The bin statistics block does this by considering two delays, a tune delay and a dwell delay. The tune delay, in quantity of FFT vectors, is the amount of vectors that the block will discard before trusting the accuracy of the vectors. This system is constantly re-tuning itself because it is a wideband system that observes many narrow bands at a time. Each time the system re-tunes various sub-systems, such as phase locked loops, time is needed to settle. If the system is not given enough time to settle, the data measured will not be accurate. The second delay is the dwell delay, which is also a quantity of FFT vectors. The

[1] `http://www.fftw.org/`

dwell delay directs the bin statistics block to keep that number of FFT vectors before the system re-tunes. When it is time for the system to re-tune, it is the bin statistics block that issues the callback for the re-tune. This callback will check the current frequency and then call for the system to re-tune to next frequency. If the current frequency is not the maximum frequency, then the next frequency is simply the sum of the current frequency plus a step frequency. If the current frequency is maximum frequency, then the system will re-tune to the minimum frequency. This process repeats until the number of sweeps is met.

The entire GNU Radio signal processing chain is shown in Figure 3.5. The flow chart for re-tuning are shown in Figure 3.6.

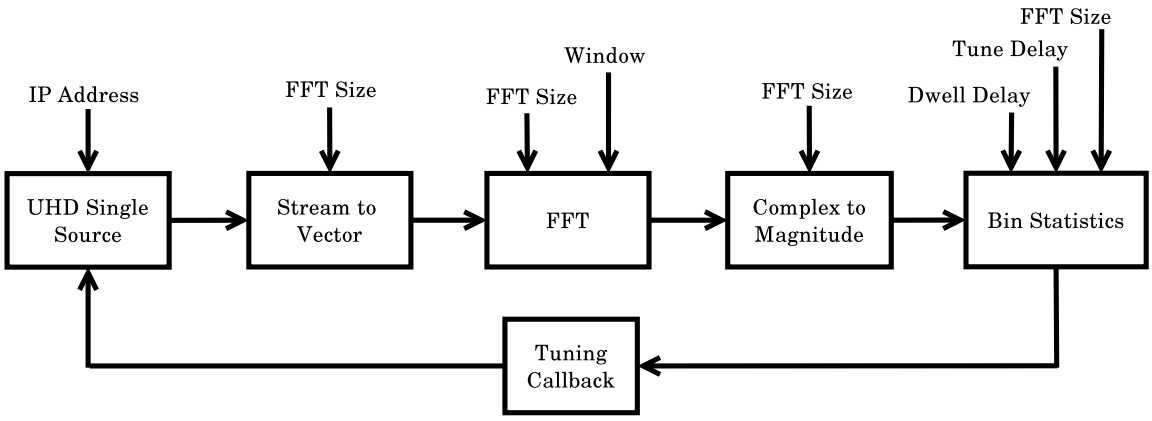


Figure 3.5: The GNU Radio Signal Processing Chain.

The HDF5 File Format

The hierarchical data format version 5 (HDF5) is the data storage format chosen for this project. HDF5 is a binary file format designed for storing scientific data. The HDF5 file format was chosen for a variety of reasons:

- HDF5 is a binary file format meaning that it is efficient in terms of storage space than non-binary file formats such as comma separated values (CSV).
- There is an official HDF5 library in C/C++ [58] and an unofficial Python wrapper for the C/C++ library [59]. MATLAB also supports reading and writing to HDF5

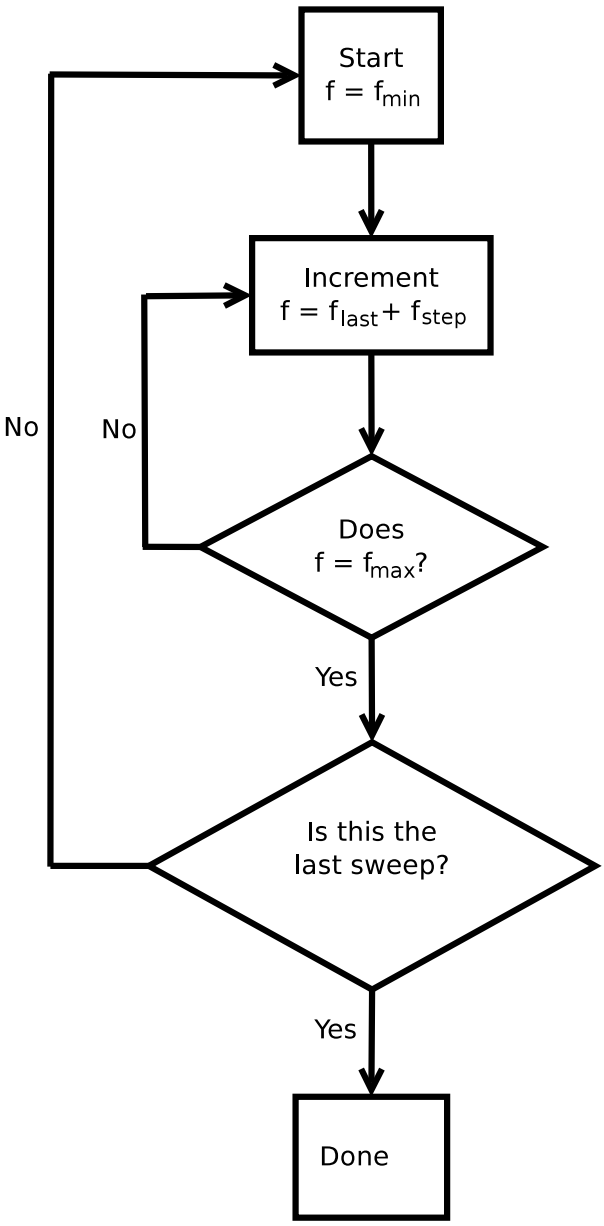


Figure 3.6: The Re-Tuning Flow Chart.

files.

- HDF5 is capable of storing all the datatypes needed for this project.
- HDF5 is free to use [60].

In HDF5 sets of data are grouped into datasets, which are organized in groups. Datasets may be any grouping of data such as strings, scalar number, vectors or matrices. The model in which HDF5 organizes datasets and groups is analogous to the model that Unix-style file systems organizes files and directories. As in Unix file systems there is a root directory, "/," in which every directory or file is a member. In HDF5 there is a root group in which every group or dataset is a member. The root group contains all datasets or groups below.

The data generated using this project takes advantage of the hierarchical features of HDF5. In the root directory there are two datasets that make up the meta-data from the sweeps and a group of datasets where each dataset is a sweep. The first dataset of meta-data is "Time," which is a 7 by 2 matrix of strings that describe the start of the scan. The second dataset, which is called "Scan Info," is a 13 by 2 matrix of strings that describe the items such sampling frequency, FFT size, and other scan characteristics. The group in the root directory contains the datasets for each sweep. The group is called "scans" and contains a dataset for each sweep named "sweep-000001". Each sweep dataset is an N by 2 matrix, where N is the number of scans per sweep. The first column is frequency and the second column is power. This format is visualized in Figure 3.7 and an example data set is shown in Figure 3.8.

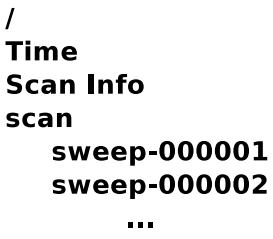


Figure 3.7: The HDF5 Format Used in This Project.

/

Scan Info

FFT Size	8192
Decimation Rate	N/A
ADC Rate	500000
Min Frequency	50187500
Max Frequency	2200440000
Sample Rate	500000
Number of Sweeps	N/A
IP Address	192.168.10.2
Receive Buffer Size	60000000
Initially Set Gain	5
Minimum Frequency	18750000
Maximum Frequency	2250000000
Antenna	TX/RX
Name	uhd single_usrp source

Time

Day	Monday
Month	01
Day	03
Year	2011
Hour	13
Minute	31
Second	02

scans

sweep-000001

5.0075E7	3.818649
5.0225E7	7.219326
...	...

sweep-000002

5.0075E7	3.818649
5.0225E7	7.219326
...	...

Figure 3.8: An Example of HDF5 File Used This Project.

System Usage

This spectrum sensing system is intended to be used as a single or multisensor system. However, when one chooses to use a single sensor or a set of distributed sensors, the process varies only slightly. For each sensor there is a text file, called `sensor_info.txt`, that is easily modified to change the characteristics about that particular sensor. Before a scan occurs, the file must be modified accordingly. In this file the following fields are kept:

- Sensor Number.
- Total Number of Sensors.
- Sensor Longitude.
- Sensor Latitude.
- Sensor Name.

The purpose of this file is to keep track of where the sensor is physically as well as establish a unique name or number for each sensor. The fields are copied into the HDF5 file for storage and are then used when analyzing the data.

Once the `sensor_info.txt` file is set up properly the scan parameters need to be determined. Examples of these parameters are the FFT size, sampling rate, or gain. The parameters take the form of command line arguments and are given by the `help` command as shown in Appendix A.

3.2.3 Sensor Control System

The control system is the software package or collection of packages that is used to control all the nodes. The two responsibilities for the control system is to start the all the scans simultaneously and then to manage all the data that the sensors collect. A number of control systems were considered:

- CORBA[2] - A remote procedure call (RPC) software framework.

[2]Common Object Request Broker Architecture

- XMPP[3] - A protocol for sharing data, mainly used for Internet chat clients.
- NTP[4], SSH[5], RSync, Cron and standard Unix tools - A collection of small tools that specialize in establishing secure connections and efficient file transfer respectively.

CORBA and XMPP are flexible, full featured software packages that go beyond the need for the control system for this project. The control system for this project needs only to be reliable, secure, efficient and simple to work with. The standard Unix tools provide all of this functionality without being overbearing. The tools listed, NTP, SSH and RSync provide that basic means of synchronizing, controlling and transferring files the sensors respectively.

The sensor network architecture using these tools is shown in Figure 3.9. The method used to control all the sensors has multiple steps:

1. All sensors synchronize to a common time server using NTP.
2. The master sensor establishes a secure connection using SSH and delivers scan instructions, such as start time and other scan parameters.
3. The sensors now wait for the correct time to begin their scans.
4. Once the scans have completed the master sensor retrieves the scans from each sensor.

3.3 Analysis Software

This section describes the software developed for analyzing the collected spectrum data. The tools developed for analyzing the data fall into three categories: utilities, the MATLAB GUI, and Google Earth visualizations. The utilities are for preparing the data for use with either the MATLAB GUI or Google Earth. The MATLAB GUI is used to visualize the data directly. Google Earth is used for visualizing the spatial elements to the data collected.

[3]Extensible Messaging and Presence Protocol
[4]Network Time Protocol
[5]Secure Shell

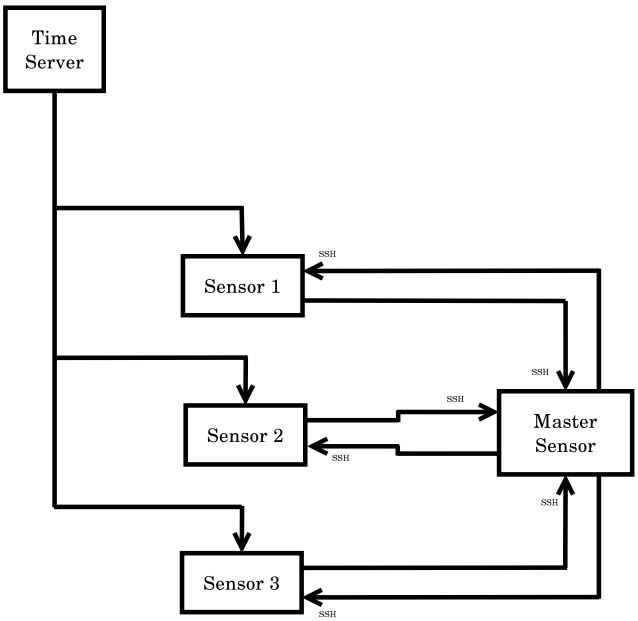


Figure 3.9: The Sensor Network Architecture.

3.3.1 Analysis Utilities

The data analysis utilities developed are for preparing the data for use with either MATLAB or Google Earth. The first utility `combineSweeps.py` is for combining each HDF5 file collected from separate sensors into one HDF5 file. This utility first creates a combined HDF5 file. All the datasets in the HDF5 files that the individual sensors produce have two columns; for example each sweep dataset has a frequency and power column. In the combined HDF5 file, columns are appended for each sensor. For example, a system with four sensors will have 5 columns for each sweep, one column for frequency and four columns for each sensors measurement. This utility is show in Appendix C.

The second utility, `makeKML.py`, is for generating a Keyhole Markup Language (KML) file. This utility takes the combined HDF5 file generated by the utility described in the preceeding paragraph and produces a KML file that is used by Google Earth to produce show the sensor positions on a map. A sample spectrum sensing system is shown in Figure 3.11(a). This utility is shown in Appendix B.

3.3.2 MATLAB Graphical User Interface

A wideband spectrum sensing network gathers a very large quantity of data. The datasets that are gathered are all multi-dimensional, as seen by the following variation:

- Spatial.
- Frequency.
- Time.

Data that is multi-dimensional must be analyzed in every dimension. That is, the data gathered should be analyzed by the sensor, by the frequency band, and by the time in which the sensing took place.

The need for multi-dimensional analysis leads to a need for several different analysis tools. The two most frequently used tools are the periodogram and the spectrogram. The periodogram is an estimate of the power spectral density (PSD) of a signal using a finite collection of samples [15]. The spectrogram is a set of periodograms taken one after the other in time that show how the estimate of the PSD changes with time [61]. The periodogram and spectrogram allow for the data collected to be analyzed by frequency and by time. This leaves the spatial dimension, which is not considered by either the periodogram or spectrogram.

In Figure 3.10 the MATLAB GUI with a spectrogram of an ATSC television signal loaded is shown. Starting in the upper left hand corner is the drop down menu, `File Name`, that selects the file to view, the list of files is generated automatically. Proceeding downwards is the `Sweep Number` and `Sensor Number` which are used to select the sweep and sensor to analyze respectively. Underneath that is the `Turn On Averaging` radio button, for multi-sweep scans this will shown an average of the periodogram for all the sweeps. After that is the `Turn On Spectrogram` radio button, when selected this radio button shows the spectrogram. This is followed by the `Show Distribution` radio button that will show the user the power distribution histogram for the frequency bands that the user selects with the two text boxes below. Then there is the `Update` button which, when pressed, updates the graph on the right. Underneath the update button is the `Scan Information` text box, this

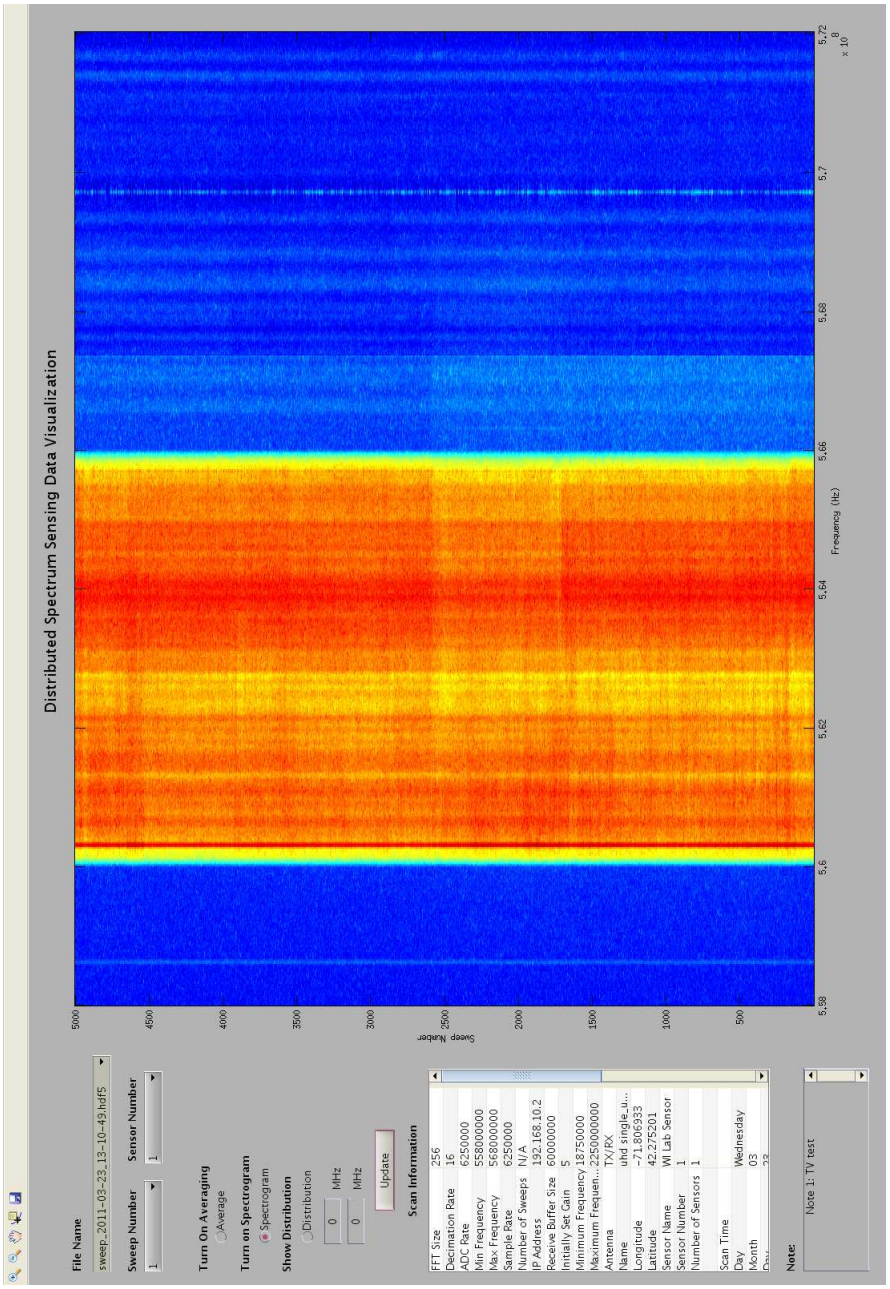


Figure 3.10: The MATLAB GUI Displaying Television Spectrogram.

box displays all the information relevant to this scan. Items such as sampling frequency, scan start time, scan end time, and estimated scan end time are shown. In the bottom left corner is the box labeled `Note:`, in this box any notes that the user leaves when starting the scan are shown.

3.3.3 Google Earth and Spatial Analysis

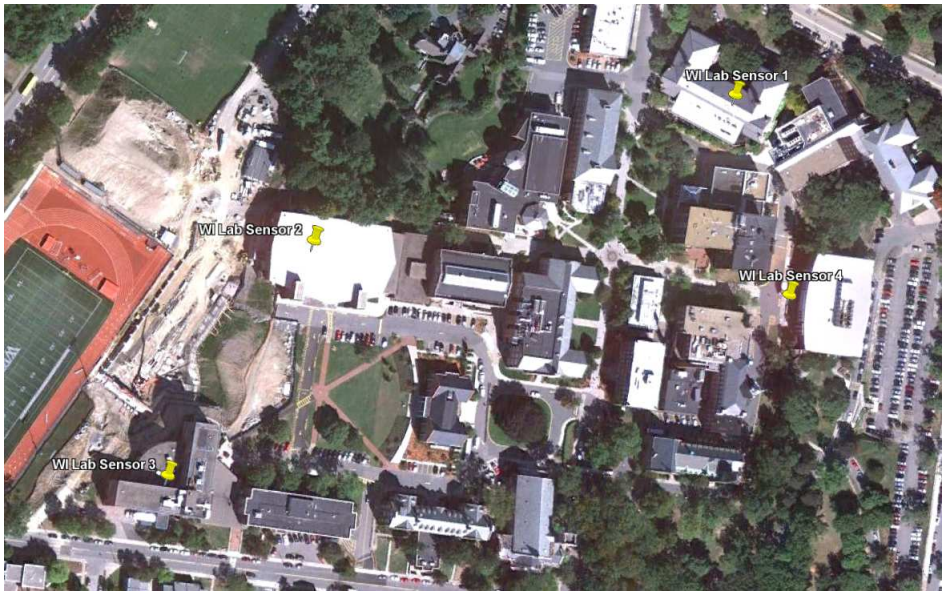
Analyzing spectrum data in the spatial dimension is new, with techniques beginning to be developed in only the past several years. Since this analysis is new there are a variety of techniques proposed for analyzing spatially distributed spectrum data. The technique most applicable to this system is the PSD map [62]. The PSD map is a visualization that uses the measurements from different spatially varying sensors to make a map of the distribution of power in space when a particular frequency is held constant.

The PSD map can only be created using the measurements from the several sensors. What is desired is that the map shows how power varies over continuous space. However, it is assumed that spectrum sensors cannot be placed with the resolution required to make useful PSD maps. The density of spectrum sensors (e.g. sensors per square meter) cannot be high enough to create useful maps with the data alone. This is especially true when the number of sensors in low. To fill the in the unmeasured areas, the measurements that the sensors make is interpolated so spectrum use as it varies in space can be better analyzed.

The need for a high sensor density is shown in [63], where in simulations 2400 spectrum sensors were used to detect three transmitters. For the system proposed in this work, 2400 spectrum sensors is unrealistic from two perspectives: implementation complexity and cost. To deploy a 2400 node spectrum sensing network would mean that all 2400 spectrum sensors would need to be separately powered and connected to a network. Additionally, this spectrum sensing system is intended for low cost applications. Using the $ 2000 USD cost per node estimate, a 2400 node spectrum sensing network would cost $ 4.8 million USD.

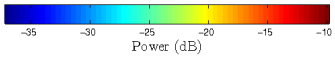
The PSD map is useful for analyzing how spectrum use varies in space. The natural extension of this is to compare the PSD map that this spectrum sensing network generates to the environment around the spectrum sensing network. Overlaying the PSD map onto a geographic map allows for geographic features to taken into the analysis. An ideal software

platform to do this is Google Earth. Google Earth accepts the KML files that the utilities generate and can display sensor locations along with the PSD maps overlaid on top of the surrounding environments. A sample of this is shown in Figure 3.10(b), where one can see four spectrum sensors distributed over the WPI campus as well as a PSD map that was generated using hypothetical data.



(a) A Map on its Own

(b) A Map with a PSD Map Overlaid on Top



(c) The Key for the PSD Map

Figure 3.10: Two Maps of the WPI Campus.

3.4 Chapter Summary

This chapter discusses the proposed design of this system. The chapter starts with an overview of some of the features a spectrum sensor should provide and then addresses how this project addresses those features. Then, there is a discussion on the proposed hardware and software packages that are used. Following this, there is a discussion on the utilities that were created to help with the analysis of the measurements.

Chapter 4

Experimental Results

This chapter discusses several experiments performed using the distributed spectrum sensing system described in Chapter 3. Two major experiments were conducted: the first tests the performance of a single sensor compared to an instrument and the second tests the performance of a distributed spectrum sensing system with multiple sensors.

4.1 Hardware and Software Testing Configuration

This section describes the software and hardware configuration for the sensors that were tested. The tools used as part of the testing are also described in this section.

The hardware testing configuration is homogeneous for the software and software radios. The configuration for host PCs are all heterogeneous. The configuration for the software radios, the USRP2s, is homogeneous for all the radios used in all the tests as part of this project. All the software radios that were used were USRP2s equipped with the WBX daughtercards. All the USRP2s used the same firmware and FPGA image that was burned onto the SD card. The firmware and FPGA images were built on March 17th, 2011 by Ettus Research, LLC [64]. The host PC testing configuration was heterogeneous, that is several different PCs were used to conduct the testing, brief descriptions of these PCs are listed in Table 4.1.

The software configuration is completely homogeneous for all sensors. This homogeneity was accomplished by using the same software versions for all the sensors. Using the same

Table 4.1: A Listing of the Different PCs Used for Testing.

Hostname	Host 1 & 2	Host 3	Host 4
Make	Dell	Dell	Lenovo
Model	XPS	XPS 420	x61
Processor Speed	1.87 GHz	2.4 GHz	2 GHz
Processor Cores	4	4	2
Memory	8 GB	4 GB	2 GB
Ethernet Interface	Realtek RTL-8111	Realtek RTL-8169	Intel 82566M

software version allows for everything from the operating system to the spectrum sensing script to be configured once and then re-installed on each machine.

In this project, the software for each sensor is Ubuntu Linux version 10.10, the latest version at the time of this research project. Each system was updated with all the most recent software. The version of GNU Radio used was pulled Wednesday March 23rd, 2011 and has commit ID `34313eace681a82e230c38a8cd26c0001ee823ea`, dated Monday, March 23rd, 2011. The version of the UHD used was pulled Wednesday March 23rd, 2011, the commit ID is `95b966a599c0030921dc6b530ca8c94633d905f6`, dated March 23rd, 2011.

4.2 Mean Squared Error Calculation

As one will see in the remaining sections, there is a clear difference in the spectrum produced by the spectrum analyzer and the spectrum sensor. To measure the difference quantitatively, one should find the mean squared error. The mean squared error is found first by finding the difference between the two signals, then squaring each sample of the difference signal and finally finding the mean of the resulting vector of samples. Before finding the minimum mean squared error, the signals must be adjusted so that they can be compared properly. There are three parameters of the signals that must be adjusted before finding the mean squared error, they are:

- Sampling rate - the two signals may be sampled at slightly different rates.
- Offset - the two signals must have any constant offsets removed.

- Scaling - one signal may be scaled by a constant multiplicative factor.

The first item, the sampling rate, is simple to correct. A rational resampler is used for this. A rational resampler is a three stage device that changes the sampling rate of a signal by an arbitrary rational number [15]. The first stage of the rational resampler is an interpolator which increases the sampling rate by an integer, I. The second stage is a low pass filter, which is used for anti-aliasing. The third stage is a decimator, which decreases the sampling rate by an integer number, D. The output signal of the rational resampler is a signal that has a sampling rate changed by a factor $\frac{I}{D}$. For example, if an input signal has a sampling rate, $f_s = 400$ Hz, and the resampler has an interpolation rate, $I = 3$, and a decimation rate, $D = 4$, the sampling rate of the output signal will be 300 Hz. A block diagram of a rational resampler is shown in Figure 4.1. In this figure $x(n)$ is the input, $y(n)$ is the output, I is the interpolation rate and D is the decimation rate.

After the signal is resampled it must be scaled and offset. The scaling factor, a, and the

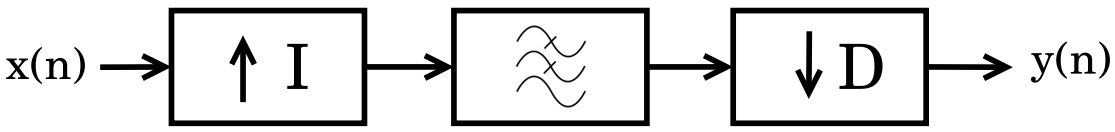


Figure 4.1: A Rational Resampler Block Diagram.

offset, b, are chosen to find the minimum mean squared error. Equation (4.1) is the equation for mean squared error and Equation (4.2) is the equation that will be used to find the minimum squared error. In this equation A_i is the i^{th} sample from the spectrum analyzer and S_i is the i^{th} sample from the spectrum sensor. The approach to finding the minimum mean squared error is to find the partial derivative of Equation (4.2) with respect to a and b separately. Then, since the a and b that minimize Equation (4.2) are needed, both equations are set to zero. The system of equations is then solved. A derivation is available in Appendix E.

$$\text{MSE} = \frac{1}{N}\sum_{i=0}^{N-1}(A_i - S_i)^2 \tag{4.1}$$

$$\frac{1}{N}\sum_{i=0}^{N-1}(A_i - (a \cdot S_i + b))^2 \tag{4.2}$$

4.3 Single Sensor Measurements

This section details the tests conducted on a signal spectrum sensor. These tests are mainly conducted using an instrument grade spectrum analyzer as a reference. The first test conducted compares the periodograms from the spectrum analyzer to those from the spectrum sensor of a known television signal. The second test compares the FM radio spectrum as measured by the spectrum analyzer and spectrum sensor. Television and FM radio signals were chosen since they are well documented technologies and are predictable. Also, they both have a duty cycle at or close to 100 percent.

The same equipment and equipment configuration was used for both experiments. The equipment used was:

- Agilent CSA N1996A Spectrum Analyzer.
  - Rated for 100 kHz - 3 GHz.
- Mini-Circuits ZFRSC-42 Two-Way-0° Power Splitter [65].
  - Rated for DC - 4.2 GHz.
  - 50 Ω impedance.
  - 0.1 dB insertion loss (typical).
- Diamond D220 Wire Discone Antenna [56].
  - Receive Frequency: 100 MHz to 1.6 GHz.
  - Impedance: 50 Ω
- USRP2 with WBX Daughterboard.
  - See Chapter 3.2.1.
- Host 4.
  - See Table 4.1.

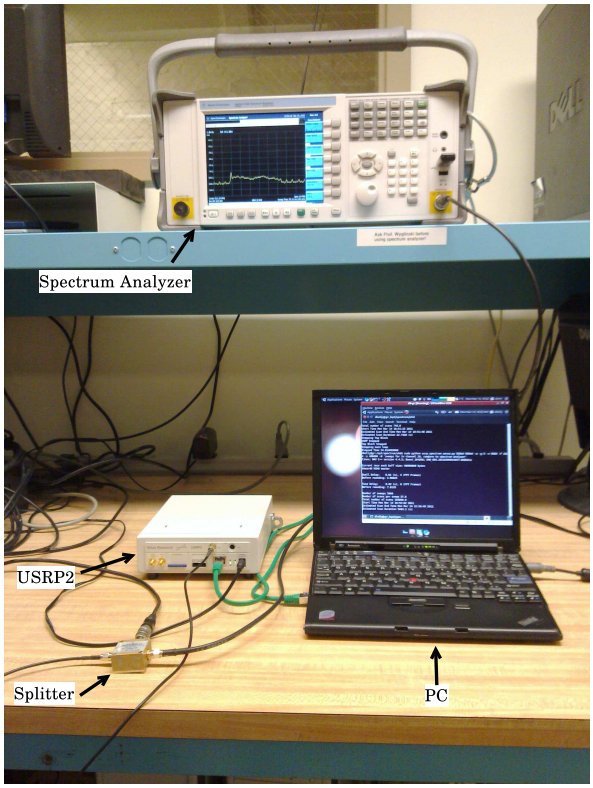
The configuration of the equipment used is as follows. The discone antenna, located on the roof of Atwater Kent Laboratories, is connected to the power splitter using an SMA cable. The first output of the power splitter is connected directly to the spectrum analyzer using an SMA cable. The second port from the splitter is connected to a USRP2 with the daughtercard WBX, this connection is also via an SMA cable. Finally, the USRP2 is connected to the laptop using an Ethernet cable. This configuration photographed in Figure 4.2(a) and diagrammed in Figure 4.2(b).

4.3.1 Television Signal Comparison

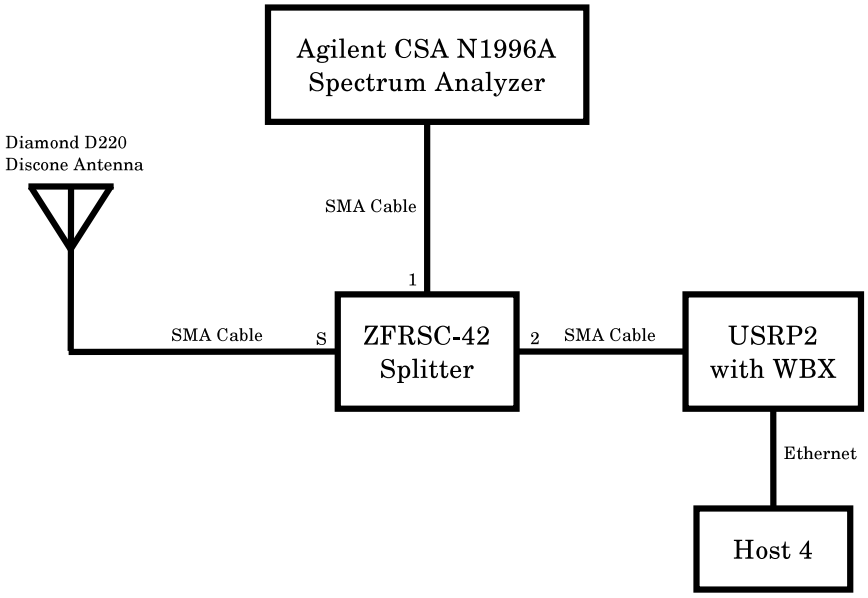
The first test conducted was a measurement of the periodogram by the spectrum analyzer and spectrum sensor from television station 29. Television station 29 has the callsign WUNI, transmits from 560 MHz - 566 MHz and is located in Boylston, Massachusetts. WUNI transmits at 270 kW and is approximately 6.4 miles away from the Atwater Kent Laboratories building [66]. The location of the WUNI transmitter is shown in Figure 4.3 and is labeled 'WUNI Transmitter,' the receive antenna is located at Atwater Kent Laboratories building and is labeled 'AK Discone.' WUNI is a digital TV station, meaning that the signal it transmits adheres to the Advanced Television System Committee (ATSC) standard. Two notable features of the ATSC signal that are discussed here are the 6 MHz bandwidth and a narrowband pilot signal that is added 310 kHz from the nominal lower edge [67].

For this test, the spectrum sensor and spectrum analyzer were configured as follows. The spectrum analyzer was tuned to a center frequency of 562 MHz with a span of 11 MHz, from 556.5 MHz to 567.5 MHz. The reference level was set to 0 dBm with the preamp set to 'On.' The video and resolution bandwidths were set to 1 kHz to provide a high level of detail. For a 1 kHz resolution bandwidth the speed for a 401 point sweep was slowed down to 628.26 ms, but a low resolution bandwidth allowed for the carrier that is part of the ATSC signal to be easily seen in the periodogram. Finally, found using 100 point exponentially weighed average [68].

The spectrum sensor was configured in a similar manner. Starting with the frequency of interest, which was set from 558 MHz to 572.5 MHz. The number of sweeps was set to



(a) A Photograph of Equipment for Testing a Single Sensor.



(b) A Block Diagram of Equipment for Testing a Single Sensor.

Figure 4.2: The Experimental Setup for the TV Spectrum Comparison.

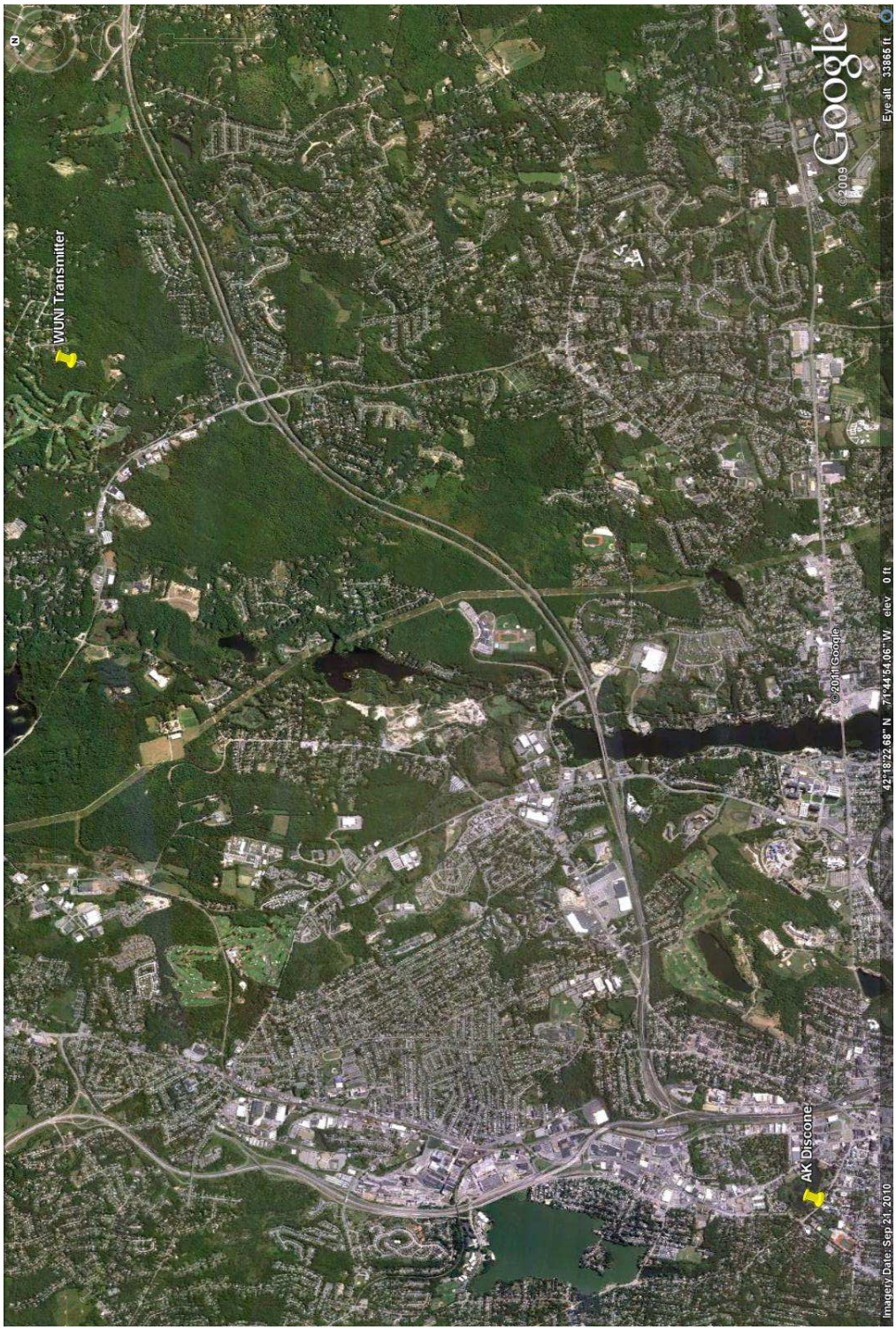
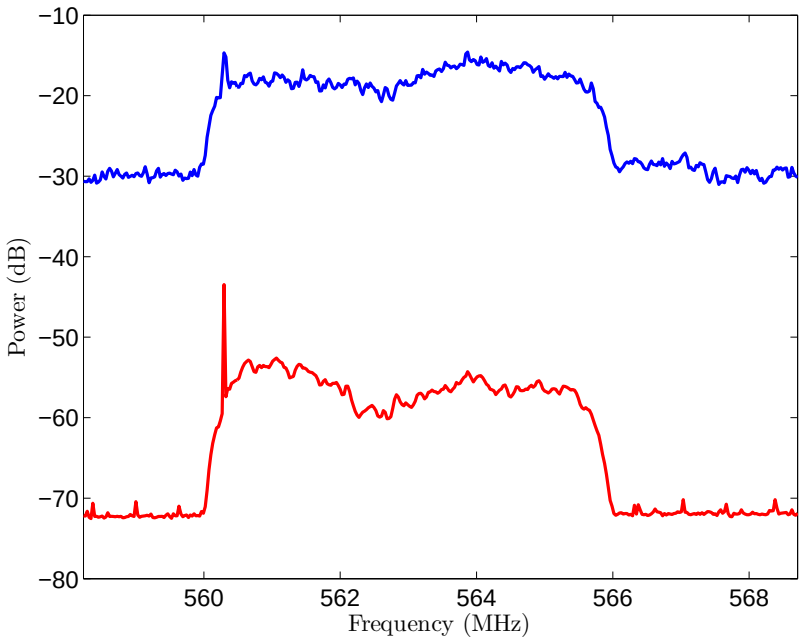
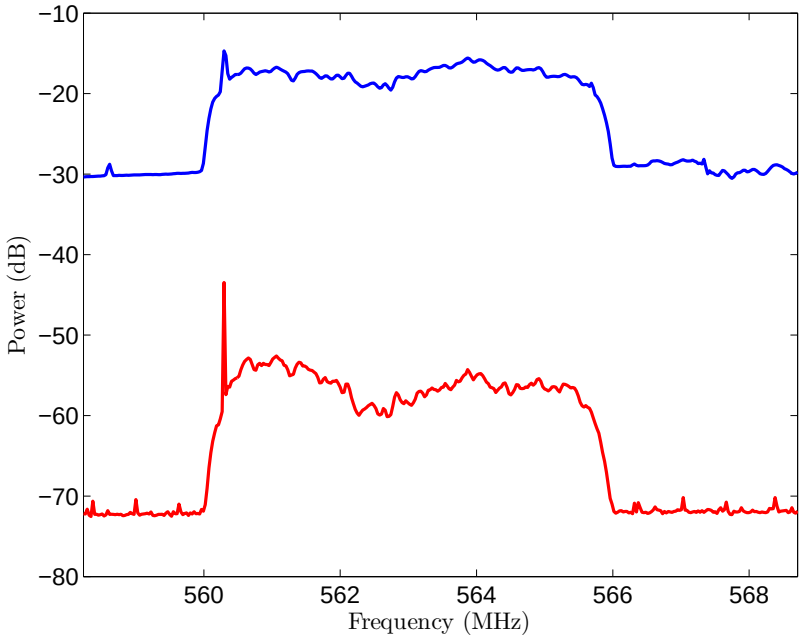


Figure 4.3: The Discone Antenna Located on top of Atwater Kent Laboratories (AK Discone) and the WUNI TV Transmitter (WUNI Transmitter).



(a) The Original Single Sweep Periodogram of TV Channel 29 WUNI from the Spectrum Sensor (Blue) along with the Spectrum Analyzer Measurement (Red).



(b) The Original 5000 Sweep Average of a Periodogram of TV Channel 29 WUNI from the Spectrum Sensor (Blue) along with the Spectrum Analyzer Measurement (Red).

Figure 4.4: The Original Sweeps from the Spectrum Sensor.

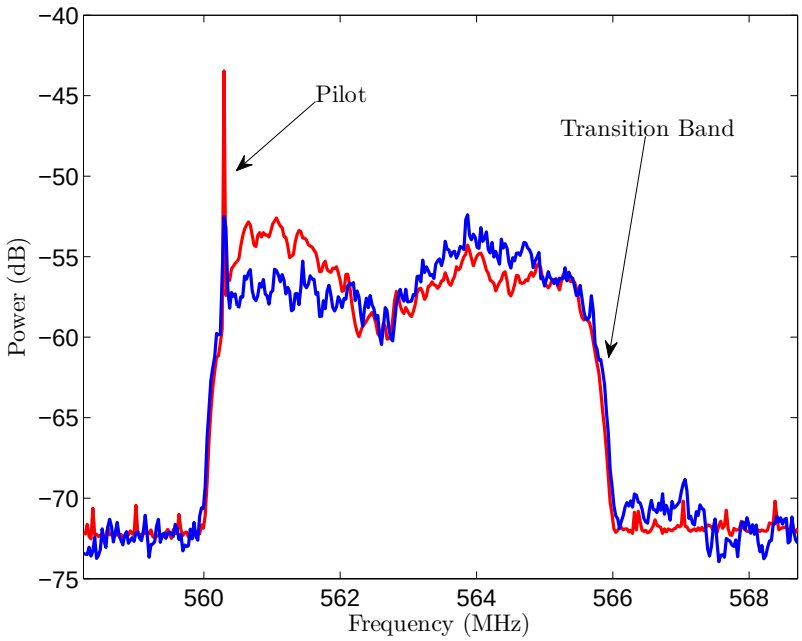
5000, which took approximately 52 minutes, 30 seconds. For a discussion on the selection of 5000 for the number of sweeps see appendix F. The decimation rate was set to 16 resulting in a sampling rate of 6.25 million samples per second. The FFT size was set to 256 points and the gain was set to 5. The receive buffer was set to 60 MB. The tune delay was set to approximately 200 ms and the dwell delay was set to approximately 10 ms.

The original sweeps from the spectrum sensor and spectrum analyzer are shown in Figure 4.4. Figure 4.4(a) shows a single sweep from the spectrum sensor (in blue) and an averaged sweep from the spectrum analyzer (in red). Figure 4.4(b) shows an average of 5000 sweeps from the spectrum sensor (in blue) and an averaged sweep from the spectrum analyzer (in red). In both figures the spectrum sensor periodogram is offset and scaled from the spectrum analyzer. Since the spectrum sensor only measures relative power, it will be readjusted to fit the spectrum analyzes measurements. It is also important to note that since one signal is being readjusted to fit another that this comparison can only measurement the relative difference in each signal and does not represent the absolute error. The adjustments for these measurements are all made and then applied to the spectra on a logarithmic (dB)scale.

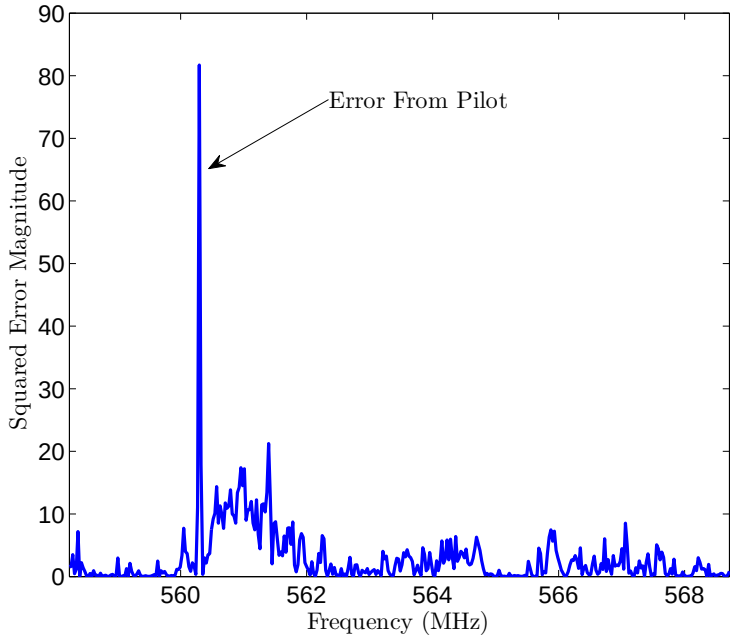
The adjusted signals are shown in Figures 4.5 and 4.6. A single sweep comparison is shown in Figure 4.5 and a comparison of a 5000 sweep average is shown in Figure 4.6.

The two plots in Figure 4.5 are the sweeps from the spectrum sensor and spectrum analyzer overlaid on top of one another and the squared error between the two. Figure 4.5(a) shows the spectrum sensor measurement in blue and the spectrum analyzer measurement in red. The spectrum sensor measurement in this figure has been adjusted to find the minimum mean squared error magnitude which is 2.864. The region with the most error is the location of the carrier at 560.31 MHz. This region of large error is shown more clearly in Figure 4.5(b).

The plots in Figure 4.6 are the same measurements as in Figure 4.5 except instead of a single sweep from the spectrum sensor there is an average of 5000 sweeps from the spectrum sensor. Figure 4.6(a) shows the two signals overlaid on top of each other and Figure 4.6(b) plots the difference between the two signals. The mean squared error magnitude here is 1.803, which is 37% lower than the single sweep error.

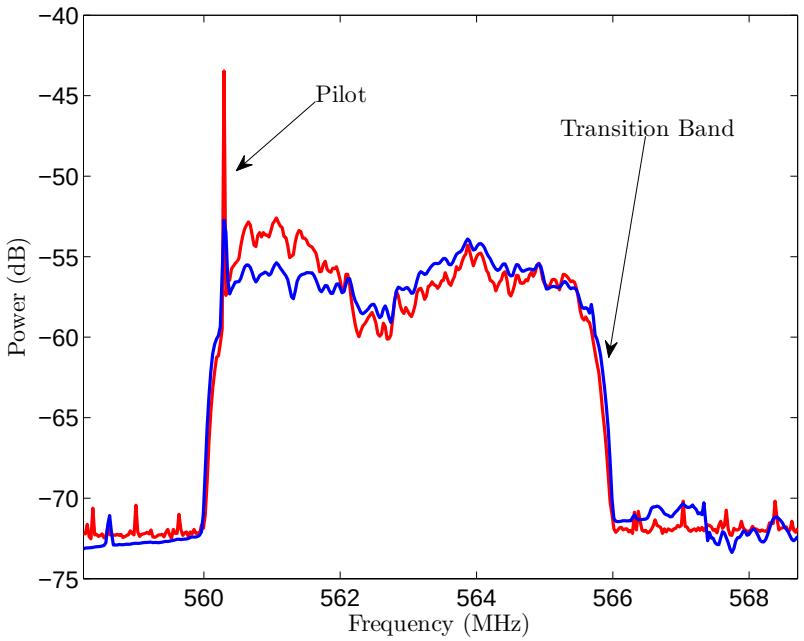


(a) The Spectrum Analyzer and a Single Sweep Spectrum Sensor Measurements Overlaid on Top of Each Other. The Spectrum Analyzer is in Red, the Spectrum Sensor in Blue.

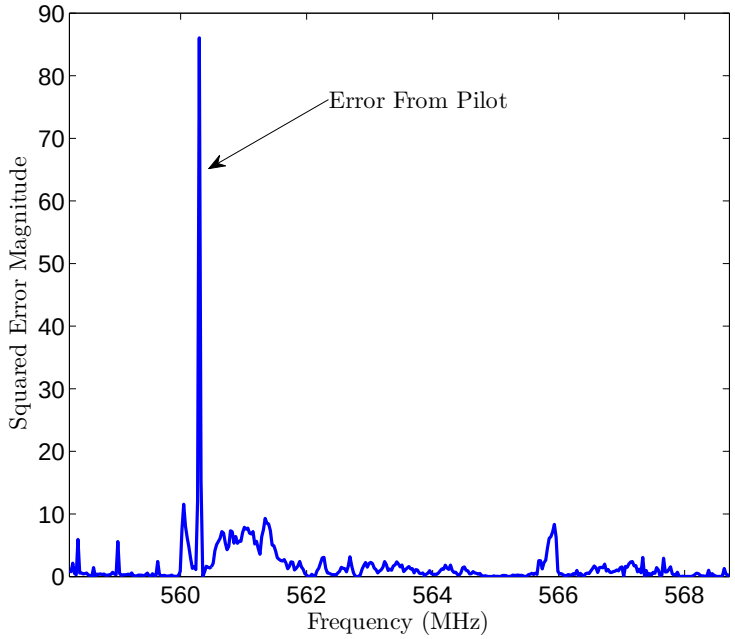


(b) The Squared Difference Between the Two Periodograms.

Figure 4.5: Periodogram Comparison.



(a) The Spectrum Analyzer and the Averaged Spectrum Sensor Measurements Overlaid on Top of Each Other. The Spectrum Analyzer is in Red, the Spectrum Sensor in Blue.



(b) The Squared Difference Between the Two Periodograms.

Figure 4.6: Periodogram Comparison.

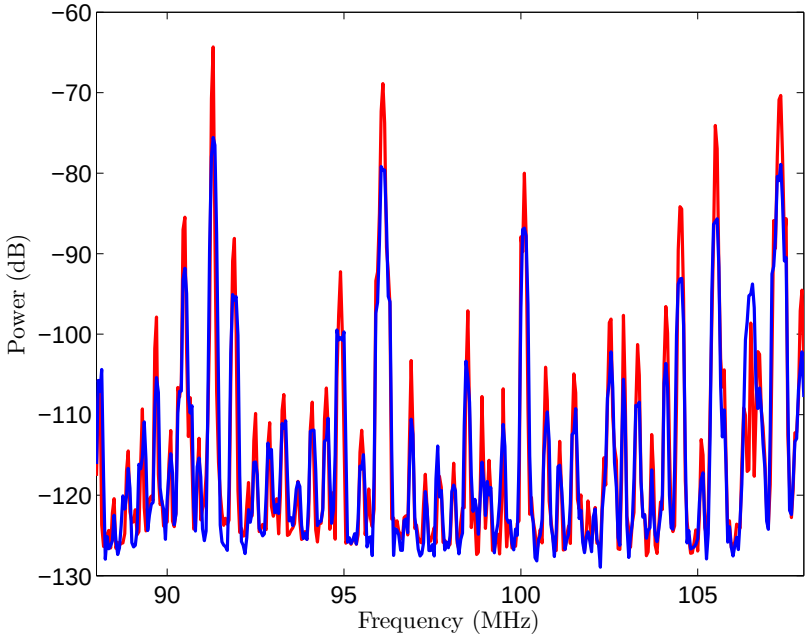
4.3.2 FM Radio Signal Comparison

This subsection compares the performance of a spectrum sensor to the spectrum analyzer in the FM radio band, 88 MHz to 108 MHz. The objective of this test is to compare the signals that the spectrum analyzer can distinguish from noise to the signals that the spectrum sensor can distinguish from noise. The FM radio band was chosen because it has a large collection of signals over a narrow band where the averaged received power varies.

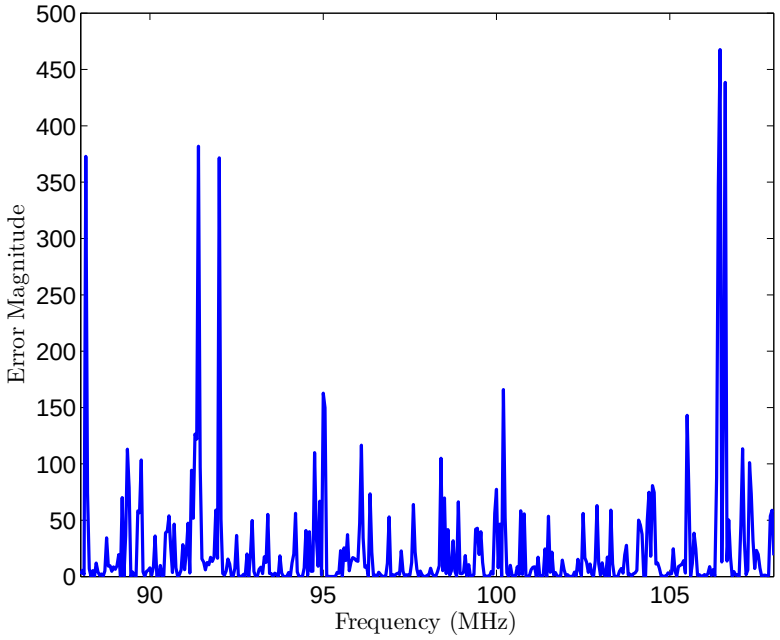
For this test the spectrum analyzer was configured in the same way as the previous test but with the exceptions of center frequency and span. In this, the center frequency was set to 98 MHz and the span was set to 20 MHz. The total band recorded was from 88 MHz to 108 MHz.

The spectrum sensor was also configured in a similar way the to the previous test. The first difference was the frequency range that was set was from 83 MHz to 116 MHz. The second difference was that the FFT size was set to 2048 points. Every other parameter was the same. The scan took approximately 2 hours, 20 minutes, 29 seconds.

Figure 4.7 shows two plots. Figure 4.7(a) shows the FM Radio spectrum as measured by the spectrum sensor (blue) and the spectrum analyzer (red). Figure 4.7(b) shows the squared error between the two measurements in the previous plot. The goal of this experiment was to compare the signals that the spectrum analyzer and spectrum sensor found. The spectrum sensor found all of the stations the spectrum analyzer found, so test was a success. However, as shown in the previous experiment, the spectrum sensor has large error values at the peaks or signals. The peaks in the squared error plot, Figure 4.7(b), all correspond to the center frequency of each radio station. Due to all the different FM radio stations over the 20 MHz band, the mean squared error is much higher for this scan in comparison to the TV scan. The squared error for the single sweep FM radio scan is 23.737. Finally, observe that some of the radio stations in Figure 4.7(a) are radiating out of band, one of them is labeled ‘WAAF.’



(a) One Sweep of the Adjusted Periodogram of the FM Radio Spectrum from the Spectrum Sensor (Blue) and the Spectrum Analyzer Measurement (Red).



(b) The Squared Difference Between the Spectrum Sensor and Spectrum Analyzer FM Radio Periodograms.

Figure 4.7: The FM Radio Spectrum.

4.3.3 Errors at Peaks

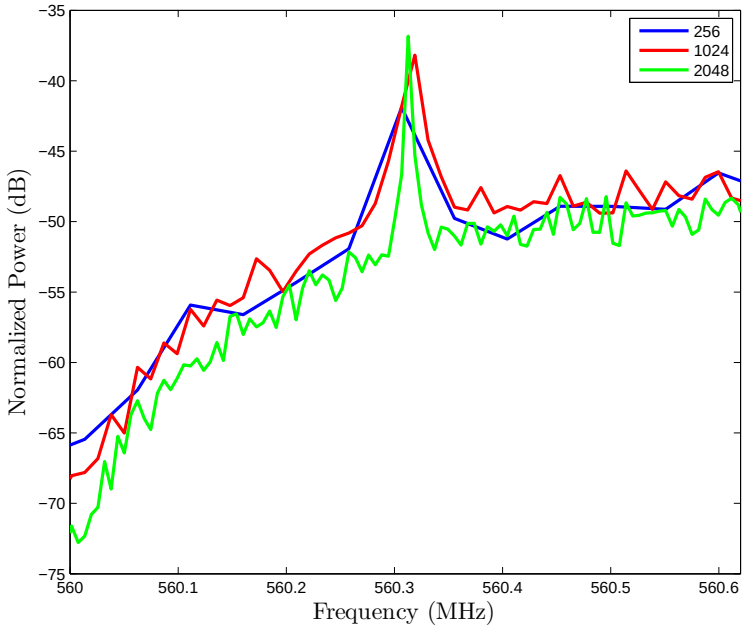
After conducting both the television signal comparison test and the FM radio signal comparison test it is clear that the spectrum sensor has large error values at peaks. The spectrum sensor records peaks that are significantly attenuated compared to the measurements made by the spectrum analyzer. There are several possible explanations for this.

The first possibility is that the spectrum sensor does not have as large a dynamic range as the spectrum analyzer. The dynamic range for the spectrum sensor is 84 dB [69] and the dynamic range for the spectrum analyzer is 158 dB [70] with configuration that the measurements were made with. The spectrum analyzer has a much larger dynamic range, which allows it to capture signals that have a larger range. It is possible that the spectrum sensor experiences clipping that the spectrum analyzer does not experience.

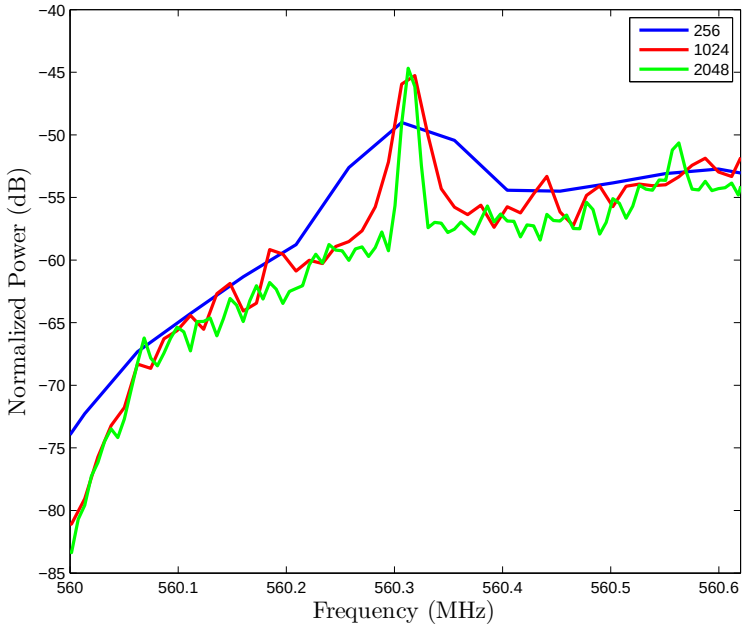
There are two more factors that can attenuate peaks: the window used and the FFT size. The window can affect the peak by smearing the peak into neighboring bins, so a window with a wider main lobe will smear the peak more. In these experiments a Blackman-Harris window was used. Blackman-Harris windows have a wide main lobe so the peak was smeared substantially. The FFT size can also affect the perceived magnitude of the peak. Since there are fewer bins, a lower FFT size is more likely to miss the peak than a larger FFT size.

Experiments confirm that both the window and FFT size affect the magnitude of the peak. Six additional tests were performed that show that the magnitude of the pilot tone from a television signal increases when the FFT increases and when a window with a narrow main lobe was used. Three FFT sizes were used for two different windows. The FFT sizes were 256, 1024 and 2048. The windows used were a Blackman-Harris window and a rectangular window. The Blackman-Harris window has a wide main lobe, so the peaks get smeared more. The rectangular window has a narrow main lobe so the peaks get smeared less. The results from these experiments are shown in Figure 4.8, where the rectangular window is shown in 4.8(a) and the Blackman-Harris window is used in 4.8(b). Each plot shows the three FFT sizes being used: 256 (blue), 1024 (red) and 2048 (red).

Table 4.2 shows the values at each peak for each FFT size and window type. The magnitude increases with the FFT size, which is what is expected. The magnitude is also



(a) The Pilot Tone of a Television Signal with a Rectangular Window.



(b) The Pilot Tone of a Television Signal with a Blackman-Harris Window.

Figure 4.8: Periodograms of a Television Signal Pilot Tone.

Table 4.2: Magnitude of the Pilot Tones from a Television Signal.

FFT Size	Magnitude (dB)	Magnitude (dB)
	Blackman-Harris	Rectangular
256	-49.017063	-41.899799
1024	-45.261200	-38.189610
2048	-44.680538	-36.845665

greater for rectangular window than the Blackman-Harris window for each FFT size.

4.4 Multi-Sensor Tests

This section characterizes tests conducted using multiple sensors. There are two categories of tests performed: the first is multi-sensor verification, and the second test is a demonstration of a distributed spectrum sensing system. The goal of the first test is to characterise the variation of the spectrum sensed between two independent sensors with the same input signal. The goal of the second test is to demonstrate a functional distributed spectrum sensing system using the system described in Chapter 3.

4.4.1 Multi Sensor Verification

This test is designed to characterize the variation in spectrum that two independent spectrum sensors measure. This is done in two steps, the first is to hold as many variables constant between the two sensors, the second step is to then use a more realistic configuration of spectrum sensors. In the first step the variables that are held constant are the antennas, the two spectrum sensors share the same antenna, where in the second step the two spectrum sensors use different antennas in order to create a realistic spectrum sensing system.

Both experiments were configured in the same manner other than the antenna configuration. The experiments were both performed using two USRP2s with WBX daughtercards, one host PC was Host 3, the other host PC was Host 4 with, both PCs are listed in Table 4.1. The transmitter was a USRP1 with the Flex 900 daughtercard. All the radios in these

experiment used the Vert 900 antenna, described in Section 3.2.1. The transmissions were a random stream of data that was modulated using a 16-QAM modulator and then filtered using a root-raised cosine filter, the transmission were centered at 940 MHz. There were two samples per symbol. The GNU Radio transmitter was created with GNU Radio Companion (GRC), a block a diagram of the transmitter is shown in Figure 4.9. **Shared Antenna**

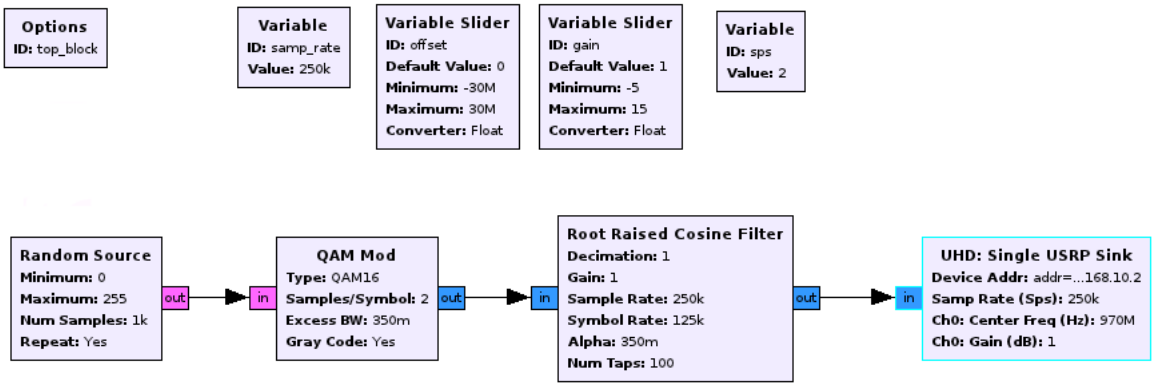


Figure 4.9: A GRC Block Diagram of the Transmitter.

In this experiment, two independent spectrum sensors shared a single single antenna via a Mini-Circuits ZFRSC-42 Power Splitter described in Section 4.3. The experimental setup is shown in Figure 4.10. The spectrum sensors were tuned from 937 MHz to 943 MHz, the decimation rate was set to 16, the FFT size was set to 256, the tune delay was set to 200 ms, the sensors took 5000 sweeps.

Figure 4.11 shows two measurements from a single sweep overlaid on top of each other as well as the squared error between the two. Also observe that a majority of the error comes from where there is a signal present. The blue periodogram, from Host 3, peaks at -7.253 dB and the red periodogram, from Host 4, peaks at -5.927 dB a difference of 1.326 dB. The mean squared error magnitude is 0.01021.

Figure 4.12 contains two plots. In the first plot, Figure 4.12(a), the average of 5000 sweeps from each sensor are shown overlaid on top of each other. Figure 4.12(b) shows the squared error between the two periodograms. The peak from Host 3, in blue, is -7.968 dB, the peak from Host 4 is, in red, is -7.030 dB, a difference of .938 dB. The mean squared

error magnitude is 0.009023.

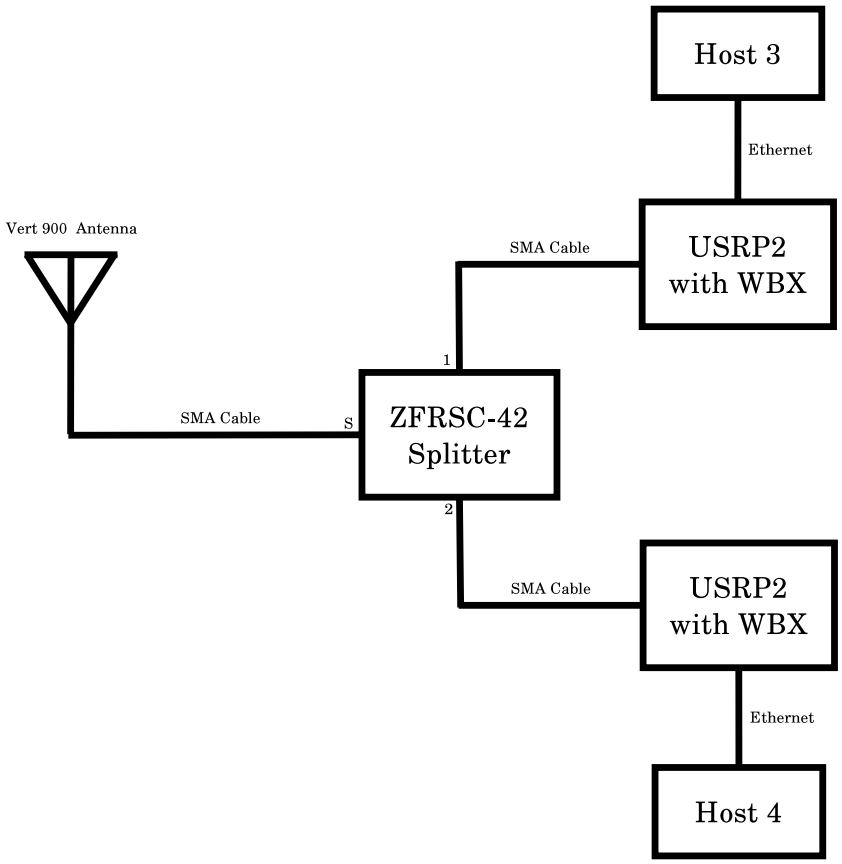
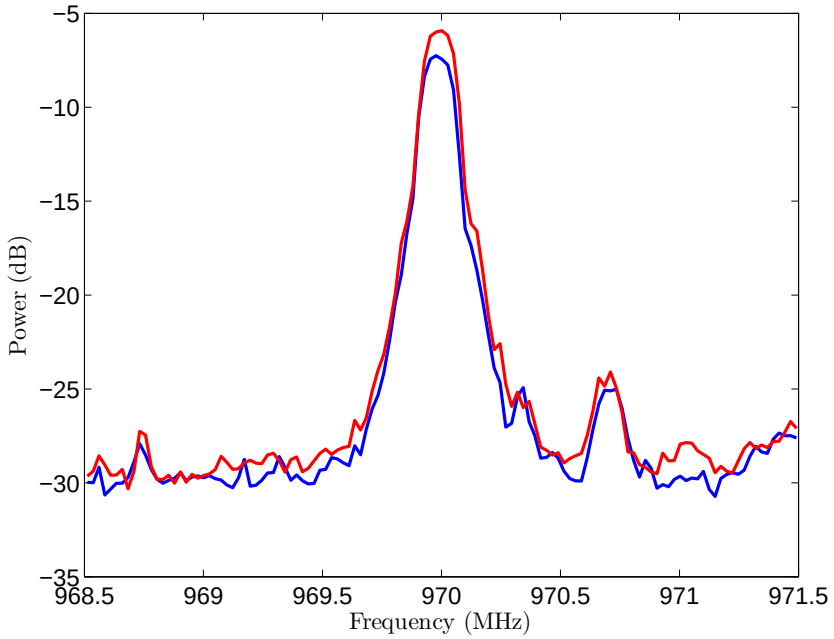


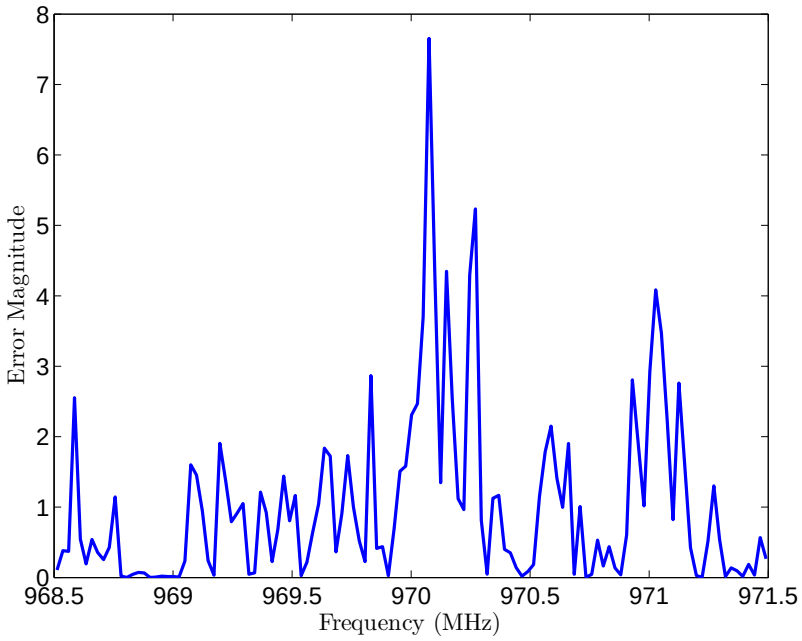
Figure 4.10: A Block Diagram of the Shared Antenna Experiment.

Different Antennas

This section details the same experiment performed as the previous section, but now with two separate antennas. The goal of this experiment is to characterize the difference in spectrum measured by two completely independent spectrum sensors while receiving approximately the same signal. This experiment was conducted in exactly the same fashion as the previous experiment with the exception of the sharing of antennas and use of the splitter. In this experiment two Vert 900 antennas were used. A block diagram of the configuration of this experiment is shown in 4.13. Note that the two antennas are separated by approximately 10 cm.

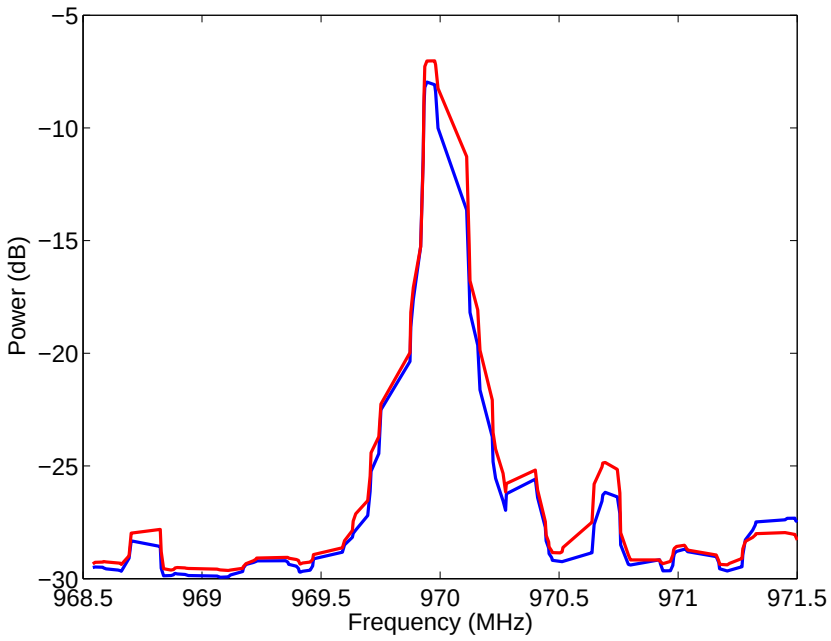


(a) A Single Sweep from the Two Sensors Sharing an Antenna overlaid on Top of Each Other.

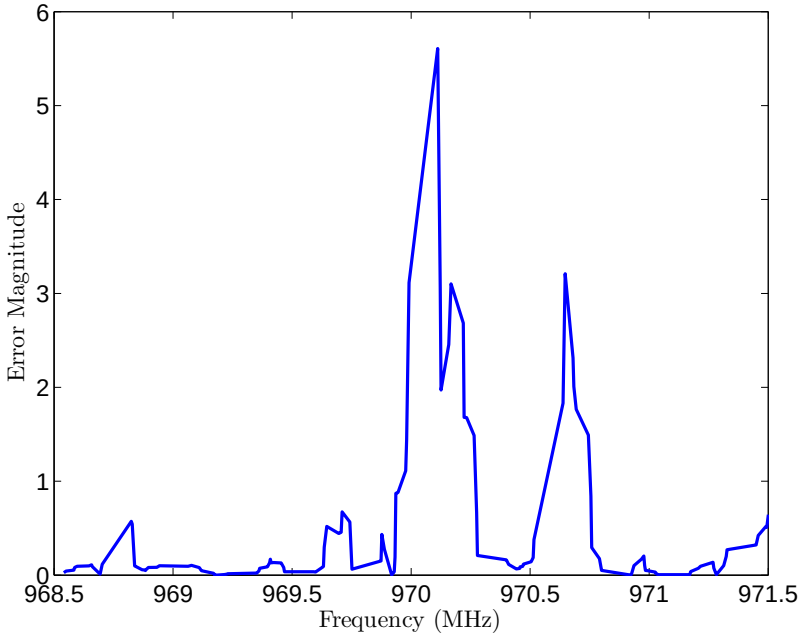


(b) The Squared Difference of the Two Periodograms.

Figure 4.11: A Comparison of the Two Spectrum Sensors Sharing the Same Antenna.



(a) An Average of 5000 Sweeps of Spectrum Sensed from the Two Sensors Overlaid on Top of Each Other.



(b) The Squared Difference of the Two Averaged Periodograms.

Figure 4.12: A Comparison of the 5000 Sweep Average from the Two Spectrum Sensors Sharing the Same Antenna.

Figure 4.14 shows two plots. The first a single sweep from each sensor overlaid on

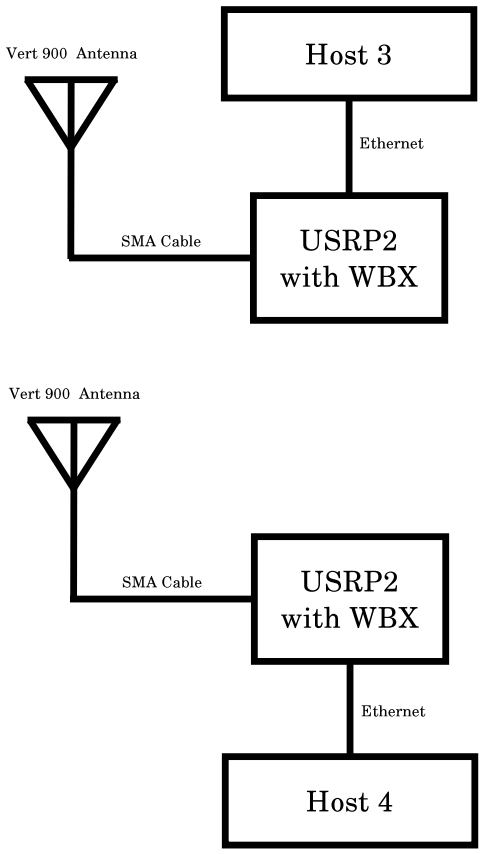
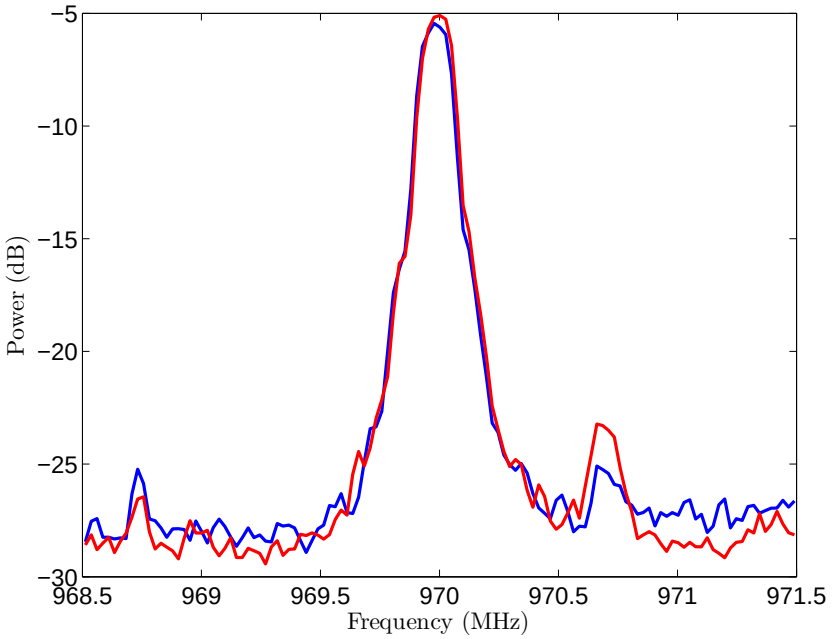


Figure 4.13: A Block Diagram of the Independent Antenna Experiment.

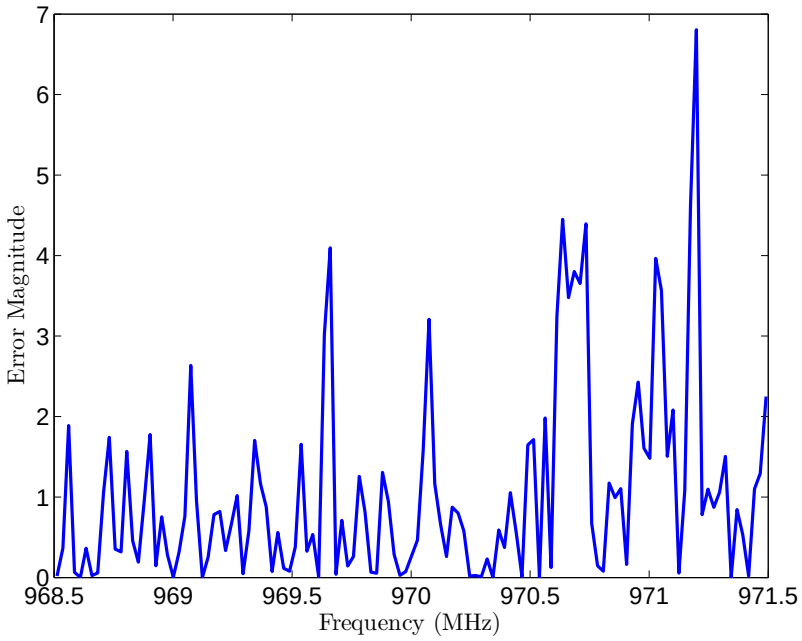
top of each other, the second is the squared error between the two. Also observe that a majority of the error comes from where there is a signal present. The blue periodogram, from Host 3, peaks at -5.172 dB and the red periodogram, from Host 4, peaks at -5.098 dB, a difference of 0.074 dB. The mean squared error magnitude is 0.0141.

Figure 4.15 shows the overlay of the 5000 sweep average from the two sensors as well as the squared error between the two periodograms. These plots show the same characteristics of the single sweep measurements, except that the error is greater. The peak from Host 3, in blue, is -5.742 dB, the peak from Host 4, in red, is -7.968 dB, a difference of 2.226 dB. The mean squared error magnitude is 0.01358. The mean squared error for the averaged sweeps is greater than the mean squared error for the single sweep, an unexpected result.

Upon further examination one can see that there appears to be an offset between the two periodograms, as shown in Figure 4.15. One can also see this offset in plot of the squared error, Figure 4.15(b), in this plot there is an offset for almost the entire 3 MHz band. Also, observe that in the squared error plot for Figure 4.11(b), the plot for the shared antenna, there is no offset. The source of the offset is likely to be from the different antennas. This offset could explain the why the error for the single sweep is less than the error for the averaged sweeps.

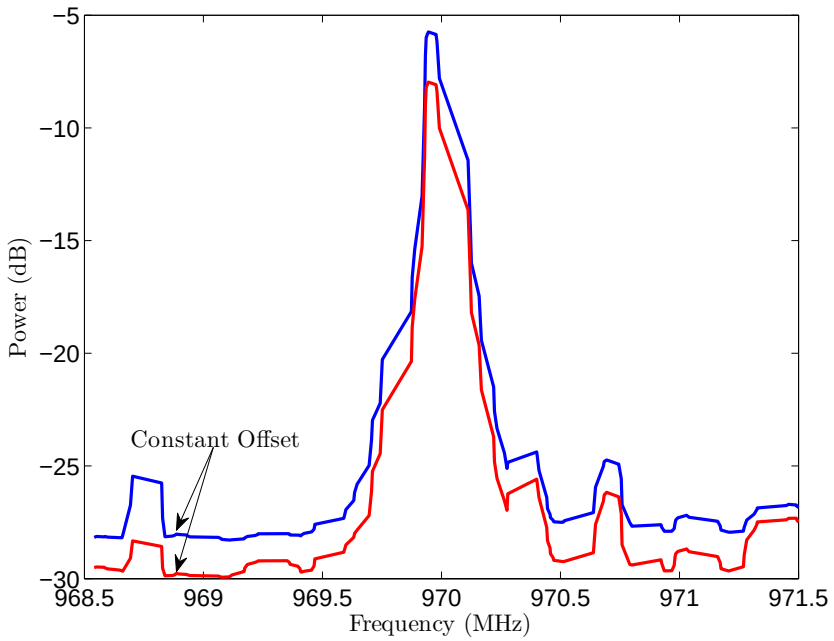


(a) A Single Sweep from the Two Sensors Using Different Antennas Overlaid on Top of Each Other.

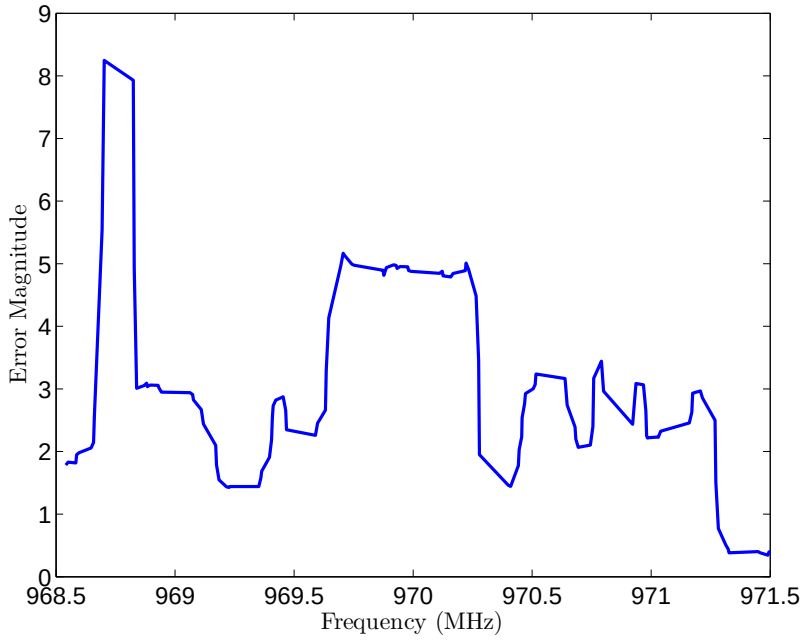


(b) The Squared Difference of the Two Periodograms.

Figure 4.14: A Comparison of the Two Spectrum Sensors Sharing the Same Antenna.



(a) An Average of 5000 Sweeps of Spectrum Sensed from the Two Sensors overlaid on Top of Each Other.



(b) The Squared Difference of the Two Averaged Periodograms.

Figure 4.15: A Comparison of the 5000 Sweep Average from the Two Spectrum Sensors Sharing the Same Antenna.

4.4.2 PSD Maps

This subsection discusses the creation of PSD maps generated using the spectrum sensing system. Included is a description of the experiment configuration and results, the PSD map.

This experiment took place in Olin Hall, a large lecture hall at WPI. Olin Hall was chosen because it is a large room with power supplies and it was available. The experiment consisted of three spectrum sensors and a transmitter. The spectrum sensors and transmitter were configured in exactly the same way as the previous experiment. The single exception to this is the change in the positions and number of the radios.

In this experiment the sensors were located at spatially varying positions as shown in Table 4.3. The measurement was setup so that the transmitter and Host 1 were approximately one foot apart. The transmitter and Host 1 were located at the front of the lecture hall. The two remaining spectrum sensors, Host 4 and Host 2, were located at the back of lecture hall on opposite sides of the room. The path from the transmitter to all three spectrum sensors was line of sight. The signal transmitter from the transmitter was centered at 970 MHz. The PSD map requires that only one frequency be observed at a time; the chosen frequency here is the 970 MHz. The PSD map in Figure 4.16(b) is a representation of how the power at a single frequency changes as a function of space.

Figure 4.16(b) shows that Host 1, the closest spectrum sensor to the transmitter, recorded the highest power. This figure also shows that two other spectrum sensors, which are located further way from the transmitter, recorded a lower measurement. In this experiment all the sensors were located indoors, that is, they were located inside the lecture hall. This result is expected: since Host 2 and Host 4 are further away from the transmitter they experience more path loss than Host 1. As a result of this path loss the two sensors record a lower power signal.

Table 4.3: The Spectrum Sensor Coordinates.

Radio	Longitude	Latitude
Transmitter	-71.808027	42.274764
Host 1	-71.808026	42.274764
Host 2	-71.808026	42.274801
Host 4	-71.807919	42.274675

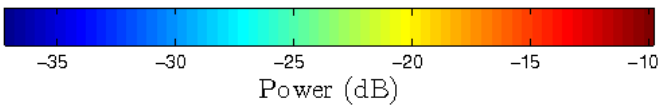
Figure 4.16 contains two maps. Figure 4.16(a) shows a map of Olin Hall with each sensor labeled. Figure 4.16(b) shows the same image but with the PSD map overlaid on top. The PSD map was created using the three measurements from the spectrum sensors. The power values for the remaining space were interpolated using the measurements from the spectrum sensor data. The data was interpolated using the MATLAB function `TriScatteredInterp` using linear interpolation. The MATLAB function `TriScatteredInterp` is used for interpolating data that is randomly scattered over a two dimensional surface [71].

Interpolating with this function has a clear problem: linear interpolation is used when free space path loss models are nonlinear. Free space path loss is the amount of loss, in dB, a receiver incurs in an environment with no impairments. This measure of loss does not include any fading or interference. An idealized free space path loss model for a 970 MHz signal is defined in Equation (4.3) [72]. This model is also idealized since it assumes both the transmitters had isotropic antennas. An isotropic antenna is an ideal antenna where power is radiated equally in all directions. In Equation (4.3) $L(d)$ is the free space path loss, in dB, and d is the distance between the transmitter and receiver, in kilometers. The amount of loss depends on the base 10 logarithm of the distance between the two radios. Linear interpolation is not the ideal interpolation method, however it is a first step in creating PSD maps from randomly scattered measured data.

$$L(d) = 92.135 + 20 \log_{10} d \tag{4.3}$$

(a) A Map of Olin Hall with the Spectrum Sensors Displayed.

(b) A Map of Olin Hall with the PSD Map Overlaid on Top.



(c) The Key for the PSD Map.

Figure 4.16: A Plain Map of Olin Hall and a PSD Map of Olin Hall.

4.5 Chapter Summary

This chapter focused on displaying the results and capabilities of the spectrum sensing system proposed in this document. The chapter starts with a description of the testing environment and then proceeds to develop a metric so these different tests can be compared to one another directly. Then the discussion moves on to the tests that were performed. The first test compares a single sensor to an instrument for verification. The next test compares the sensor to an instrument again and demonstrates a weakness in the sensor, which is inability to capture full peaks. Next, a test is performed that compares an identical input to two different sensors. Finally, in the last test a distributed measurement is made and the PSD map is generated.

Chapter 5

Conclusions

This chapter summarizes the work performed as part of this project and then suggests related research that can be performed in the future. The research achievements includes a description of the system developed, the capabilities of the system and the results it produced. The future work section includes a list of improvements that can be made to the system and some ideas that can start future research projects.

5.1 Research Achievements

As part of this work the following was achieved:

- A wideband spectrum sensing system was developed. This spectrum sensing system can operate over frequencies from 50 MHz to 2.2 GHz.
- A distributed spectrum sensing system was developed. This spectrum sensing system can scale to as many nodes as the user wishes to use. The system can also collect the measurements from each spatially varying sensor and create a PSD map.
- A practical spectrum sensing system was created. This spectrum sensing system is affordable, each node costs approximately $ 2000 USD.
- Tools for analyzer the spectrum sensing data were developed. Users may use individual periodograms, averaged periodograms to observer how spectrum changes as a function

of frequency. Users may also use spectrograms to observe how spectrum changes with time.

5.2 Future Work

Future research activities that are related to this work are discussed in this section.

- Re-implement this system such that realtime measurements can be observed. An implementation that allows for realtime measurements to be made will allow users to observe each individual sweep as they are completed. Currently, the user must wait for all the sweeps to finish before observing the data.

- Implement a system without a master node, that is a decentralized system. A decentralized system is a more robust system since all measurements no longer need to report all their data to the master node. A decentralized system is more difficult to implement since nodes must locate each other and the volume of data that must be shared increases dramatically if every node is to have a copy of every other nodes data.

- Redeploy this system on the USRP E100. The USRP E100 is less expensive and more compact since it combines the host PC and the software radio into one package. The USRP E100 fits this project very well since it is easier to deploy (less components), has a smaller package, and is less expensive.

- Implement a system that can share data using wireless data sharing service. Sensors sharing data using a wireless data service is advantageous since this eliminates the need for a preexisting network. The challenge of this is system is that any wireless data sharing occupies spectrum the author must then decide to measure this spectrum or attempt to avoid those transmissions.

- Implement other multi-sensor architectures. In this project the technology that is used to observe the spatially varying data is the PSD map. Future work may investigate other techniques to utilize spatially diverse spectrum measurements. For

example, researchers may attempt to use an energy detector that sensors uncertain measurements.

- Test with a large number of nodes. This work used only four nodes for testing, other works that may want to observe how the data changes with a large number of nodes.
- Measure the effects of sensor network size and arrangement. Future researchers can find what the returns are to adding sensors to the sensor network and when adding sensors stops adding to the quality of the data gathered. Researchers may also wish to see how sensor arrangement affects the gathered data.
- How the needs of the sensor network changes with the environment. Future researchers may wish to find how the number and arrangement of sensors changes in differing environments while attaining the same quality of data. That is, researchers can find how the number of sensors that are needed to measure spectrum use changes in a suburb compared to a city.
- Create PSD maps using accurate interpolation methods. The interpolation methods used to create PSD maps in this project do not match real models for RF propagation. A future work could implement a interpolation algorithm that applies already existing RF propagation models, such as Longley-Rice or Egli models, to the algorithm to get more accurate results.

Appendix A

USRP Spectrum Sense Options

```
Usage: usrp_spectrum_sense.py [options] min_freq max_freq

Options:
  -h, --help            show this help message and exit
  -g GAIN, --gain=GAIN  set gain in dB (default is midpoint)
  --tune-delay=SECS     time to delay (in seconds) after changing frequency
                        [default=0.02]
  --dwell-delay=SECS    time to dwell (in seconds) at a given frequncy
                        [default=0.01]
  -F FFT_SIZE, --fft-size=FFT_SIZE
                        specify number of FFT bins [default=256]
  -d DECIM, --decim=DECIM
                        set decimation to DECIM [default=-1]
  -s SAMPLE_RATE, --sample-rate=SAMPLE_RATE
                        set sample rate [default=1000000.0]
  --real-time           Attempt to enable real-time scheduling,
                        [default=False]
  -n NSWEEPS, --nSweeps=NSWEEPS
                        Number of sweeps, [default=none]
  -a, --recordMax       Record Max values of FFTs, [default=False]
  -c NCHANNELS, --nChannels=NCHANNELS
```

```
                      Sets the number of channels, [default=1]
-D DURATION, --duration=DURATION
                      Set the duration of the scan in seconds, minutes,
                      hours, days, or weeks, [default=none]
-N NOTE, --note=NOTE  A note about the particular scan, [default=none]
--daughter-card=DAUGHTER_CARD
                      Set the daughtercard, [default=wbx]
--ref                 If connected to a 10 MHz reference, enable this option
                      to synchronize to the reference clock
-v, --verbose         Verbose, [default=False]
--antenna=ANTENNA     Select the antenna port on the USRP Daughtercard,
                      [default=TX/RX]
```

Appendix B

makeKML.py

```
#1/usr/bin/python

import h5py
from optparse import OptionParser
from os import listdir, getcwd
from sys import exit
import subprocess as sp

class kml():
    def __init__(self, hdf5file=None):
        self.kml_start = u'<?xml version="1.0" encoding="UTF-8"?>\n'+\
            '<kml xmlns="http://www.opengis.net/kml/2.2">\n<Folder>'
        self.kml_end   = '</Folder>\n</kml>\n'
        self.endLen    = len(self.kml_end)
        self.kml       = self.kml_start + self.kml_end
        self.nSensors  = None

        self.hdf5file = hdf5file

        if self.hdf5file is not None:
            self.f = h5py.File(self.hdf5file)
            self.sensor_info = self.f.get('Sensor Info')
```

```
        self.__findNSensors__()
        self.addAllSensors()

    def addAllSensors(self):
        props = map(None, self.sensor_info[:,0])
        for kk in xrange(self.nSensors):

            # make list from np array
            values = map(None, self.sensor_info[:,kk+1])
            # make the two lists into a dict
            sensor_props = dict(zip(props, values))
            self.addPlace(sensor_props)

    def addPlace(self, sensor_props, desc=''):
            if len(desc) > 0:
                desc = ', ' + desc
            # this is hard to read, but this has to fit
            # on a 8.5 x 11 page...
            newPlace = \
            u'    <Placemark>\n'+\
             '        <name>' +\
                      str(sensor_props['NAME']) +\
                      '</name>\n'+\
             '        <description>Sensor ' +\
                      str(sensor_props['NUMBER']) +\
                      ' of ' + str(sensor_props['TOTAL']) +\
                      str(desc) + '</description>\n'+\
             '        <Point>\n'+\
             '            <coordinates>' +\
                          str(sensor_props['LNG']) + ',' +\
                          str(sensor_props['LAT']) +\
                          ',0</coordinates>\n'+\
             '        </Point>\n'+\
             '    </Placemark>\n'
```

```
        end = len(self.kml)
        self.kml = self.kml[0:end-self.endLen] +\
                   newPlace +\
                   self.kml[end-self.endLen:end]

def write(self, kmlFilename=None):
    if kmlFilename is None:
        kmlFilename = self.hdf5file.replace('.hdf5','.kml')

    self.kmlFilename = kmlFilename
    self.kmlfile = file(self.kmlFilename, 'w')
    self.kmlfile.write(self.kml)
    self.kmlfile.close()

def file(self, hdf5file):
    self.hdf5file = hdf5file
    self.f = h5py.File(self.hdf5file)

def addImage(self,\
             image,\
             name='',\
             desc='',\
             north='',\
             south='',\
             east='',\
             west=''):
    # find a point in the kml string that we can enter
    imageStr = u'<GroundOverlay> \n'+\
                '    <name>'+ name + '</name>\n'+\
                '    <description>' + desc + '</description>\n'+\
                '    <color>90ffffff</color>\n'+\
                '    <Icon>\n'+\
                '        <href>' + image + '</href>\n'+\
                '    </Icon>\n'+\
```

```
                '    <LatLonBox>\n'+\
                '        <north>' + str(north) + '</north>\n'+\
                '        <south>' + str(south) + '</south>\n'+\
                '        <east>' +  str(east) + '</east>\n'+\
                '        <west>' +  str(west) + '</west>\n'+\
                '        <rotation>0</rotation>\n'+\
                '    </LatLonBox>\n'+\
                '</GroundOverlay>\n'

    insertPoint =  self.kml.find('</Folder>')
    self.kml = self.kml[0:insertPoint] + imageStr + '</Folder>\n</kml>'


def close(self):
    self.f.close()

def __findNSensors__(self):
    if self.hdf5file is None:
        self.nSensors = None
        return

    si = self.f.get('Sensor Info')
    print si
    for kk in range(si.shape[0]):
        if si[kk, 0]=='TOTAL':
            try:
                self.nSensors = int(si[kk,1])
                return
            except:
                print 'HDF5 File Damanaged'
                exit(-1)

def __del__(self):
    self.f.close()
```

```
def isHDF5(s):
    end=len(s)
    if s[end-4:end] == 'hdf5':
        return True
    else:
        return False

if __name__ == "__main__":
    sweepsDir = 'test_sweeps'
    cwd = getcwd()
    end = len(cwd)
    if cwd[end-5:end] == 'utils':
        sweepsDir = '../' + sweepsDir
    files = listdir(sweepsDir)
    files = filter(isHDF5, files)
    files.sort()

    for ii in files:
        if ii.find('combined') > -1:
            default_file = ii

    parser = OptionParser()
    parser.add_option('-f',\
                      '--filename',\
                      type='str',\
                      default=default_file,\
                      help='Select the file to generate'+\
                           ' KML for, [default=%default]')

    (options, args) = parser.parse_args()

    k = kml(sweepsDir + '/' + options.filename)
```

```
# make the psd map
print 'Using MATLAB to make the PSD map'
matlab_string=('/opt/r2010b/bin/matlab',\
               '-nosplash',\
               '-nodesktop',\
               '-r makeMap')
matlab = sp.Popen(matlab_string, stdout=sp.PIPE)
matlab.wait()

msg = matlab.communicate()[0].split()
msg_len = len(msg)
print msg
print 'MATLAB done'

for ii in xrange(msg_len):
    if msg[ii].find('Latitudes') >= 0:
        lats = (msg[ii+1], msg[ii+2])
    elif msg[ii].find('Longitudes') >= 0:
        lngs = (msg[ii+1], msg[ii+2])
    elif msg[ii].find('Filename:') >= 0:
        mapname = msg[ii+1] + '.png'

print lats[0], lats[1]
print lngs[0], lngs[1]
print mapname

k.addImage(mapname,\
           'PSD Map',\
           'PSD Map Overlayed on My Sensor Network',\
            lats[0], lats[1], lngs[0], lngs[1])
k.write()
```

Appendix C

HDF5 Combiner

```
#!/usr/bin/python

import h5py, os
from string import letters
from optparse import OptionParser
hdf5_str = h5py.new_vlen(str)

def isHDF5(s):
    l = len(s)
    s=s.lower()
    if s[l-5:l] == '.hdf5':
        return True
    else:
        return False

def isFirstSensor(s):
    if s.find('-combined.') >= 0:
        return False

    # read the HDF5 file method
    #f = h5py.File(s)
    #for ii in f.items():
```

```
    #    if ii[0] == 'Sensor Info':
    #        for kk in ii[1]:
    #            if kk[0] == 'NUMBER':
    #                if kk[1] == '1':
    #                    f.close()
    #                    return True
    #f.close()

    # read the filename way
    if s.find('-a.hdf5') >= 0:
        return True
    else:
        return False


if __name__ == "__main__":

    parser = OptionParser()
    parser.add_option('-d',
                      '--directory',
                      type='string',
                      default='test_sweeps',
                      help='Directory to search for HDF5 '+\
                           'files to combine [default=%default]')

    (options, args) = parser.parse_args()

    sweepsDir  = options.directory
    sensorInfo = 'Sensor Info'
    scanInfo   = 'Scan Info'
    times      = 'Times'

    filelist = os.listdir(sweepsDir)
```

```
# first, delete all items in filelist
# that are not HDF5 files
filelist = filter(isHDF5, filelist)

filelist_full = []
for ii in filelist:
    filelist_full.append(sweepsDir+'/'+ii)

# now remove HDF5 files that aren't the sensor number 1
filelist_full = filter(isFirstSensor, filelist_full)

for files in filelist_full:

    # just get the number of sensors
    try:
        f = h5py.File(files, 'r')
    except:
        print files + ' failed to open'
        break

    try:
        info = f.get(sensorInfo)
    except:
        print 'Cannot Find Dataset \'Sensor Info\''
        break

    try:
        scans_group = f.get('scans')
    except:
        print 'Cannot Find Group \'scans\''
        break

    totalSensors  = int(info[2,1])
    sweepsPerScan = int(scans_group.items()[0][1].shape[0])
```

```
totalScans    = len(scans_group.keys())

combinedFilename = files.replace('-a.', '-combined.')
cf=h5py.File(combinedFilename , 'w') # combined file

nStartTime      = 7
nEndTime        = 7
nEstTime        = 7
nScanInfo       = 13
nSensorInfo     = 5

startTime       = cf.create_dataset('Time',
                                    (nStartTime, totalSensors+1),
                                    dtype=hdf5_str)
endTime         = cf.create_dataset('End Time',
                                    (nEndTime, totalSensors+1),
                                    dtype=hdf5_str)
estEndTime      = cf.create_dataset('Estimated End Time',
                                    (nEstTime, totalSensors+1),
                                    dtype=hdf5_str)
scanInfo        = cf.create_dataset('Scan Info',
                                    (nScanInfo,totalSensors+1),
                                    dtype=hdf5_str)
sensorInfo      = cf.create_dataset('Sensor Info',
                                    (nSensorInfo,totalSensors+1),
                                    dtype = hdf5_str)
note            = cf.create_dataset('Note',
                                    (1,totalSensors),
                                    dtype = hdf5_str)
scans           = cf.create_group('scans')
for ii in xrange(totalScans):
    scans.create_dataset('sweep-' + str(ii).zfill(6),
                         (sweepsPerScan,totalSensors+1),
                         dtype=float)
```

```
startTime[0,0]  = 'Day'
startTime[1,0]  = 'Month'
startTime[2,0]  = 'Year'
startTime[3,0]  = 'Day of Week'
startTime[4,0]  = 'Hour'
startTime[5,0]  = 'Minute'
startTime[6,0]  = 'Second'
endTime[0,0]    = 'Day'
endTime[1,0]    = 'Month'
endTime[2,0]    = 'Year'
endTime[3,0]    = 'Day of Week'
endTime[4,0]    = 'Hour'
endTime[5,0]    = 'Minute'
endTime[6,0]    = 'Second'
estEndTime[0,0] = 'Day'
estEndTime[1,0] = 'Month'
estEndTime[2,0] = 'Year'
estEndTime[3,0] = 'Day of Week'
estEndTime[4,0] = 'Hour'
estEndTime[5,0] = 'Minute'
estEndTime[6,0] = 'Second'
scanInfo[0,0]   = 'Minimum Frequency'
scanInfo[1,0]   = 'Maximum Frequency'
scanInfo[2,0]   = 'Antenna'
scanInfo[3,0]   = 'Name'
scanInfo[4,0]   = 'ADC Rate'
scanInfo[5,0]   = 'Decimation Rate'
scanInfo[6,0]   = 'Sampling Rate'
scanInfo[7,0]   = 'IP Address'
scanInfo[8,0]   = 'Receive Buffer Size'
scanInfo[9,0]   = 'FFT Size'
scanInfo[10,0]  = 'Minimum Allowable Frequency'
scanInfo[11,0]  = 'Maximum Allowable Frequency'
scanInfo[12,0]  = 'Initially Set Gain'
```

```
sensorInfo[0,0] = 'LAT'
sensorInfo[1,0] = 'LNG'
sensorInfo[2,0] = 'TOTAL'
sensorInfo[3,0] = 'NAME'
sensorInfo[4,0] = 'NUMBER'

newStr = '-a.hdf5'
oldStr = '-a.hdf5'
for ii in range(totalSensors):
    try:
        filename = files.replace(oldStr, newStr)
        print 'Copying File: ' + filename
        f = h5py.File(filename, 'r')
        newStr = '-' + letters[ii+1] + '.hdf5'
    except:
        print files + ' failed to open'
        break
    # find all the datasets and associate
    # them with their order or appearance
    startTime_old  = f.get('Time')
    endTime_old    = f.get('Actual End Time')
    estEndTime_old = f.get('Estimated End Time')
    scanInfo_old   = f.get('Scan Info')
    sensorInfo_old = f.get('Sensor Info')
    note_old       = f.get('Note')

    # There might be a more sophisticated way to
    # copy all this data.  h5py.File.copy doesn't
    # provide exactly what I need

    for kk in xrange(nStartTime):
        startTime[kk,ii+1] = startTime_old[kk,1]
    for kk in xrange(nEndTime):
        endTime[kk,ii+1] = endTime_old[kk,1]
```

```
            for kk in xrange(nScanInfo):
                scanInfo[kk,ii+1] = scanInfo_old[kk,1]
            for kk in xrange(nSensorInfo):
                sensorInfo[kk,ii+1] = sensorInfo_old[kk,1]
            for kk in xrange(nEndTime):
                estEndTime[kk,ii+1] = estEndTime_old[kk,1]
            note[0,ii] = note_old[0]

            # grp => group, ds => dataset
            scan_grp_old = f.get('scans')

            # FIXME change the way that the frequencies col is copied,
            # this is redundant
            # if we're on the first file copy the frequencies column
            #if ii == 0:
            #   for kk in xrange(sweepsPerScan):
            #       sweep_str = 'sweep-' + str(ii).zfill(6)
            #       scan_ds_old = scan_grp_old.get(sweep_str)
            #       scan_ds_tmp = scans.get(sweep_str)
            #       scan_ds_tmp[:,0] = scan_ds_old[:,0]

            for kk in xrange(totalScans):
                sweep_str = 'sweep-' + str(kk).zfill(6)
                scan_ds_old = scan_grp_old.get(sweep_str)
                scan_ds_tmp = scans.get(sweep_str)
                scan_ds_tmp[:,0] = scan_ds_old[:,0]
                scan_ds_tmp[:,ii+1] = scan_ds_old[:,1]

            f.close()


    # Used to filter out unwanted files this way,
    # but I like the new way better
    #
```

```
# remove all the duplicate scans, ie if we have
#   sweep_2010-11-02_11-48-20_01.hdf5
#   sweep_2010-11-02_11-48-20b.hdf5
#   sweep_2010-11-02_11-48-20c.hdf5
#   sweep_2010-11-02_11-48-20d.hdf5
# then remove the first three from the list
# this is so we combine these scans four times
#filelist.sort()
#last = filelist[-1]
#for i in range(len(filelist)-2, -1, -1):
#    if last[0:25] == filelist[i][0:25]:
#        del filelist[i]
#    else:
#        last = filelist[i]
```

Appendix D

Frequency Resolution Derivation

The Spectrum Sensor

- The sampling rate is set to 400 kHz, $f_s = 400$ kHz.
- The USRP2 performs quadrature sampling.
- Only $\frac{3}{4}$ of the frequency domain data is used, the lowest $\frac{1}{8}$ and highest $\frac{1}{8}$ of the frequency domain samples are discarded.
- The Nyquist Rate is $\frac{1}{2}$, $f_N = \frac{1}{2}$.

Considering these four items one may calculate the frequency resolution, f_r:

$$\begin{aligned} f_r &= 2 \cdot f_s \cdot \frac{3}{4} \cdot f_N \text{ Hz} && \text{(D.1)} \\ &= 2 \cdot 400,000 \cdot \frac{3}{4} \cdot \frac{1}{2} \text{ Hz} && \text{(D.2)} \\ &= 300\text{kHz} && \text{(D.3)} \end{aligned}$$

The Spectrum Analyzer

- The span was set to 20 MHz.
- The number of points in the trace was 401.

The frequency resolution, f_r, is simply:

$$f_r = \frac{20 \text{ MHz}}{401} \tag{D.4}$$

$$= 49.875 \text{ kHz} \tag{D.5}$$

$$\approx 50 \text{ kHz} \tag{D.6}$$

Appendix E

Minimum Squared Error Derivation

In this derivation S_i the vector from the spectrum sensor and A_i is the vector from the spectrum analyzer. The mean squared error is being minimized with respect to a and b.

1. Equation (E.1), the equation to be minimized.

$$\begin{aligned} \text{MSE} &= \frac{1}{N}\sum_{i=0}^{N-1}\left(A_i - (a \cdot S_i + b)\right)^2 && \text{(E.1)} \\ &= \frac{1}{N}\sum_{i=0}^{N-1} A_i^2 + a^2 S_i^2 + b^2 + -2aA_iS_i + -2bA_i + 2abS_i && \text{(E.2)} \end{aligned}$$

2. Equation (E.3), the partial derivative of Equation (E.1) with respect to a.

$$\frac{\partial \text{MSE}}{\partial a} = \frac{1}{N}\sum_{i=0}^{N-1} 2aS_i^2 + -2A_iS_i + 2bS_i \qquad \text{(E.3)}$$

   (a) Equation (E.4), the second order partial derivative of Equation (E.1) with respect to a. Equation (E.4) is always positive making this a minimization of Equation (E.1).

$$\frac{\partial^2 \text{MSE}}{\partial b^2} = \frac{1}{N}\sum_{i=0}^{N-1} 2S_i^2 > 0 \qquad \text{(E.4)}$$

3. Equation (E.5), the partial derivative of Equation (E.1) with respect to b

$$\frac{\partial \text{MSE}}{\partial b} = \frac{1}{N} \sum_{i=0}^{N-1} 2b + -2A_i + 2aS_i \tag{E.5}$$

   (a) Equation (E.6), the second order partial derivative of Equation (E.1) with respect to b. Equation (E.6) is always positive making this a minimization of Equation (E.1).

$$\frac{\partial^2 \text{MSE}}{\partial b^2} = \frac{1}{N} \sum_{i=0}^{N-1} 2 > 0 \tag{E.6}$$

4. Set Equation (E.3) and Equation (E.5) to zero and then rearrange.

$$\sum_{i=0}^{N-1} 2A_i S_i = \sum_{i=0}^{N-1} 2aS_i + \sum_{i=0}^{N-1} 2bS_i \tag{E.7}$$

$$\sum_{i=0}^{N-1} 2A_i = \sum_{i=0}^{N-1} 2b + \sum_{i=0}^{N-1} 2aS_i \tag{E.8}$$

5. Put Equation (E.3) and Equation (E.5) into matrix form and solve for a and b:

$$\begin{bmatrix} \sum_{i=0}^{N-1} 2S_i^2 & \sum_{i=0}^{N-1} 2S_i \\ \sum_{i=0}^{N-1} 2S_i & \sum_{i=0}^{N-1} 2 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N-1} 2A_i S_i \\ \sum_{i=0}^{N-1} 2A_i \end{bmatrix} \tag{E.9}$$

Appendix F

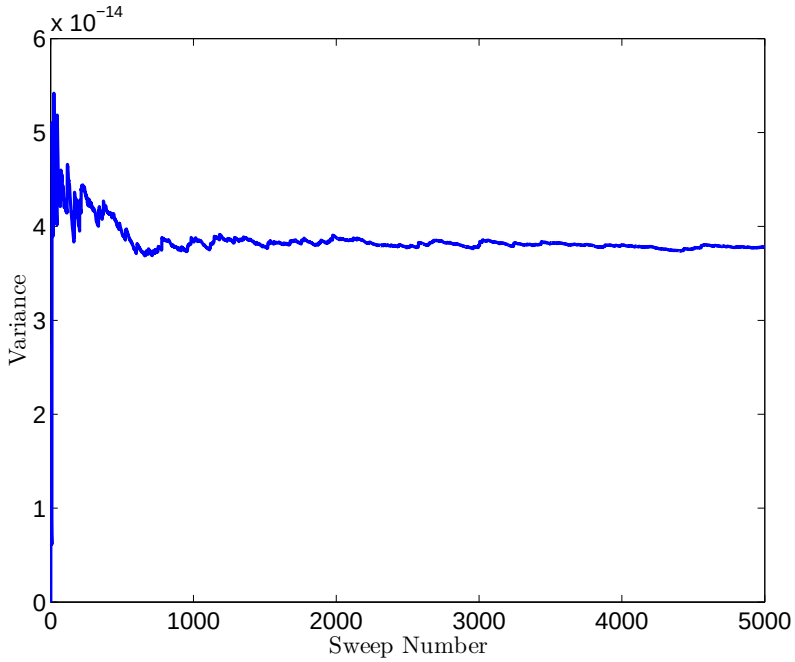
Sweep Selection

This appendix will discuss the selection of 5000 for a sufficient number of sweeps. The technique used to verify that 5000 sweeps is sufficient is to plot the variance as a function of the number of sweeps at a particular frequency.

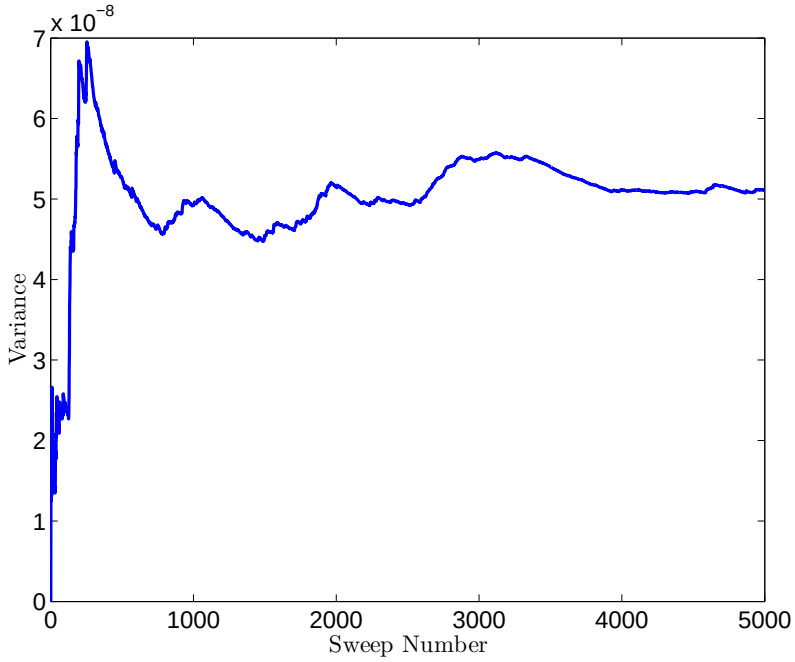
For example, call this function $g(f, n)$ where $g(f, n)$ is the variance for n sweeps at frequency f. When $g(f, n)$ converges as n increases with f held constant, the number of sweeps is sufficient.

To show that 5000 sweeps is sufficient three frequencies were chosen from the spectrum sensor measurement from the TV periodogram in Figure 4.4(a). The three selections are from noise, the pilot tone, and a from the signal. These frequencies are 558.7 MHz, 560.31 MHz, and 563.5 MHz respectively.

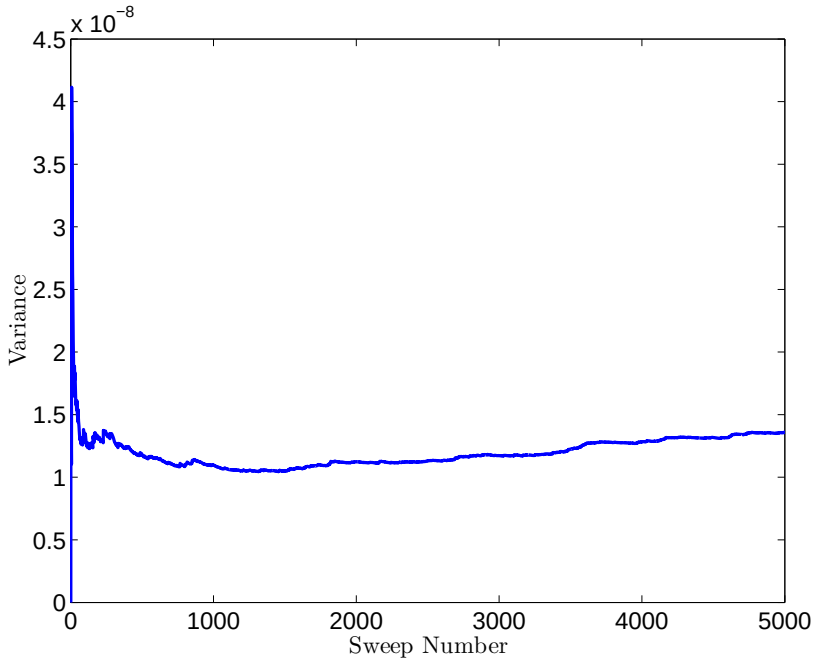
A plot is shown for each of frequency bin starting with the noise variance plot in Figure F.1(a). Next is the variance plot of from the carrier in Figure F.1(b). Finally the variance plot for the TV signal is shown in Figure F.0(c). In all three plots the variance converges to one value meaning that adding sweeps does not change the variance periodogram.



(a) The Variance Plot for the Noisy Signal



(b) The Variance Plot for the Pilot Tone



(c) The Variance Plot from the TV Signal

Figure F.0: The Variance Plot for Several Signals

Bibliography

[1] The Federal Communications Commission, "United States Frequency Allocations," Oct. 2003.

[2] Roke Manor, "The UK Frequency Allocations," 2007.

[3] GNU Radio, "Usrp2 general faq." `http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ`. [Online; accessed Nov., 10, 2010].

[4] J. Mitola, "The Software Radio Architecture," *IEEE Communications Magazine*, pp. 26 – 38, May 1995.

[5] A. S. on Spectrum Management in Cognitive Radio Networks, "I. akyildiz and w. lee and m. vuran and s. mohanty," *IEEE Communications Magazine*, Apr. 2008.

[6] Q. Zhao and B. Sadler, "A Survey of Dynamic Spectrum Access," *IEEE Signal Processing Magazine*, May 2007.

[7] J. Mitola, "Software Radios: Wireless Architecture for the 21st Century," *Proceedings of the IEEE International Workshop on Mobile Multimedia Communications*, pp. 3 – 10, 1999.

[8] M. McHenry and D. Roberson, "Spectrum Occupancy Measurements," tech. rep., Shared Spectrum Company, Chicago, Illinois, Nov. 2005.

[9] J. MacDonald, "A Survey of Spectrum Utilization in Chicago," tech. rep., Wireless Interference Laboratory of the Illinois Institute of Technology, Mar. 2007.

[10] H. Islam, C. L. Koh, S. W. Oh, X. Qing, Y. Lai, C. Wang, Y. Liang, B. E. Toh, F. Chin, G. L. Tan, and W. Toh, "Spectrum Survey in Singapore: Occupancy Measurements and Analyses," *CrownCom*, May 2008.

[11] The Shared Spectrum Company, "General Survey of Radio Frequency Bands - 30 MHz to 3 GHz," August 2010. Spectrum Survey in Vienna, VA, USA.

[12] A.P. Iyer, et. al., "SpecNet: Spectrum Sensing Sans Frontières," *8th USENIX Symposium on Networked Systems Design and Implementation*, Mar. 2011.

[13] T. Yücek and H. Arslan, "A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 116 – 130, 2009.

[14] H. Urkowitz, "Energy Detection of Unknown Deterministic Signals," *Proceedings of the IEEE*, vol. 55, Apr. 1967.

[15] J. Proakis, *Digital Communications*. McGraw-Hill, 4 ed., 2000.

[16] E. Like, V. Chakravarthy, P. Ratazzi, and Z. Wu, "Signal Classification in Fading Channels Using Cyclic Spectral Analysis," *EURASIP Journal on Wireless Communications and Networking*, 2009.

[17] H. Tang, "Some Physical Layer Issues of Wide-band Cognitive Radio Systems," *New Frontiers in Dynamic Spectrum Access Networks*, pp. 151 – 159, 2005.

[18] A. Ghasemi and E. Sousa, "Asymptotic Performance of Collaborative Spectrum Sensing under Correlated Log-Normal Shadowing," *IEEE Communications Letters*, vol. 11, Jan. 2007.

[19] D. Cabric and S. Mishra and R. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, pp. 772–776, 2004.

[20] A. Kattepur, A. H. Hoang, Y. Liang, and M. Er, "Data and Decision Fusion for Distributed Spectrum Sensing in Cognitive Radio Networks," *Proceedings of the 6th International Conference on Information, Communications & Signal Processing*, 2007.

[21] L. Berry, "Spectrum Metrics and Spectrum Efficiency: Proposed Definitions," *IEEE Transactions on Electromagnetic Compatibility*, vol. 9, Aug. 1977.

[22] M. Wellens, J. Riihijärvi, and P. Mähönen, "Evaluation of Spectrum Occupancy using Approximate and Multiscale Entropy Metrics," *Proceedings of the 5th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*, pp. 1 – 8, Jun. 2008.

[23] R. Matheson, "The electrospace model as a frequency management tool," *International Symposium on Advance Radio Technologies*, 2003.

[24] F. Sanders, B. Ramsey, and V. Lawrence, "Broadband Spectrum Survey at Los Angeles, California," *U.S. Department of Commerce*, May 1997.

[25] V. Valenta, R. Marsalek, G. Baudoin, M. Villegas, M. Suarez, and F. Robert, "Survey on Spectrum Utilization in Europe: Measurement, Analyses and Observations," *Proceedings 5th International ICST Conference on Cognitive Radio Oriented Wireless Networks and Communications*, Nov. 2010.

[26] D. Hatfield and P. Weiser, "Property rights in spectrum: Taking the next step," *1st IEEE Symposium New Frontiers Dynamic Spectrum Access Networks*, pp. 43 – 55, Nov. 2005.

[27] L. Xu, R. Tonjes, T. Paila, W. Hansmann, M. Frank, and M. Albrecht, "DRiVE-ing to the Internet: Dynamic radio for IP services in vehicular environments," *Proceedings of the 25th Annual IEEE Conference on Local Computer Networks*, pp. 281 – 289, Nov. 2000.

[28] M. Buddhikot, "Understanding Dynamic Spectrum Access: Models, Taxonomy and Challenges," *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, pp. 649 – 663, Apr. 2007.

[29] J. Mitola, "Cognitive radio for flexible mobile multimedia communications," *Proceedings of the IEEE International Workshop Mobile Multimedia Communications*, pp. 3 – 10, 1999.

[30] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting Mobile 3G Using WiFi," *Proceedings of the 8th International Conference On Mobile Systems, Applications And Services*, pp. 209 – 222, 2010.

[31] Federal Communications Commission, "Spectrum Policy Task Force Report," 2002.

[32] Federal Communications Commission, "Code of Federal Regulations, title 47 (telecommunication), volume 1 (FCC), part 15 (radio frequency devices), subpart 15, subpart c (intentional radiators), sec. 15.247 operation within the bands 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz," Oct. 2001.

[33] Federal Communications Commission, "Revision of Part 15 of the communications rules regarding ultra-wide band transmissions systems," Feb. 2002. ET Docket Number 98-153.

[34] Federal Communications Commission, "Facilitating opportunites for flexible, efficient, and reliable spectrum use employing cognitive radio technology," Mar. 2005. ET Docket Number 03-108.

[35] Federal Communications Commission, "In the Matter of Unlicensed Operation in the TV Broadcast Bands," Sept. 2010. ET Docket Number 04-186.

[36] A. Wyglinski, M. Nekovee, and Y. Hou, *Cognitive Radio and Communications Networks*. Elsevier, 1 ed., 2010.

[37] M. Gandetto and C. Regazzoni, "Cognitive radio communications and networks: principles and practice," *IEEE Journal On Selected Areas In Communication*, vol. 25, Apr. 2007.

[38] E. Visotsky, S. Kuffner, and R. Peterson, "On Collaborative Detection of TV Transmissions in Support of Dynamic Spectrum Sharing," *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*, pp. 338 – 345, Nov. 2005.

[39] M. Renzo, F. Graziosi, and F. Santucci, "Cooperative Spectrum Sensing in Cognitive Radio Networks over Correlated LogNormal Shadowing," *Proceedings of the 69th Annual IEEE Vehicular Technology Conference*, Apr. 2009.

[40] T. Weiss, J. Hillenbrand, and F. Jondral, "A diversity approach for the detection of idle spectral resources in spectrum pooling systems," *In the Proceedings of the 48th Int. Scientific Colloquium*, pp. 37 – 38, Sept. 2003.

[41] S. Mishra, A. Sahai, and R. Brodersen, "Cooperative Sensing among Cognitive Radios," *Proceedings of IEEE International Conference on Communications*, pp. 1658 – 1663, Jun. 2006.

[42] Peh and Y. Liang, "Optimization for Cooperative Sensing in Cognitive Radio Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference*, pp. 27 – 32, 2007.

[43] V. Poor, *Introduction to Signal Detection and Estimation*. Springer-Verlag, 1 ed., 1988.

[44] A. Tanenbaum, *Computer Networks*. Prentice Hall, 4 ed., 2003.

[45] J. Reed, *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall, 2002.

[46] J. Lim and A. Oppenheim, *Advanced Topics in Signal Processing*. Prentice-Hall, 1 ed., 1988.

[47] F. Digham, M. Alouini, and M. Simon, "On the Energy Detection of Unknown Signals Over Fading Channels," *IEEE Transactions on Communications*, vol. 55, pp. 21 – 24, Jan. 2007.

[48] W. A. Gardner and A. Napolitano and L. Paura, "Cyclostationarity: Half A Century of Research," *Elsevier*, Dec. 2004.

[49] E. Like, V. Chakravarthy, R. Husnay, and Z. Wu, "Modulation Recognition In Multipath Fading Channels Using Cyclic Spectral Analysis," *IEEE GlobeCom*, 2008.

[50] M. Ettus, "Ettus Research LLC." `http://www.ettus.com`. [Online; accessed Nov., 10, 2010].

[51] "SWIG Summary." `http://www.swig.org/exec.html`. [Online; accessed Nov., 10, 2010].

[52] Ettus Research, LLC, "Sales Page." `http://www.ettus.com/order`. [Online; accessed Dec., 30, 2010].

[53] GNU Radio, "USRP2 General FAQ." `http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQHow-are-the-USRP2-DDCs-implemented-How-many-CIC-stages-does-it-` [Online; accessed Dec., 29, 2010].

[54] Ettus Research, LLC, "USRP2 Schematic, rev. 4." `http://code.ettus.com/redmine/ettus/attachments/1/usrp2.pdf`. [Online; accessed Dec., 30, 2010].

[55] Ettus Research, LLC, "WBX Schematic." `http://code.ettus.com/redmine/ettus/attachments/51/wbx.pdf`. [Online; accessed Jan., 5 2011].

[56] Diamond Antenna, "D220 Wire Discone Antenna Website." `http://diamond-ant.co.jp/english/amateur/antenna/ante_6dis.html`. [Online; accessed Apr., 12 2011].

[57] Ettus Research, LLC, "WBX Schematic." `http://www.ettus.com/downloads/VERT900.pdf`. [Online; accessed Mar., 1 2011].

[58] The HDF Group, "HDF Homepage." `http://www.hdfgroup.org/HDF5/`. [Online; accessed Jan., 6 2011].

[59] , "H5Py Homepage." `http://code.google.com/p/h5py/`. [Online; accessed Jan., 6 2011].

[60] The HDF Group, "HDF5 Copyright Notice." `http://www.hdfgroup.org/HDF5/doc/Copyright.html`. [Online; accessed Jan., 6 2011].

[61] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Prentice-Hall, 3 ed., 2009.

[62] J. Bazerques and G. Giannakis, "Distributed Spectrum Sensing for Cognitive Radio Networks by Exploiting Sparsity," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1847–1862, Mar. 2010.

[63] S. Barbarossa and G. Scutari and T. Battisti, "Cooperative Sensing For Cognitive Radio Using Decentralized Projection Algorithms," *10th Workshop on Signal Processing Advances in Wireless Communications*, pp. 116 – 120, 2009.

[64] Ettus Research, LLC, "USRP2 Firmware and FPGA Image." `http://www.ettus.com/downloads/uhd_images/UHD-images-most-recent/UHD-images-003-20110317204706-2d906e6.tar.gz`. [Online; accessed Mar., 25 2011].

[65] Mini-Circuits, Inc., "ZFRSC-42 Datasheet." `www.minicircuits.com/pdfs/ZFRSC-42.pdf`. [Online; accessed Mar. 15., 2011].

[66] Microsoft Research, "White Space Finder." `http://whitespaces.msresearch.us/WSWebGUI/whitespaces.aspx`. [Online; accessed Mar. 1 2011].

[67] Advanced Television Systems Committee, Inc., *A/53: ATSC Digital Television Standard Parts 1 - 6*, Jan. 2007.

[68] Agilent, Inc., *N1996A CSA Spectrum Analyzer User's and Programmer's Reference*, March. 2007. `http://www.home.agilent.com/agilent/redirector.jspx?action=ref&cname=AGILENT_EDITORIAL&ckey=781056&lc=eng&cc=US&nfr=-536902453.927043`.

[69] L. Technology, "Ltc2284 datasheet." `http://cds.linear.com/docs/Datasheet/2284fa.pdf`. [Online; accessed Apr., 26, 2011].

[70] I. Agilent Technologies, "Specifications guide agilent csa spectrum analyzer." `http://www.home.agilent.com/agilent/product.jspx?cc=US&lc=eng&nid=-536902453.927043&pageMode=PL`, Sept. 2010. [Online; accessed Apr., 26, 2011].

[71] The Mathworks, "TriScatteredInterp Documentation." `http://www.mathworks.com/help/techdoc/ref/triscatteredinterp.html`. [Online; accessed Apr., 13, 2011].

[72] C. Haslett, *Essentials of Radio Wave Propagation*. Cambridge University Press, 2008.