Master Thesis
Electrical Engineering
Thesis no: MEE-27675
November 2010

Spectrum Sensing Through Implementation of USRP2

Adnan Aftab
Muhammad Nabeel Mufti

School of Computing
Blekinge Institute of Technology
371 79 Karlskrona
Sweden

This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Authors:
Adnan Aftab
E-mail: adnan.aftab@ymail.com
Muhammad Nabeel Mufti
E-mail: nabeel.mufti@gmail.com

University advisor:
Alexandru Popescu
School of Computing

School of Computing
Blekinge Institute of Technology
371 79 Karlskrona
Sweden

Internet : www.bth.se/com
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ACKNOWLEDGMENT

We really want to thank our supervisor Mr. Alexandru Popescu for his time, guidance, support and friendly attitude towards us all the time. His valuable advices paved the way for our research. We would like to extend our thanks to Dr. David Erman for providing us the experimental equipments and to Mr. Yong Yao, his interest towards our research and guiding us from precision towards accuracy.

We would also like to thank our families for their support, both financially and morally. Last but not least we like to say thanks to our friends in Sweden, they make our stay memorable and joyful.

Adnan Aftab and M. Nabeel Mufti

ABSTRACT

Scarcity of the wireless spectrum has led to the development of new techniques for better utilization of the wireless spectrum. Demand for high data rates and better voice quality is resulting in the development of new wireless standard making wireless spectrum limited than ever. In this era of wireless communication, service providers and telecom operators are faced with a dilemma where they need a large sum of the wireless spectrum to meet the ever increasing quality of service requirements of consumers. This has led to the development of spectrum sensing techniques to find the unused spectrum in the available frequency band.

The results presented in this thesis will help out in developing clear understanding of spectrum sensing techniques. Comparison of different spectrum sensing approaches. The experiments carried out using USRP2 and GNU radio will help the reader to understand the concept of underutilized frequency band and its importance in Cognitive Radios.

Keywords: Cognitive Radio, Spectrum sensing, GNU Radio, USRP2.

Contents

List of abbreviations

Acronym	**Description**
A	Ampere
ADC	Analogue to Digital Converter
CPC	Cognition enabling pilot channel
CPU	Central processing unit
DAC	Digital to analogue converter
dBm	Milliwatt-decibel
DC	Direct current
DDC	Digital down converter
DSSS	Direct sequence spread spectrum
DUC	Digital up converter
F	Frequency
FFT	Fast Fourier transforms
FPGA	Field programmable gate arrays
GHz	Giga hertz
GUI	Graphical user interface
HPSDR	High power software defined radio
IEEE	Institute of Electrical and Electronic Engineer
IF	Intermediate frequency
IP	Internet protocol
ISM	Industrial Scientific Medical
ITU-T	International telecommunication union
MAC	Media access control
MHZ	Mega hertz
MIMO	Multi input multi output
mm	Millimeter
MS	Mega samples
NCO	Numerically Controlled Oscillator
OSSIE	Open source SCA implementation embedded
PU	Primary user
PHY	Physical
RADAR	Radio detection and ranging
RF	Radio frequency
SCA	Software Communication Architecture
SDR	Software defined radio
SNR	Signal to noise ratio
SU	Secondary user
SWIG	Simplified wrapper and interface grabber

Acronym	**Description**
UBCR	Unlicensed based cognitive radio
UDP	User datagram protocol
UHD	Universal hardware devices
USB	Universal serial buss
USRP	Universal software radio peripheral
WiFi	Wireless Fidelity
WPAN	Wireless personal area network
WT	Wavelet transforms

List of Figures

List of Tables

CHAPTER 1

Introduction

1 INTRODUCTION

As the wireless communication has become the de facto standard for our growing and diverse demands, there's a need to use the spectrum as efficiently as possible to accommodate future innovations. In order to do that we need to analyze the spectrum carefully and deduce conclusions that will help us make the spectrum utilization process more efficient.

The electromagnetic radio spectrum can be considered as a natural resource. The use of radio spectrum by various different transmitters and receivers is governed by the different regulatory authorities and agencies.CR provides a unique solution to the spectrum utilization problem in terms of spectrum sensing techniques. Spectrum sensing has a twofold approach. Firstly available spectrum is sensed then it is assigned to the non-serviced users for efficient utilization. The underutilized frequency sub-bands are commonly known as "*spectrum holes*" or "*white spaces*". The spectrum hole can be defined as a band of frequencies not utilized by a serviced user at a particular time and specific geographical location. There are several different dimensions of spectrum sensing including frequency, time, space (geographical location), code and angle. Most of the spectrum sensing algorithms exploits any of the above mentioned dimensions to find the spectrum holes.

The thesis presents the implementation of spectrum sensing through energy detection and wavelet transformation algorithm using GNU Radio and Universal Software Radio Peripheral 2 (USRP2) by means of time and frequency dimensions. Furthermore comparison of all available spectrum sensing techniques is presented to identify the most efficient method. Spectrum sensing is considered as the prime element of any Cognitive radio (CR). By finding the underutilized frequency sub-bands or spectrum holes in the available spectrum, the radio spectrum scarceness issue can be resolved effectively.

1.1 Aims and objectives

The main aims and objectives of this project are:

1. To analyze different spectrum sensing techniques and find out which one is the most precise in terms of finding spectrum holes in available radio resource.
2. To develop a spectrum sensing algorithm and implement it over Universal Software Radio Peripheral 2(USRP2). Capture the raw data frames for ISM bands specifically 2.4 GHz and 5.8 GHz in the campus environment at different times according to the traffic intensity.

3. Extraction of raw data from ‘.bin’ and ‘.dat’ files and recompile it using graphic modeling tools.
4. To convert data to graphical form so that results can be analyzed and new decision making algorithm can be proposed later based on analysis of graphical results.

1.2 Research Questions

The main research questions for our thesis are as follows:

1. Which spectrum sensing technique is the most robust in terms of available radio resource or wireless spectrum?
2. Is it possible to sense the spectrum without having any prior information about it?
3. Which is the most suitable method to detect the spectrum holes in available radio spectrum using time and frequency dimension?
4. How to find the precise threshold level for sensed spectrum?
5. How to collect the received signals as close as possible to the USRP2 hardware with minimum overhead?
6. What is the most suitable graphical method to analyze the raw data collected through USRP2?

1.3 Thesis outline

The thesis report constitutes of 7 chapters.

- **Chapter 2** gives insight of technical background and related work done in the area of spectrum sensing.

- **Chapter 3** presents the methodology used to gather and present the results.

- **Chapter 4** provides familiarity with the software tools and hardware used in the research.

- **Chapter 5** presents the complete experimental setup.

- **Chapter 6** gives results and the synopsis of results.

- **Chapter 7** Conclusion and future work related to the topic under consideration.

CHAPTER 2

Spectrum Sensing

2 SPECTRUM SENSING

Cognitive Radio (CR) is a model for radio communication in which a wireless system alters its transmission or reception to effectively communicate with end user avoiding interference from other users present in the spectrum [10]. CR continuously learns about the radio spectrum by sensing the spectrum and changes its transmission or reception parameters according to the user behavior. The spectrum sensing principals of cognitive radio is shown in Figure 2.1.

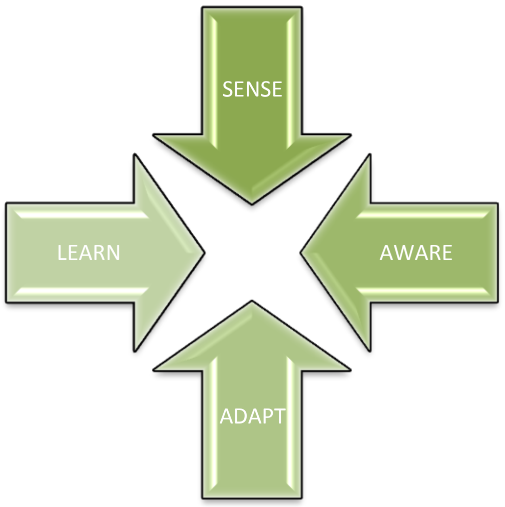


Figure 2.1 Cognitive Radio spectrum sensing

CR finds the free spectrum holes in the available spectrum range through sensing and learning. CR adapts to the changes in available radio spectrum and varies transmit or receive parameters according to the network condition. In the CR paradigm, there are two types of users known as Primary Users (PU) and Secondary Users (SU). Primary users are the licensed spectrum users who have direct access to the network whereas SUs are the users who rely on the CR decision for spectrum access [18]. There are two main types of spectrum sensing CRs

1. Licensed Band Cognitive Radios (LBCR): In which CR is capable of using licensed frequency bands assigned to users [10].
2. Unlicensed Band Cognitive Radios (UBCR): Which can only make use of the unlicensed part of Radio Frequency (RF) spectrum [10].

The rest of the discussion in this thesis focuses only on the second category of Cognitive Radios referring to it as CR.

2.1 Cognitive Radio Operation

CR emerged as an answer to spectrum crowding problem. Any CR's operation comprises of four states as shown in the Figure 2.2. First the available spectrum is sensed and analyzed to find any available spectrum holes. On the basis of spectrum analysis a decision is made to opportunistically assign the available frequency to the secondary user. Spectrum sensing is the most integral part of CR because all the remaining operations of CR rely on precise sensing of available spectrum [18].

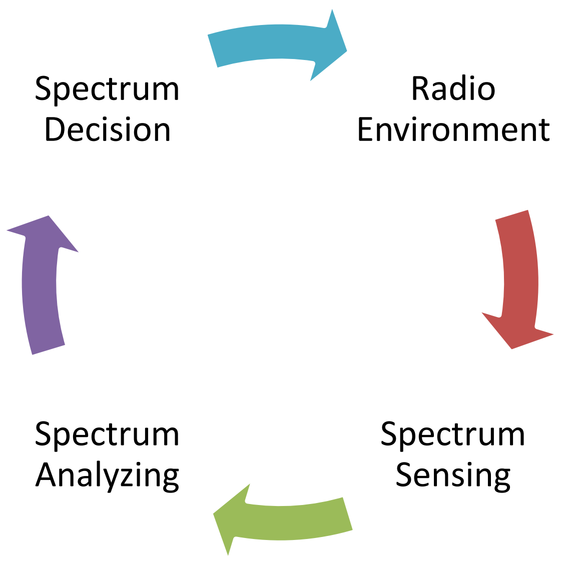


Figure 2.2 Cognitive Radio operation

2.2 Types of Spectrum Sensing

The most important task of spectrum sensing is transmitter detection. Spectrum sensing plays a key role in the decision making part of CR. There are several different ways to sense the spectrum. Some of the key methods used for spectrum sensing are as follows:

1. Energy Detection
2. Cyclostationary Method
3. Matched Filter detection
4. Wavelet detection

Explanation and comparison of all four methods is given below.

2.2.1 Energy Detection

In energy detection method we measure the energy of available radio resource and compare it against a predefined threshold level. If the measured energy falls below the defined threshold level spectrum is marked as available. When the measure energy level is above the defined threshold, it's considered as occupied. Energy detection method does not require any prior information of the signal. In simple words it does not care about the type of modulation used for transmission of signal, phase or any other parameter of signal. It simply tells if the radio resource is available at any given time instant or not without considering the PU and SU [10]. Hypothetically, energy detection can be considered as a method based on binary decision, which can be written as follows:

$$\boldsymbol{x(t) = n(t)} \qquad \boldsymbol{H0} \qquad (1)$$
$$\boldsymbol{x(t) = s(t) + n(t)} \quad \boldsymbol{H1} \qquad (2)$$

Where s(t) is the received signal and n(t) is the Additive White Gaussian Noise (AWGN) i.e. equally distributed all over the signal. H0 and H1 represent the two outcomes of the energy detection method [4]. The energy detection method's working principal can be explained with the Figure 2.3

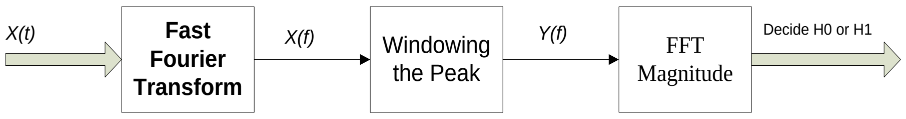


Figure 2.3: Energy Detection method

2.2.2 Cyclostationary Method

A Cyclostationary process is defined as the statistical process which repeats itself cyclically or periodically [6]. Communication signals are Cyclostationary with multiple periodicities. Mathematically Cyclostationary detection can be performed as given in equation (3):

$$\boldsymbol{Rx(T) = E\left[x(t+T)x^{*}(t-T)\right.}\mathrm{e}^{-\mathrm{j}2\alpha\pi\mathrm{t}}\boldsymbol{\big]} \qquad (3)$$

The equation shows the autocorrelation of the observed signal x(t) with periodicity T, E represents the expectation of the outcome and α represents the cyclic frequency [6]. After autocorrelation Discrete Fourier Transform over resulting correlation is performed to get the desired result in terms of frequency components. The peaks in the acquired data give us the information about the spectrum occupancy. The Cyclostationary detection method requires prior knowledge of periodicity of signal and it can only be used with the signal possessing Cyclostationary properties. The implementation of Cyclostationary method is shown in Figure 2.4.

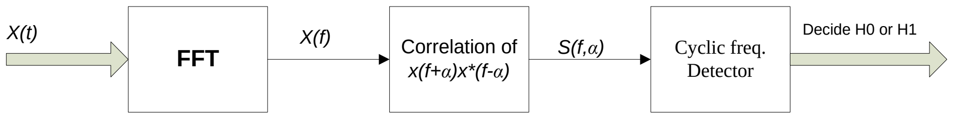


Figure 2.4: Cyclostationary Detection [10]

2.2.3 Matched filter detection

In the matched filter detection method a known signal is correlated with an unknown signal captured from the available radio resource to detect the presence of pattern in the unknown signal [6]. Matched filter detection method is commonly used in Radio Detection and Ranging (RADAR) communication [10]. The use of matched filter detection is very limited as it requires the prior information about the unknown signal. For example in case of GSM, the information about the preamble is required to detect the spectrum through matched filter detection method. In case of WiMAX signal prior information about the Pseudo Noise (PN) sequence is required for detection of spectrum.

2.2.4 Wavelet Detection

To detect the wideband signals, wavelet detection method offers advantage over the rest of the methods in terms of both simplicity and flexibility. It can be used for dynamic spectrum access. To identify the white spaces or spectrum holes in the available radio resource, the entire spectrum is treated as the sequence of frequency sub-bands. Each sub-band of frequency has smooth power characteristics within the sub-band but changes abruptly on the edge of next sub-band. By using the wavelet detection method the spectrum holes can be found at a given instance of time by finding the singularities in the attained result [10]. The Figure 2.5 shows the wavelet detection implementation for spectrum sensing.

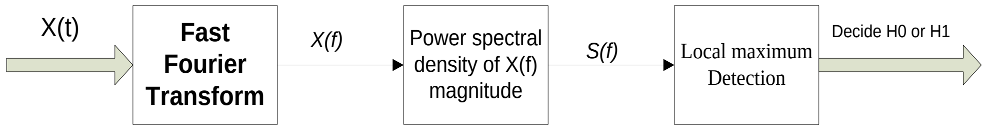


Figure 2.5: Wavelet Detection method

2.3 Qualitative analysis of spectrum sensing techniques

All of the above mentioned spectrum sensing techniques have certain advantages and disadvantages. Some of the techniques are suitable for sensing of licensed spectrum whereas others are suitable for unlicensed spectrum. The qualitative analysis of above mentioned spectrum sensing techniques are presented in Table1.

Table1. Spectrum sensing techniques comparison

Sensing technique	Advantages	Disadvantages
Energy Detection	Does not require prior information, Efficient, less complex	Limited functionality in low SNR areas, cannot differentiate between PU and SU
Cyclostationary	Works perfectly in low SNR areas, robust against interference	Requires fractional information about the PU, less efficient in terms of computation cost
Matched Filter	Low computation cost, accurate detection	Requires prior information about PU
Wavelet Detection	Works efficiently for wideband signal detection	Does not work for DSSS, more complex

It can clearly be seen from table1 that energy detection is the most suitable technique for unlicencessed spectrum bands as it does not require any prior information about the PU and SU.

2.4 Radio spectrum overview

Radio spectrum comprises of electromagnetic frequencies ranging lower than 30 GHz or having wavelength larger than 1milimeter (mm). Various parts of the radio spectrum are allocated for different kinds of communication application varying from microphones to satellite communication. Today, in most of the countries radio spectrum is government regulated i.e. governments and some other governing bodies like International Telecommunication Union (ITU-T) assign the radio spectrum parts to communication services [1].
There are several different frequency bands defined inside the radio spectrum on the basis of wavelength *(λ)* and frequency (*f*). Generically radio spectrum can be classified into two categories as licensed spectrum and unlicensed spectrum. Licensed spectrum comprises of frequency bands governed by government regulated agencies. It is illegal to use licensed frequency spectrum without taking permission from the regulatory bodies. Unlicensed frequency bands can be used by anyone for any scientific or industrial research.

One of the known unlicensed frequency band is Industrial, Scientific and Medical (ISM) frequency band or spectrum. It comprises of several different frequency bands. The ISM band defined by ITU-Regulation is given in the Table2.

Table2: ITU-R allocation of ISM Frequency bands [10]

Frequency Range	Center Frequency	Availability
6.765-6.795 MHz	6.785MHz	
13.553–13.567 MHz	13.560 MHz	
26.957–27.283 MHz	27.120 MHz	
40.66–40.70 MHz	40.68 MHz	
433.05-434.79 MHz	433.92 MHz	
902-928 MHz	915 MHz	ITU Region2 only
2.400-2.500 GHz	2.450 GHz	
5.725-5.875 GHz	5.800 GHz	
24-24.25 GHz	24.125 GHz	
61-61.5 GHz	61.25	
122-123 GHz	122.5 GHz	
244-246 GHz	245 GHz	

Most commonly used ISM bands are 2.4 GHz and 5.8 GHz band. Though these ISM bands were reserved for the purpose of research but currently these bands are used for different wireless communication standards. Examples of these bands are Wireless Local Area Network defined by Institute of Electrical and Electronics Engineers (IEEE) as *802.11a/b/g/n*, Wireless Personal Area Network (WPAN) defined by IEEE as *802.15* and Cordless phones which operate in the range of 915 MHz, 2.4 GHz, and 5.8 GHz. The research carried out in this thesis is performed using the 2.4 GHz and 5.8 GHz ISM band commonly used for WLAN and defined by IEEE as 802.11. 802.11 standards have defined 13 channels in the range of 2412MHz to 2472MHz. Each channel is 5MHz apart from each other and all the adjacent channel overlap with each other as each channel is 22MHz wide as shown in Figure 2.6.

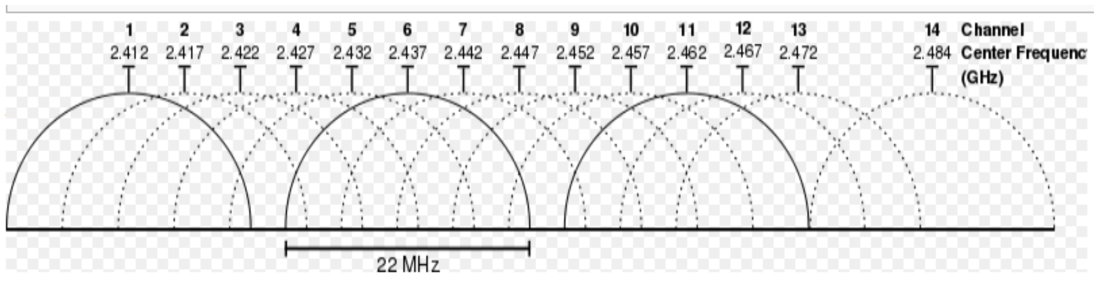

1 2 3 4 5 6 7 8 9 10 11 12 13 14 Channel
2.412 2.417 2.422 2.427 2.432 2.437 2.442 2.447 2.452 2.457 2.462 2.467 2.472 2.484 Center Frequenc (GHz)
22 MHz

Figure 2.6: Channel allocation in 2.4 GHz ISM band [23]

2.5 Related work

The recent focus on CR technology and advent of Software Defined Radio (SDR) such as GNU Radio has led to the implementation of GNU Radio in terms of the Cognitive Radio [3]. There are hundreds of citations available on IEEE explore investigating different aspects of spectrum sensing and CR. Most of the ongoing debate is concerned about finding the right spectrum sensing techniques for CR, channel allocation and transmission power handling for the Media Access Control (MAC) and Physical (PHY) layer implementation [5]. Underutilized bandwidth detection is key element of any spectrum sensing technique [9].

There are number of different platforms available for the implementation of CR in terms of SDR. One of these platforms include Open source Software Communication Architecture (SCA) implementation-Embedded (OSSIE) by Virginia Tech [24]. OSSIE is an open source software radio suite which can be used to model any of SDR and CR applications. Other platforms include High Power SDR (HPSDR) [25] and Flex Radio [26].

There are some other technique that could be considered as alternative to spectrum sensing. One of these techniques is cognition enabling pilot channel (CPC) [6]. According to CPC, a database of licensed users can be created which will monitor the use of spectrum by creating another channel and by advertising the spectrum opportunities in timely manner. But this will restult in additional infrastructure and use of another channel known as CPC. It is not the best approach to overcome spectrum scarcity as it will result in extra overhead in terms of radio resource.

CHAPTER 3

Research Methodology

3 RESEARCH METHODOLOGY

This chapter presents the research methodology used to carry out the project under consideration. This thesis covers all parts of a standard research methodology approach starting from the problem identification to soluction and from implementaion to validation of results. We have used qualitative, quantitative analysis and experimental methods to answer the research problem under consideration.

The work started with the literature review of CR. Firstly a thorough study of cognitive radio paradigm was conceded. After getting acquainted with the working principles of CRs, through journals and research papers, we narrow downed the scope to a more specific area of interest i.e. spectrum sensing. Spectrum sensing was chosen because of it's key role in making CR realizable. A further refined search was carried out on spectrum sensing using IEEE explore website to get the knowledge about the current research in the area of spectrum sensing.. Furthermore we conducted meeting with our supervisor Mr. Alexenderu Popescu, Dr. David Erman and Mr. Yang Yao to get further insight into the problem. On the foundation of knowledge base collected over a period of time we defined our research questions. we made deductive and statistical hypothesis to answer the research question. To strengthen our defined hypothesis and to motivate our answer, we first figured out if there are any white spaces in the congested 2.4 GHz ISM band.

The most essential part of any CR is to find the underutilized bandwidth in available radio spectrum in an efficeint manner. The underutilized bandwidth or spectrum holes should be found with the minimal information about spectrum as it is difficult to consider all the dimensions of radio spectrum while monitoring the spectrum for spectrum holes. Study about the all the available spectrum sensing techniques was carried out in terms of qualitative analysis. For experimental and quantitative analysis we used a testbench called USRP2 and GNU Radio which is an FPGA board with RF transciever interfaced with a Linux PC carrying GNU Radio software also known as Software defined Radio (SDR). Firstly we found out which technique of spectrum sensing is most robust for ISM bands.

After conducting a qualitative research, we found that energy detection method and wavelet detection method was most approperiate for the problem underconsideration. We implemented the algorithem over the testbench and gathered the results in forms of raw data constituting FFT values of spectrum at different instances of time. The data collected in real time was arranged in tables so that results could be monitored in graphical way. We conducted the experiments for both 2.4 GHz and 5.8 GHz ISM band. One of the major task was to present the data in a grpahical manner so that all three dimensions i.e., frequency, time, gain could be taken underconsideration at the same time. For this purpose, we developed spectrograms and 3D-plots to get a clear understanding of outcome and to validate the results in comparison to the hypothesis under consideration. The results and experiments are presented in the next section.

CHAPTER 4

GNU Radio and Universal Software Radio Peripheral 2 (USRP2)

4 GNU RADIO AND USRP2

4.1 GNU Radio Overview

GNU Radio is an open source development platform for signal processing and communication applications focusing on implementation of SDRs with low cost external RF hardware. It contains tons of libraries with signal processing routines written in C/C++ programming language. It is widely used in the wireless communication research and real time implementation of software radio systems [16].

GNU Radio applications are mainly written and developed by using Python programming language. Python provides a user friendly frontend environment to the developer to write routines in a rapid way. The performance critical signal processing routines are written in C++ [22]. Python is a high level language; it acts as a glue to integrate the routines written in C++ and executes through python. Python uses simplified wrapper and interface grabber (SWIG) for the purpose of interfacing C++ routines with python frontend application as shown in Figure 4.1. Very high speed integrated circuits hardware description language (VHDL) is a hardware descriptive language. This part of the code is executed in the Field Programmable Gate Array (FPGA) of front end hardware which is USRP2 in our scenario.

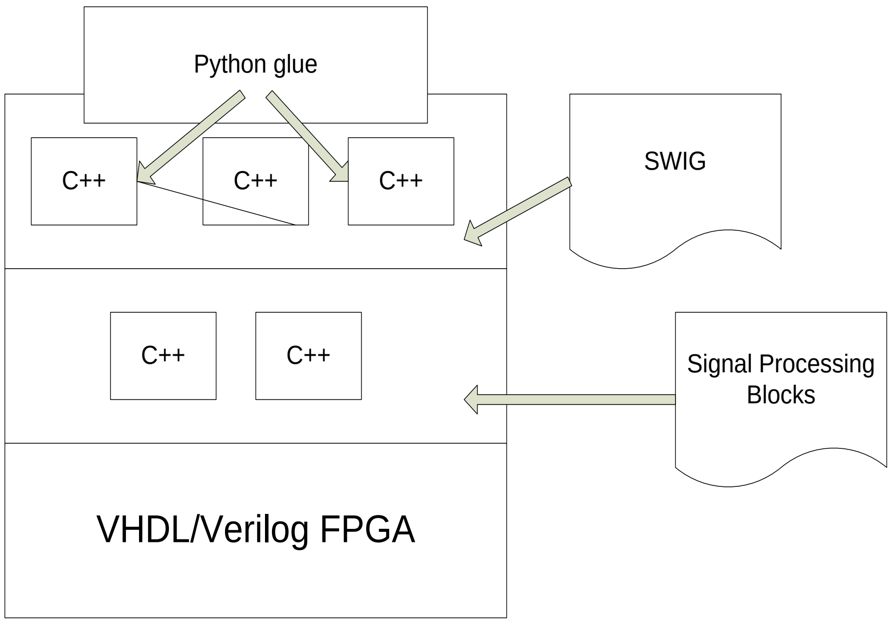
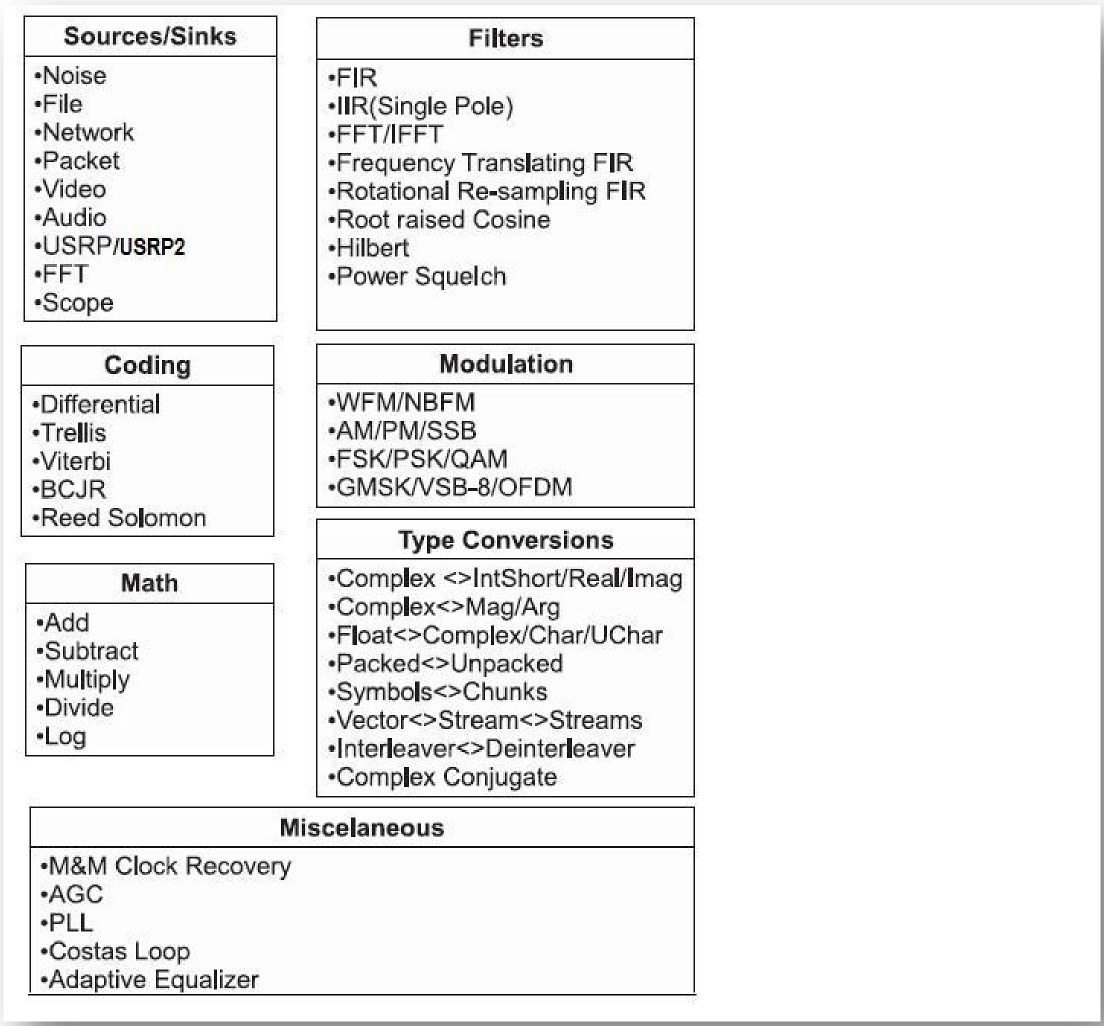


Figure 4.1: GNU Radio software architecture

GNU radio applications can be developed using both Object Oriented Approach and Procedural Approach depending upon the complexity of the problem under consideration. Some of the modules available in the current release of GNU Radio are shown in Figure 4.2:



Sources/Sinks
- Noise
- File
- Network
- Packet
- Video
- Audio
- USRP/USRP2
- FFT
- Scope

Filters
- FIR
- IIR(Single Pole)
- FFT/IFFT
- Frequency Translating FIR
- Rotational Re-sampling FIR
- Root raised Cosine
- Hilbert
- Power Squelch

Coding
- Differential
- Trellis
- Viterbi
- BCJR
- Reed Solomon

Modulation
- WFM/NBFM
- AM/PM/SSB
- FSK/PSK/QAM
- GMSK/VSB-8/OFDM

Math
- Add
- Subtract
- Multiply
- Divide
- Log

Type Conversions
- Complex <>IntShort/Real/Imag
- Complex<>Mag/Arg
- Float<>Complex/Char/UChar
- Packed<>Unpacked
- Symbols<>Chunks
- Vector<>Stream<>Streams
- Interleaver<>Deinterleaver
- Complex Conjugate

Miscelaneous
- M&M Clock Recovery
- AGC
- PLL
- Costas Loop
- Adaptive Equalizer

Figure 4.2: GNU Radio Modules [22]

4.1.1 GNU Radio Flow graphs, Sources and Sinks

Any GNU Radio application can be presented as a collection of flow graphs as in graph theory. The nodes of such flow graphs are called processing blocks. Processing blocks are the code routines written in C++. These processing blocks are tied together through flow graphs or lines connecting blocks. Data flows from one block to another through these flow graphs. All data types which are available in C++ can be used in GNU Radio applications e.g. real or complex integers, floats, etc [22]. Each block connecting one end of flow graph performs one signal processing operation for example encoding, decoding, hardware access etc. Every flow graph in GNU Radio requires at least one source or sink. Source and Sinks can be explained with the example of spectrum sensing scenario explained later in this chapter. In case of spectrum sensing our command line interface acts as sink whereas USRP2 acts as a source. When we use input command line parameters to tune USRP2 in this scenario, the interface acts as a source and USRP2 acts as a sink.

4.2 Typical Software Radio

A typical software radio consists of RF front and Analogue to Digital Converter (ADC) and Digital to Analogue Converter (DAC) interfaced with Central Processing Unit (CPU) and software. Receive and transmit path of typical software radio is shown in Figure 4.3:

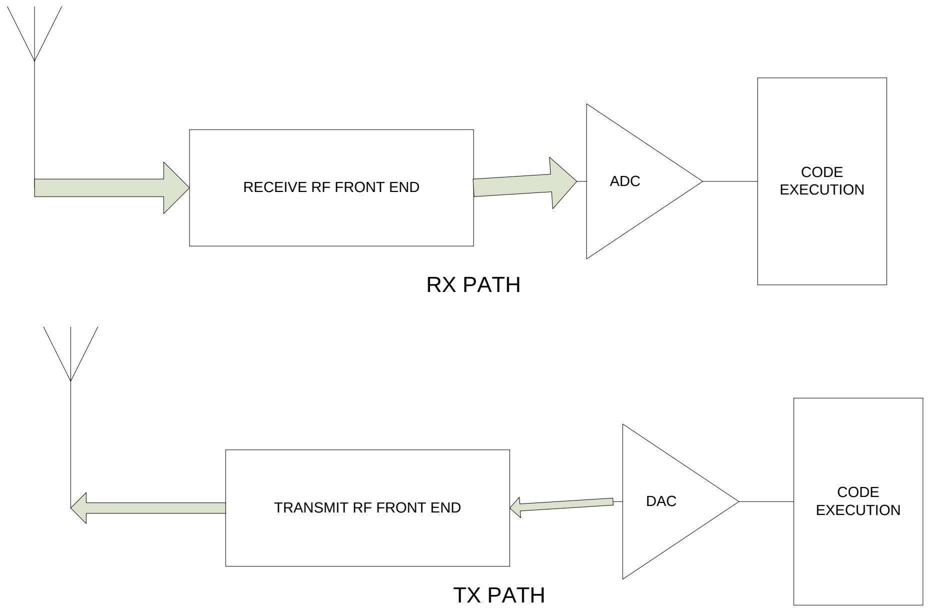


Figure 4.3: Basic Software Radio

4.3 USRP2 Architecture and Overview

USRP2 allows the creation of a software radio with any computer having gigabit Ethernet interface. It’s the upgraded version of its earlier release USRP. USRP has a USB interface limiting the data throughput from USRP to Computer at a maximum bandwidth of 8MHz. The design of USRP and USRP2 is open source and all schematics and component information can be downloaded from the website of the manufacturer [2]. USRP2 contains field programmable gate arrays (FPGA) and RF transceiver board which is connected over FPGA.

The main idea behind the design of USRP is to perform all the signal processing tasks for example modulation and demodulation, filtering at the host computer. All the general purpose tasks such as decimation, interpolation, digital up conversion and down conversion are performed inside the FPGA of USRP2. The Figure 4.4 shows the image of USRP2 with RF daughter board. USRP2 contains gigabit Ethernet controller, SD card slot and MIMO expansion slot at the front end with 8 LED indicators. SD card contains the driver for USRP2 mother board and RF transceiver. It requires 5V DC and 6A to power up USRP2 [2]. The main features of USRP2 are given in table 3.

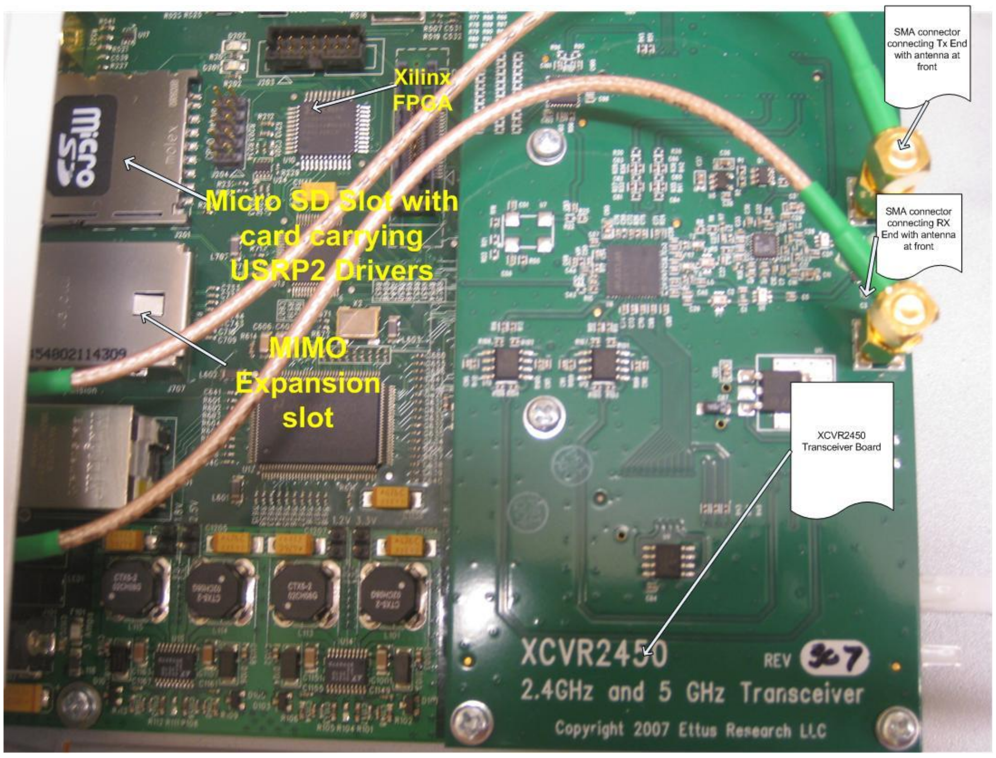

Xilinx FPGA
Micro SD Slot with card carrying USRP2 Drivers
MIMO Expansion slot
SMA connector connecting Tx End with antenna at front
SMA connector connecting RX End with antenna at front
XCVR2450 Transceiver Board
XCVR2450
REV
2.4GHz and 5 GHz Transceiver
Copyright 2007 Ettus Research LLC

Figure 4.4: USRP2 Motherboard and RF daughter card XCVR 2450

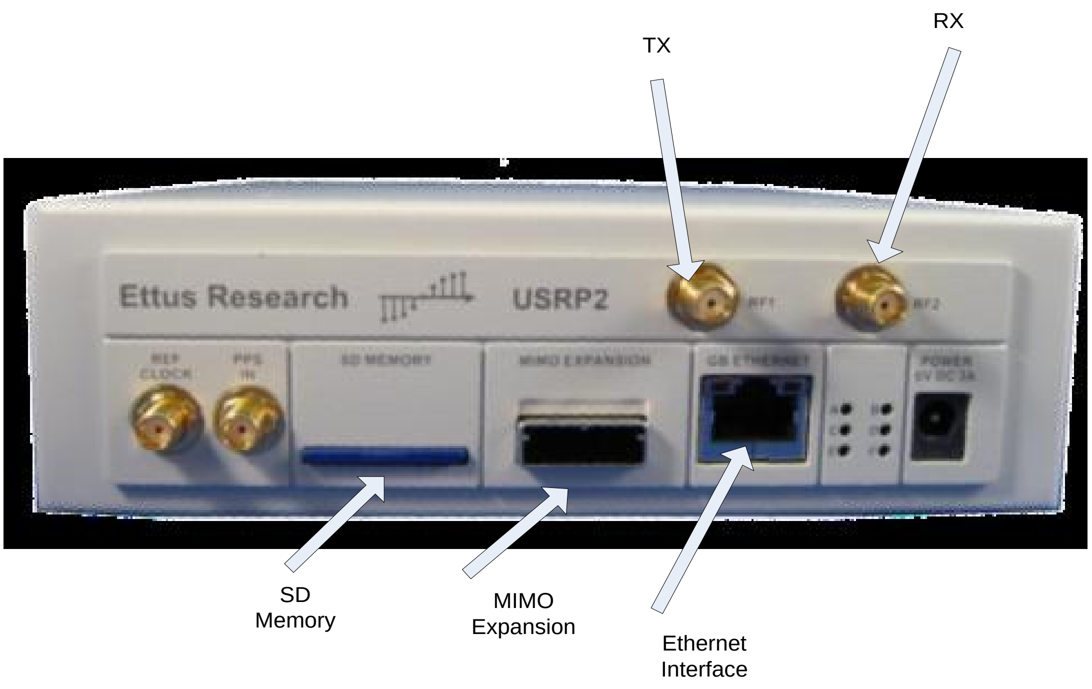


Figure 4.5: USRP2 Front end

Table 3: Main features of USRP2

Interface	**Gigabit Ethernet**
FPGA	Xilinx Spartan 3 2000
RF Bandwidth	25MHz
ADC	14 bits, 100MS/s
DAC	16 bits, 400 MS/s
Daughterboard slots	1 Tx, 1 Rx
SRAM	1 MB
Power	5V DC, 3A

4.3.1 USRP2 Operation with GNU Radio

USRP2 operation with GNU Radio can be explained with the help of Figure 3.6. RF transceiver fetches the RF signal from real time environment and converts it to Intermediate Frequency (IF) around direct current (DC). After converting it to IF the signal is passed to ADC. USRP2 contains two 14-bit ADC which provides sampling rate of 100MS/s [2]. The ADC after sampling passes the data to FPGA. The main task for the FPGA is the down conversion of remaining frequency and data rate conversion. After processing, FPGA transfers the results to gigabit Ethernet controller which passes it over to the host computer where the rest of the signal processing tasks are performed.
In case of transmission the same procedure is repeated in reverse order. Firstly gigabit Ethernet controller of host computer passes the input parameters to

USRP2. After receiving, the complex signal, digital up converter (DUC) converts the signal to IF before passing it to DAC. The DAC passes the IF converted signal to the RF transceiver where it is converted to RF signal and transmitted over the air.

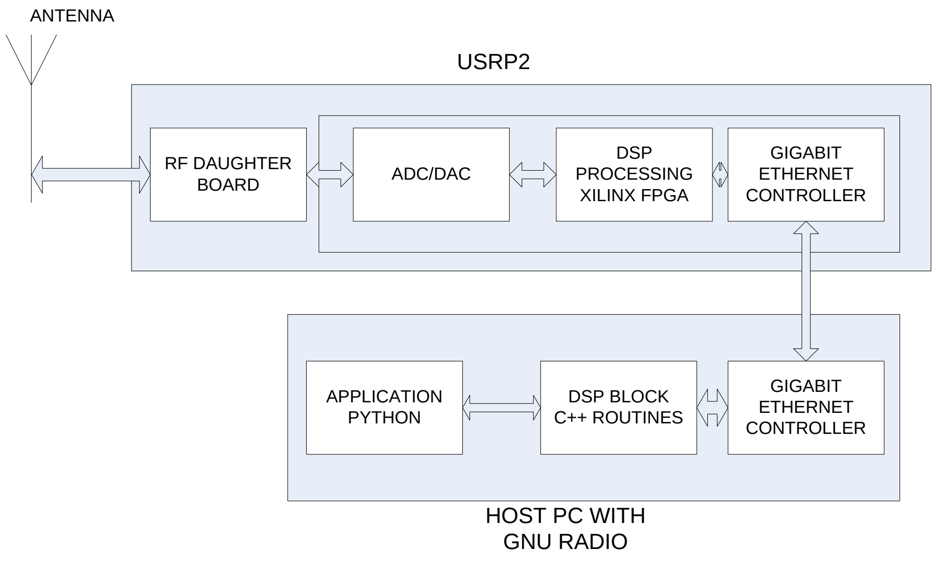


Figure 4.6: USRP2 operation with GNU Radio

In the FPGA down conversion of the signal from IF to baseband is performed. The data rate is further reduced so that it can be tailored according to the performance of transmission interface. The function of DDC in FPGA is as described in figure 4.7. The complex IF input signal is multiplied with the Numerically Controlled Oscillator (NCO). The signal resulting from the multiplication with the frequency of NCO is also complex and centered at DC. The process is similar to the one used in heterodyne receivers. The signal is then decimated with the factor N to further adjust the sampling rate. The phase generator in NCO is clocked at 125 MHz. The clock can be adjusted numerically or can be adjusted from the interface as well. The decimation factor is fed to the system through the gigabit Ethernet interface. The I and Q lines carry 16-bit signed integers through the gigabit Ethernet interface making each I and Q sample 4 bytes wide respectively. USRP2 is capable of processing signal up to 100MHz wide

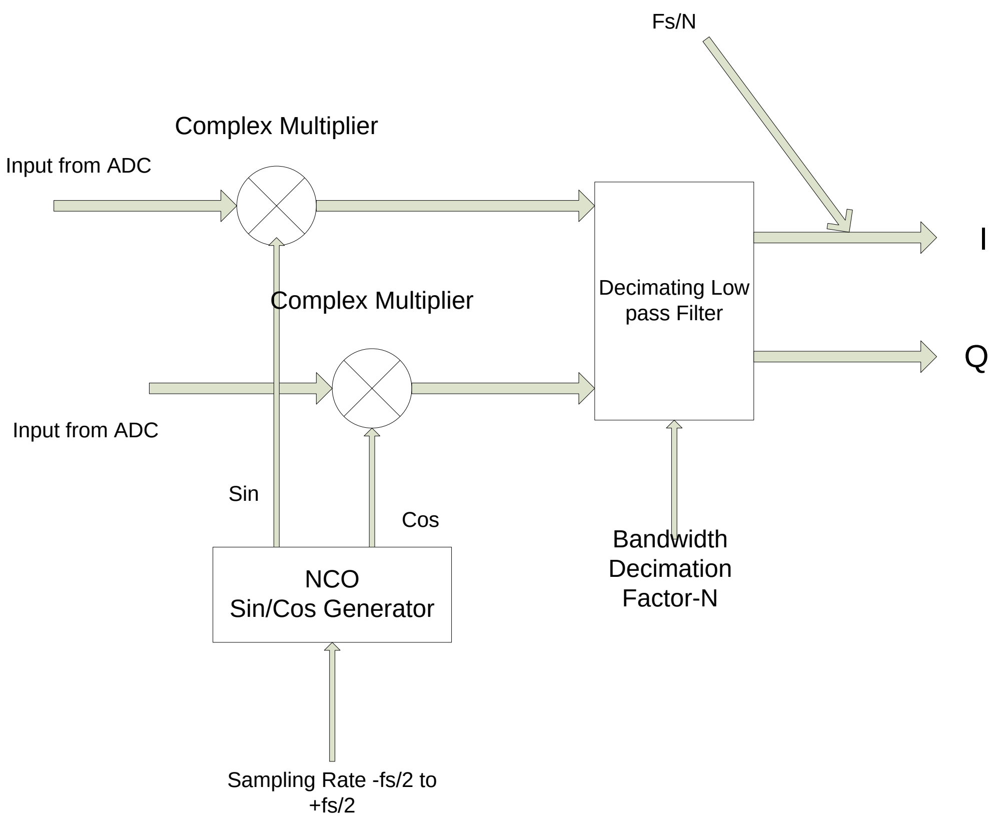


Figure 4.7: Operation of DDC in Xilinx FPGA

CHAPTER 5

Energy detection implementation using USRP2 and GNU Radio

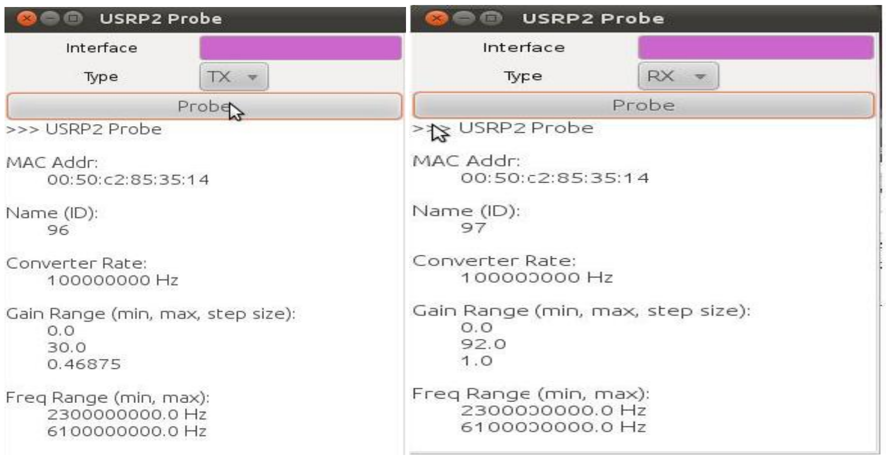
5 ENERGY DETECTION IMPLEMENTATION USING USRP2 AND GNU RADIO

5.1 Project Setup

The project setup was created by using GNU Radio installed on Ubuntu Linux 10.10 on personal computer (PC) with the specifications; core 2 Duo, 4 GB ram, 320 GB hard drive, Gigabyte Ethernet interface and ATI Radeon 512 MB graphics card. USRP2 device was used to physically receive the signal from real time environment. The USRP2 was connected to the Ethernet port of Host PC gigabit Ethernet card through CAT6 cable carrying RJ-45 jack at both ends. The USRP2 differs from its predecessor in a way that it requires certain configurations for boot up process unlike USRP which connects through USB port. It requires certain configuration at Linux terminal to make it work with GNU Radio. First the drivers were updated for USRP2 with XCVR2450. Universal Hardware Devices (UHD) provides complete support for the drivers of USRP2 product line. UHD provides both host drivers and Application programming Interface for standalone application development without GNU radio suite. USRP2 communicates at IP/UDP layer. The default IP address for USRP2 is 192.168.10.2, so to make it work Host PC should be assigned an IP address in the same subnet, for example 192.168.10.1. When USRP2 is not assigned an IP address, it communicates with the host pc using UDP broadcast packets, so it is essential to turn off the firewall before establishing the connection with USRP2 [2]. USRP2 can be found on the terminal by entering the following command provided by UHD.

```
$ sudo find_usrps
 00:50:c2:85:35:14 hw_rev = 0x0400
```

The command returns the MAC address of the USRP2 showing it is available and connected to the interface. GNU radio provides a python routine named USRP2_probe.py which acts as a software RF probe. It is a graphical user interface (GUI) application which returns the frequency range for the transceiver board and its gain in milliWatt-decibel (dBm). The output of the USRP2_probe.py for XCVR2450 dual band transceiver is given in Figure 5.1.



```
USRP2 Probe
Interface
Type        TX
Probe
>>> USRP2 Probe

MAC Addr:
      00:50:c2:85:35:14

Name (ID):
      96

Converter Rate:
      100000000 Hz

Gain Range (min, max, step size):
      0.0
      30.0
      0.46875

Freq Range (min, max):
      2300000000.0 Hz
      6100000000.0 Hz
```

```
USRP2 Probe
Interface
Type        RX
Probe
>>> USRP2 Probe

MAC Addr:
      00:50:c2:85:35:14

Name (ID):
      97

Converter Rate:
      100000000 Hz

Gain Range (min, max, step size):
      0.0
      92.0
      1.0

Freq Range (min, max):
      2300000000.0 Hz
      6100000000.0 Hz
```

Figure 5.1: USRP2_probe.py Output

After connecting the USRP2 with GNU radio and bringing it to up and running condition now we can execute any GNU Radio application by writing a routine in python or by using GNU Radio Companion (GRC). The experimental setup is shown in Figure 5.2 :

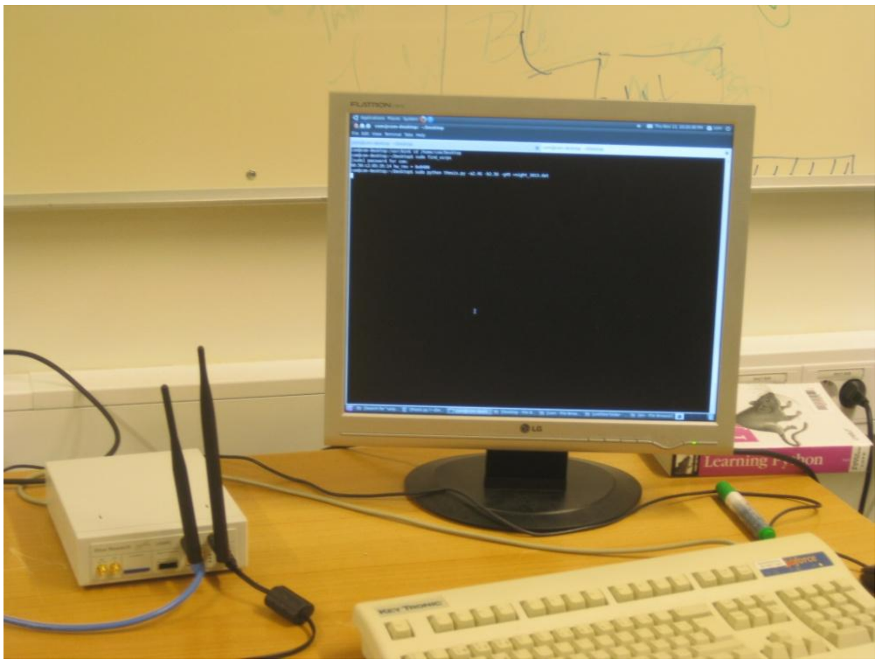
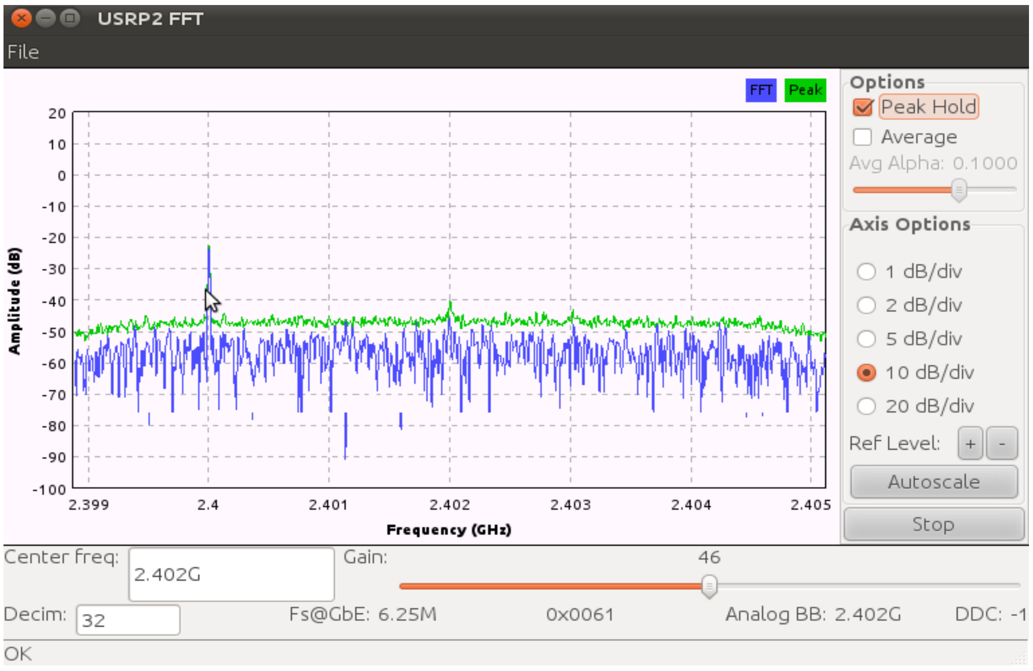


Figure 5.2: Experimental Setup

In this project we developed a routine called USRP2_spectrum.py which is based on usrp_spectrum_sense.py a standard routine available in GNU Radio latest release. The routine works as a software spectrum analyzer and is flexible enough to monitor any RF spectrum. The routine provided in GNU radio libraries was written for USRP first release and has certain limitation in terms of data rate and bandwidth due to USB interface of USRP. It can only monitor a maximum of 8MHz bandwidth at any

given time whereas USRP2 has a gigabit Ethernet making it flexible enough to monitor the 25MHz of bandwidth at any given time [2]. Hence we modified the routine to make it work with the USRP2.
There are some other routines (written in python) available in GNU Radio but none of these routines are flexible enough to monitor the whole spectrum for a long interval of time. One of these routines is USRP2_fft.py. This is a GUI application which takes the FFT of the received signal in real time and displays it over interface. The limitation with USRP2_fft.py is that, it only provides us with the gain at a given frequency without displaying the spectrum holes. Secondly it is a GUI application and it can only be executed using a low decimation rate, otherwise system specs should be very high. The output of the USRP2_fft.py is shown in Figure 5.3 showing the utilization of channel 1 of 2.4 GHz ISM band in BTH, Karlskrona Campus.



Peak Hold [✓]
Average [ ]
Avg Alpha: 0.1000

1 dB/div
2 dB/div
5 dB/div
[✓ 10 dB/div]
20 dB/div
Ref Level: + -

Center freq: 2.402G
Gain: 46
Decim: 32
Fs@GbE: 6.25M
0x0061
Analog BB: 2.402G
DDC: -1
OK

Figure 5.3: USRP2_fft.py output

Similarly, there is another application USRP2_rx_cfile.py. The application works as a broad band receiver, it fetches the signal from the external source environment and dumps the Digital down converter output in form of I and Q data directly into the appended file. The data collected at particular time t can be plotted using another GNU radio application named gr_plot_fft.py. The gr_plot_fft.py plots the raw data collection in terms of FFT as shown in Figure 5.4. USRP2_rx_cfile.py has a limitation in terms that it can only extract the data for a given center frequency as shown in Figure 5.4. The Figure show the utilization of RF spectrum at 2.437 GHz i.e. channel 6 of 2.4 GHz ISM band.

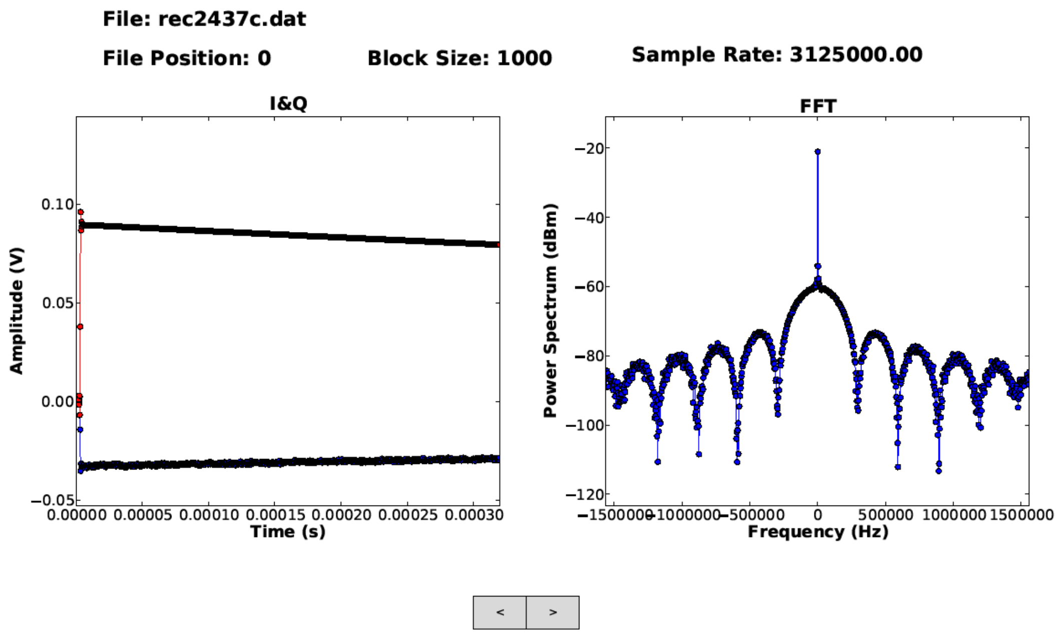


Figure 5.4: gr_plot_fft.py output with USRP2_rx_cfile.py data file

The modified spectrum sensing routine implemented by us overcomes the deficiencies which we have mentioned in available standard routines.

5.2 Spectrum sensing Algorithm implementation

Implementation of spectrum sensing algorithm can be explained with the help of flow graph presented in Figure 5.5. Flow chart shows the flow of data from source i.e., USRP2 to SINK which is Linux command line interface in our scenario. Source i.e. USRP2 fetches the signal of the desired frequency passed to it as a tuning parameter. After passing through USRP2 the raw data bit streams are converted to vectors or arrays of data. FFT is performed on the received raw data with the help of signal processing blocks and it is passed through the Blackman-Harris window to overcome the spectral leakage effect. When the FFT routine is implemented on a non periodic data, it results into spectral leakage i.e. the energy of the signal spreads out to a larger band of frequencies. Window functions helps out in reducing the effect of spectral leakage. After performing the FFT magnitude, decimation of the data is performed. Decimation is the inverse of interpolation. Decimation reduces the sample rate of data by performing down sampling to the desired rate and send to USPR2 as a tuning parameter. After performing decimation the obtained data is appended into a file or it can be plotted on the go through a GUI tool. The data collected is in the form of FFT magnitude bins. By plotting these magnitude bins against the given frequency range the frequency envelop of the signal can be monitored to find the white spaces or spectrum holes. By taking the Power Spectral Density of received FFT magnitude bins we can use the same algorithm for wavelet detection method.

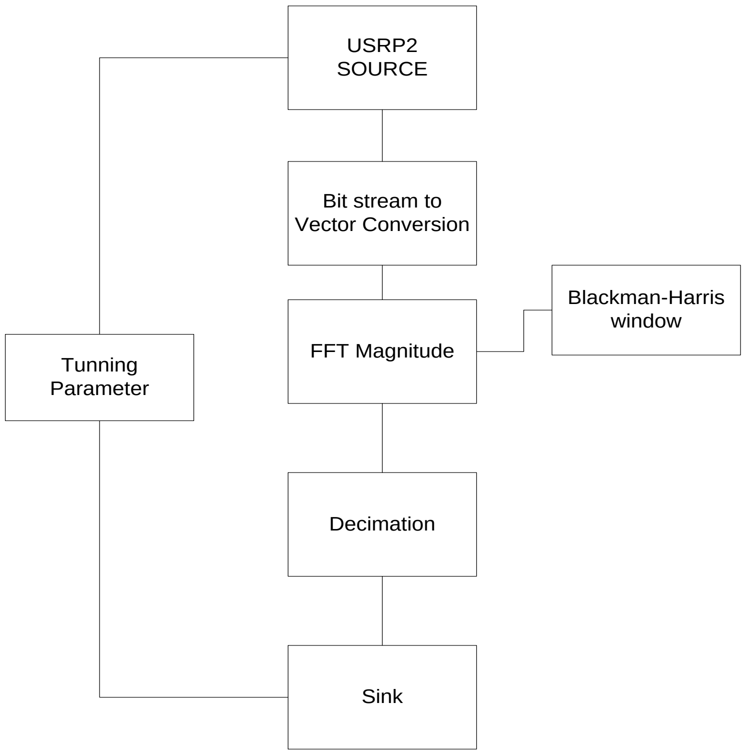


Figure 5.5: USRP2_spectrum.py flow chart

The spectrum sensing algorithm steps across the RF spectrum and takes the measurement in terms of FFT magnitude bins. The number of FFT bins, gain, decimation, time delay, dual delay and frequency range are forwarded to the USRP2_spectrum.py as tuning parameters.

The minimum number of bins is 3 and the maximum is 1024 bins. The FFT magnitude bins received at a certain frequency consist of 2 parts. First half of the bins i.e. FFT X[1] to FFT X[N/2 - 1] corresponds to the pass band spectrum from the center frequency (*fc*) to +Fs/2. Whereas the rest of the bins from X [N/2 - 1] to X [N-1] contains spectrum from center frequency to –Fs/2. Here X[1] represents the first bin value as shown in appendix B and X[N/2 - 1] represents the middle bin value. For example if we have 256 bins at 2402 MHz with 8MHz frequency step, the bin 0 to 127 corresponds to 2398 MHz to 2402 MHz and the rest of the bins correspond to 2402 MHz to 2406 MHz.

The gain parameter sets the gain of tuner card. By default the gain of the card is set to half of maximum gain which is 45dBm in case of XCVR2450. The time delay and dual delay parameters depends upon the decimation rate and length of FFT. By default decimation rate is set to 16 with 256 FFT bins and with time delay of 1ms. Time delay is a key parameter without setting the correct time delay the results could be altered or false. The reason for it is time delay displaying the amount of FFT frames from the source to sink should be forwarded in the defined time. Time delay for a given decimation rate and FFT size can be calculated as follows:

No. of FFT Frames = (Decimation Rate * Time delay) / FFT window Size **(4)**

In practice to reduce the non linear response of the DDC we performed FFT overlapping. Without FFT overlapping we were getting white spaces at the end of every sweep. We choose an overlap minimum of 25% i.e. we defined the step size to 8MHz for every step. In some cases we even choose overlapping of 0.75% to get the step size of 1MHz, so that we can get as accurate results as possible. The USRP2_spectrum.py is invoked in the following manner:

```
$ Sudo python USRP2_spectrum.py –a2.4G –b2.5G –g45 –d8 >rx.dat
```

The time delay parameter is set to 1ms in the code by default. –a represents the starting frequency and –b represents the last frequency, -g is used to set the gain parameter and-d8 represents that we are using decimation of 8.Decimation rate of 8 means that USRP2 processed(100MS/s divided by 8 i.e., Fs/N) 12.5MS/s during execution of code. The USRP2_spectrum.py is provided in appendix A at the end of the thesis. The typical output of running USRP2_spectrum.py will give us the FFT magnitude bins at a given center frequency. The gain at any given center frequency can be calculated by summing the values of all the bins and by taking square root of the result. After that by taking 20*log(x), where x is the square root of the sum of the bins, we can get the result in dBm.
The experiments were carried out in Room G403 of Building2 at Blekinge institute of technology (BTH) in Karlskrona, Sweden. Most of the results were collected during the same project room except a few instances where the results were collected in the cafeteria while turning on the microwave to check the performance of the algorithm in presence of high noise.

5.3 Project Limitations

Project limitations can be classified into two categories i.e.

- Hardware limitations
- Energy Detection Algorithm limitations

5.3.1 Hardware Limitations

The results were attained using decimation rate of 8.i.e 12.5MS/s. More precise results can be obtained by using lower decimation value. USRP2 can monitor maximum bandwidth of 25MHz. The receiver sensitivity can be increased by using a high gain antenna with the tuner card.

5.3.2 Energy detection Algorithm limitations

As previously mentioned energy detection based algorithms cannot differentiate between PU and SU [11]. The results are solely based on the threshold level received, or the energy level measured from the environment. Energy detection method cannot be used in a low noise setup.

The implemented routine can sense large bandwidth but not at the same time as it steps across the RF spectrum with the change in time domain.

CHAPTER 6

Results

6 RESULTS

The raw data collected by executing the USRP2_spectrum.py routine is appended into .dat and .bin files. These file can be loaded directly into MATLAB or any other plotting tools like octave, SigView or wavelet toolbox software. All the results in this chapter are plotted using Matlab2010 and Sigview. The sample raw data extracted in the '.dat' file is shown in appendix B. The results obtained show the usage of 2.4 GHz WiFi channels in the campus as well as free channels and spectrum holes. The results are collected using both 2.4 GHz ISM band and 5.8 GHz ISM band. The 5.8 GHz band is not used within the campus environment which can easily be observed by looking at frequency and magnitude plot. We have used three different types of plots to verify our results. These are frequency, magnitude and time, frequency, gain 3-dimensional (3D) plots and time, frequency spectrograms.

Figure 6.1 shows the results obtained for 2.4 GHz ISM band by using a frequency step of 20MHz i.e. the above defined routine steps across the ISM band in steps of 20MHz starting from 2.4 GHz and ending at 2.502G. As it can be observed from the graph, it is hard to find the exact channel utilization of 802.11 WiFi spectrum due to large sweeps across the spectrum. The Figure 6.1 shows first three cycles of algorithm i.e. algorithm steps across the spectrum in 5 steps in case of 20 MHz. Figure 6.2 shows the step plot with 20MHz step but it shows the results for a longer period of time which can be seen from the marking on the stems. The main purpose of continues and stem plot is to simply monitor the gain of the sensed signal at a particular frequency. These two dimensional plots simply tells us about the utilization of WLAN channels in BTH network. Figure 6.3 shows the same results in 3-dimensions (3-D) by using another axis for time. The advantage of time axis is that we can monitor the results at any given instance of time. The time dimension gives us flexibility to find the white spaces or spectrum holes in radio spectrum at any given instance of time *t*. Figure 6.3 shows the results in terms of Frequency, Gain (dBm), and time. The gain factor is calculate by dividing received samples by total number of bins and by taking log of FFT magnitude values as given in appendix C. The gain factor tells us that the signal strength has increased by a certain factor as compared to the original signal or other comparative signal samples depending upon the scenario.

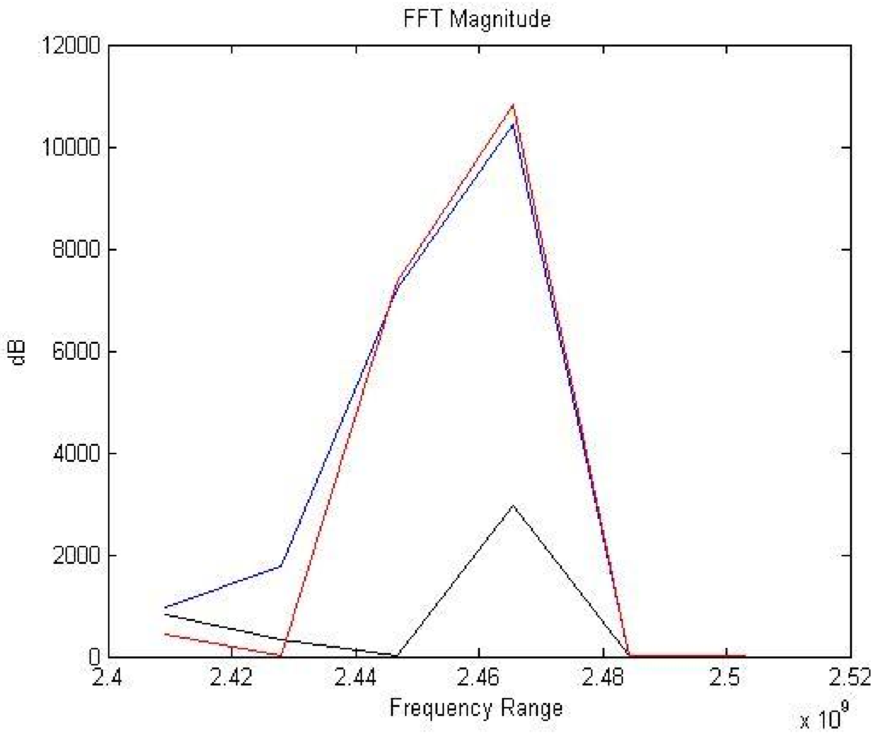


Figure 6.1: FFT magnitude plot for 2.4 GHz band with 20 MHz frequency Step

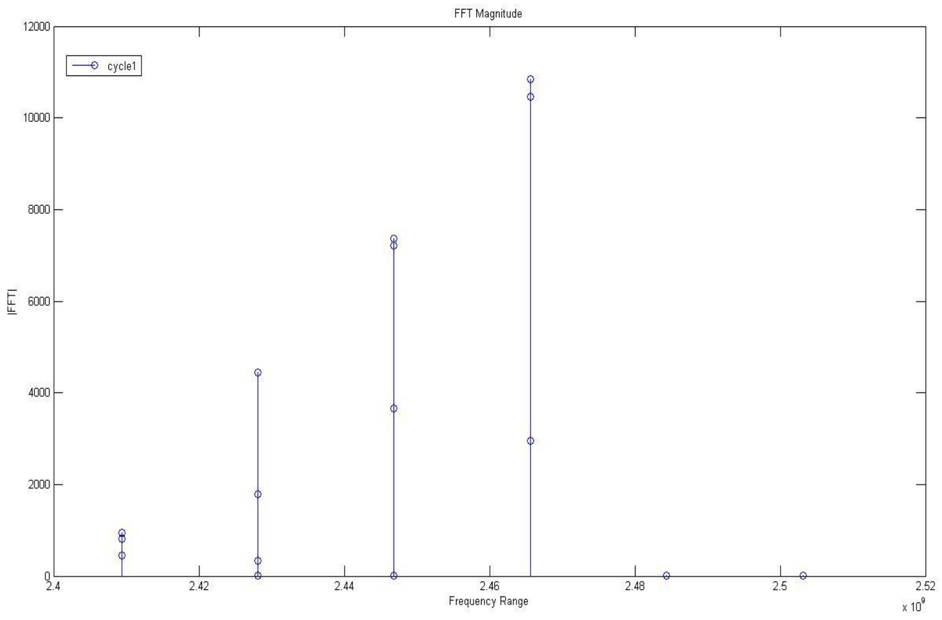


Figure 6.2: Frequency vs. FFT magnitude stem plot for 2.4-2.5 GHz ISM band

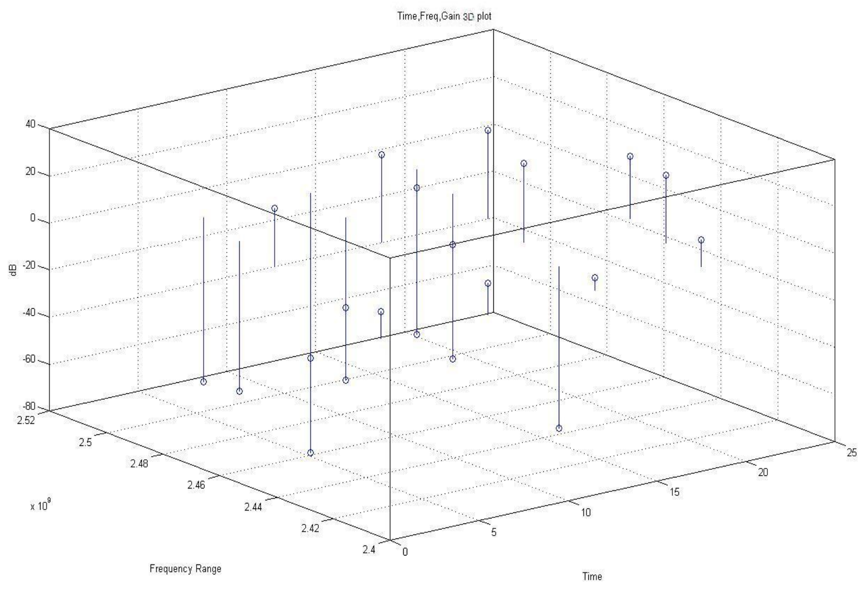


Figure 6.3: Time, Frequency, and Gain 3D plot for 2.4 GHz-2.5 GHz using 20 MHz frequency step

The results obtained using a smaller step of 10MHz is shown in Figure 6.4. We can easily find the channel utilization of campus WLAN by looking at Figure 6.4 and Figure 6.5. The spikes around 2.43x10^9 Hz in the plot show the use of Channel 7 at the time of data collection. Frequency and magnitude plots provide us gain at a given frequency but still it's not good enough to find the threshold level or to decide which part of spectrum is free because it does not has the time-axis. For this purpose we have used spectrograms.

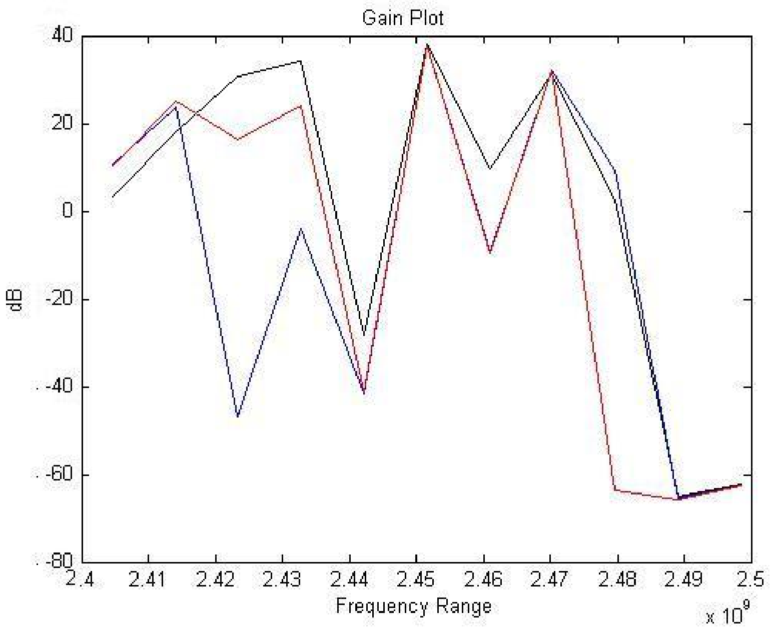


Figure 6.4: Frequency and Magnitude plot with 10 MHz step

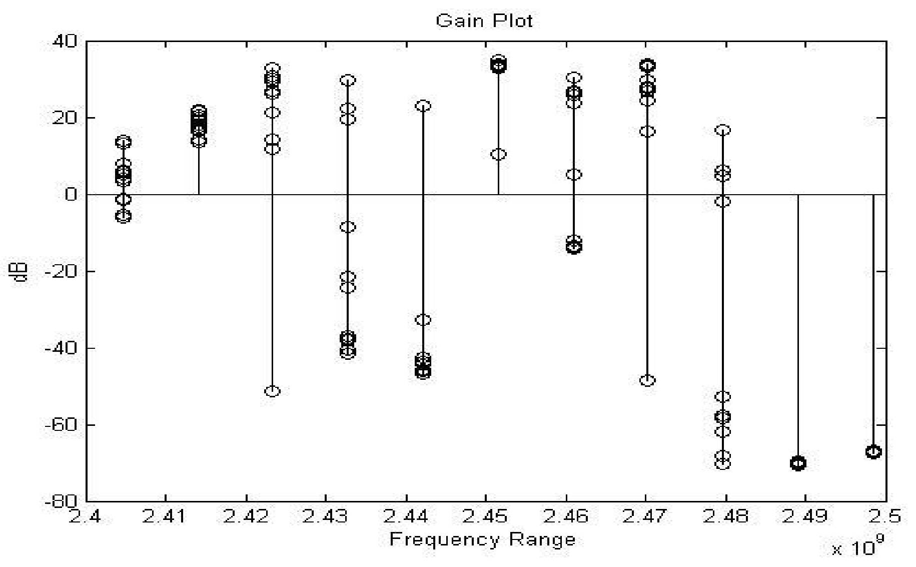


Figure 6.5: Frequency and Magnitude plot with 10 MHz step at 2.4 GHz ISM band

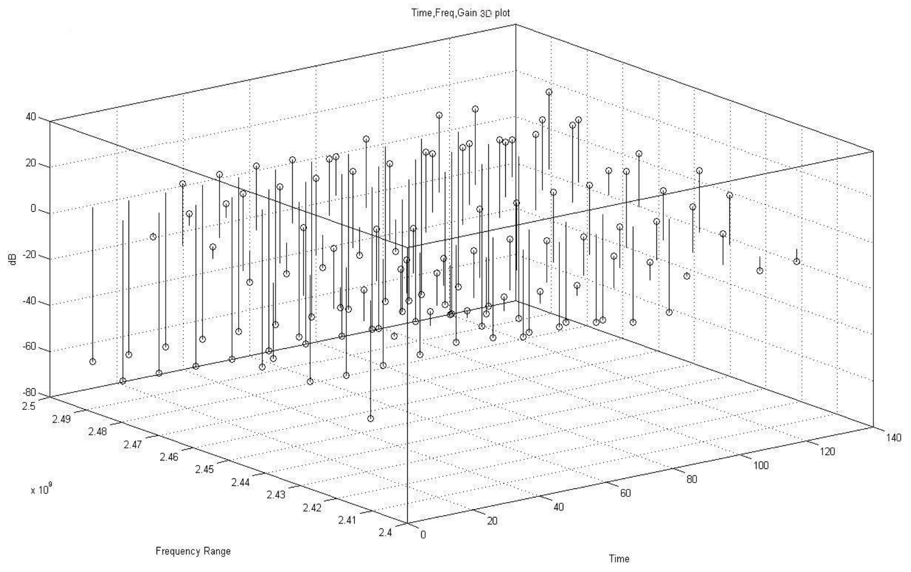


Figure 6.6: Time, Frequency, Magnitude plot using 10 MHz step

Spectrogram shown in Figure 6.7 displays the time frequency relationship with respect to gain. A spectrogram is a time varying spectral representation of a signal which shows how the spectral density of the signal varies with time. The color bar presented at the right side of the plot shows the different level of energy or gain values. To find the spectrum holes or availability at the defined threshold at any instant, we can compare the color with time and frequency axis. The color red in this

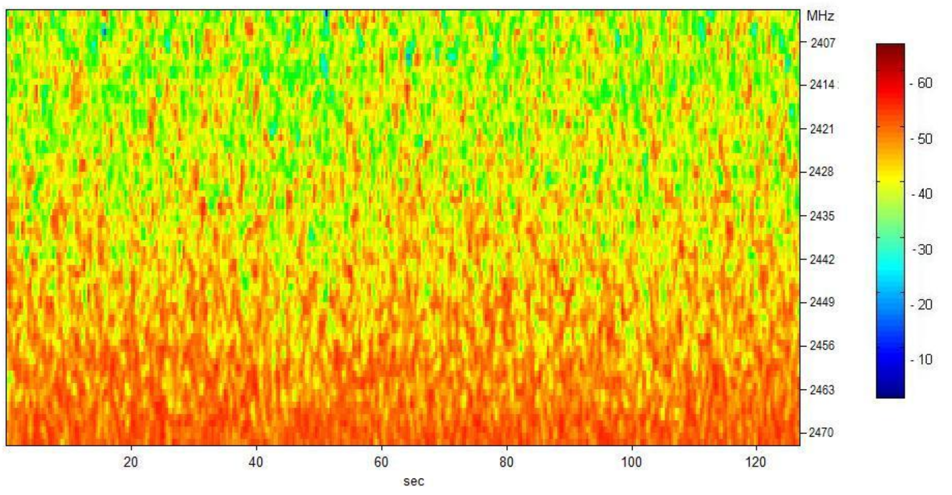


Figure 6.7: Spectrogram plot using 10 MHz step

Spectrogram shows the spectrum holes or underutilized bandwidth at a given instance of time and Frequency. Due to large frequency step it is hard to differentiate between the colors and it is hard to find the white spaces at the exact location.

To gather even more precise results we repeated the experiment with 5 MHz frequency step and sweep across the spectrum again. The results obtained are shown in Figure 6.8-6.10. Here we can easily observe the use of channel 3,7,11 in the campus WLAN environment by looking at Figure 6.8 and Figure 6.9. each stem value represents the results collected during separate cycle of data collection presented collectively in one stem plot. The utilization of channel 3,7,11 is expected as at these channel frequencies we have the maximum gain values at a given instance of time *t*.

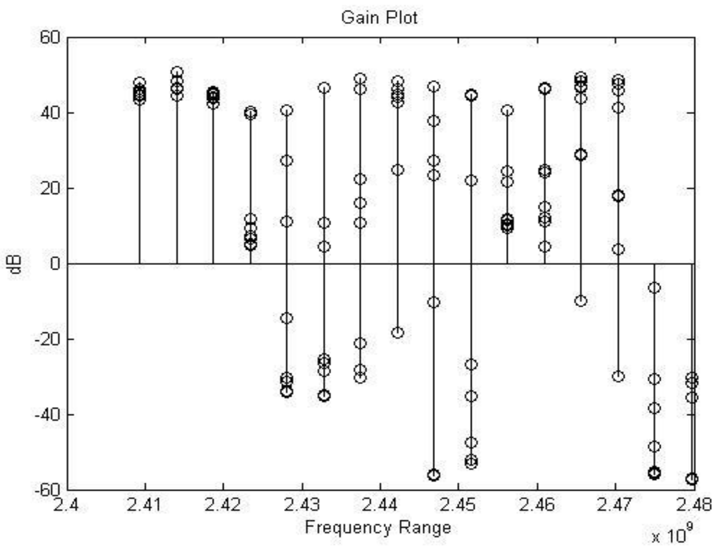


Figure 6.8: Frequency vs. magnitude plot with Frequency step of 5MHz

The spectrogram given in Figure 6.10 is more precise in terms of clarity showing spectrum holes in terms of red tiled surface in the presence of other colors representing different gain values of energy spectrum. the better results are attained by using a lower frequency resolution or step size which is 5 MHz in this case.

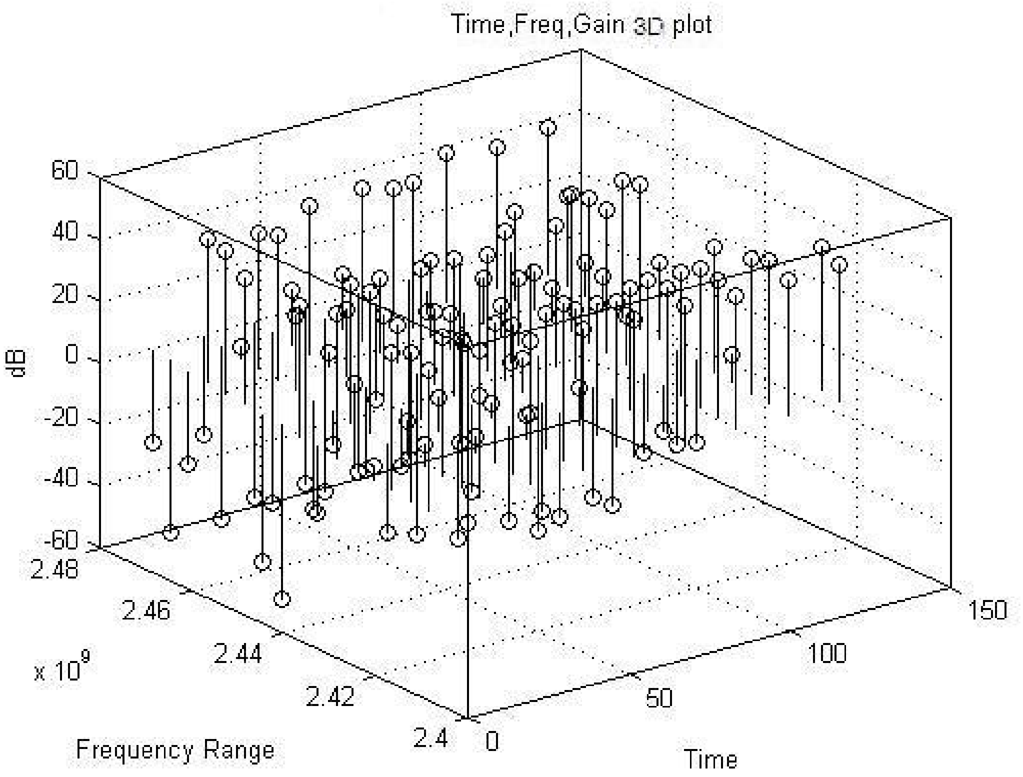


Figure 6.9: Time, Frequency, gain 3D plot with frequency step of 5MHz

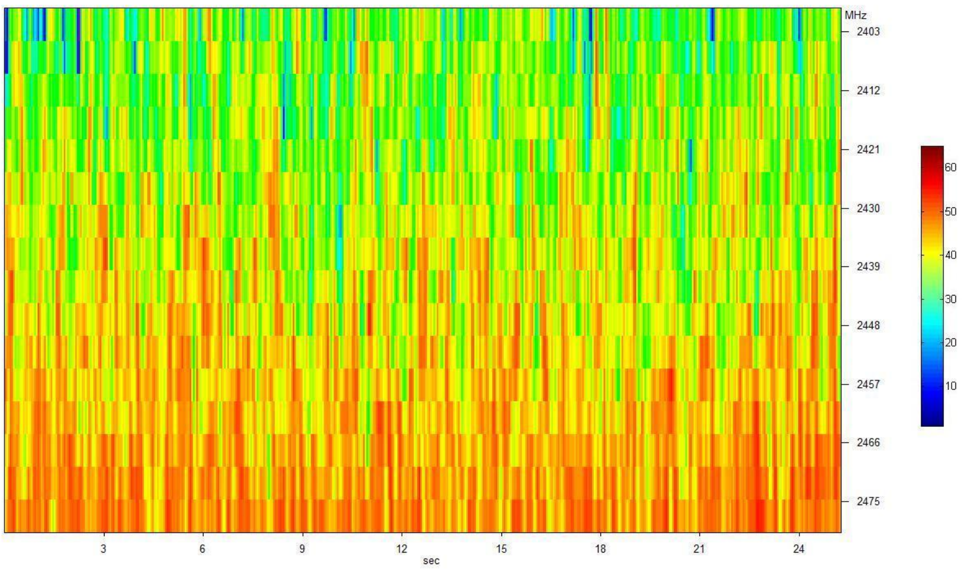


Figure 6.10: Spectrogram plot using 5MHz step

Similarly we gathered the results using 1MHz step. The results obtained are shown in Figure 6.11 and Figure 6.12 showing the utilization of channel 1, 7 and 11 in the WiFi network of the campus.

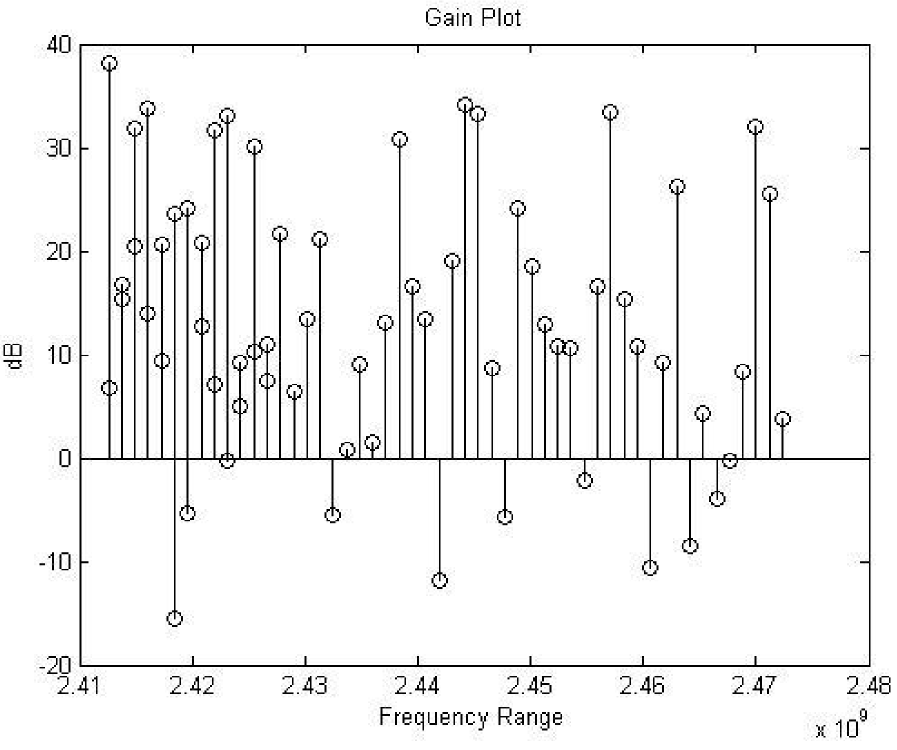


Figure 6.11: Frequency Vs. Gain plot using 1MHZ step

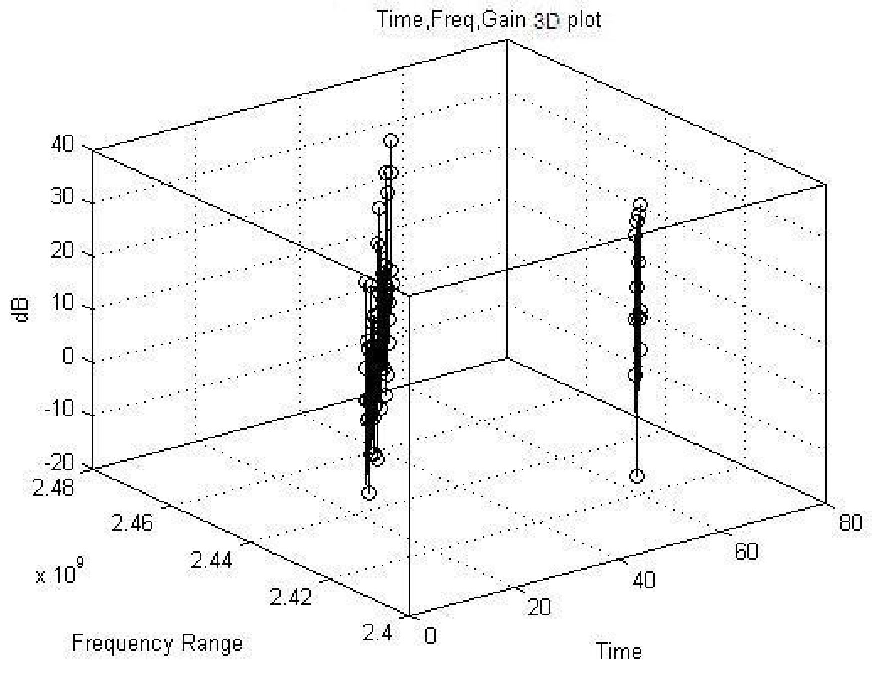


Figure 6.12: Time, Frequency, Gain 3D plot using 1MHz step

To check the limitations of our project we conducted another experiment by placing the USRP2 and host PC near microwave ovens to check if energy detection method is susceptible to the high signal strength environment or not. The results attained are shown in Figure 6.13, 6.14 and 6.15. The results clearly show increase in gain in less than 1 second. We received gain values as high as 50dBm and energy detection algorithm didn't identify any other WiFi channel though the university café is a hotspot and it has a number of wireless routers placed in the vicinity. The spectrogram in Figure 6.15 shows the use of only microwave oven frequency i.e. 2.45 GHz in different colors the rest of the frequency range is red proving microwave frequency magnitude to be high enough to suppress any other signal in the vicinity of cafeteria.

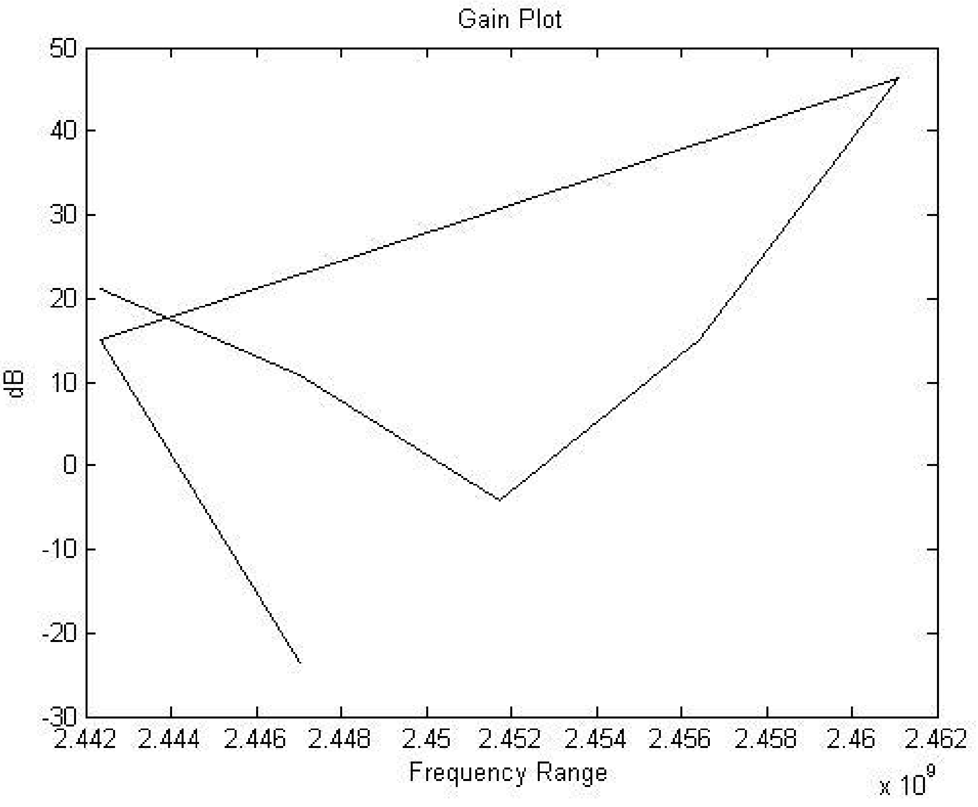


Figure 6.13: Frequency Vs. Gain plot at 2.45 GHz microwave oven range

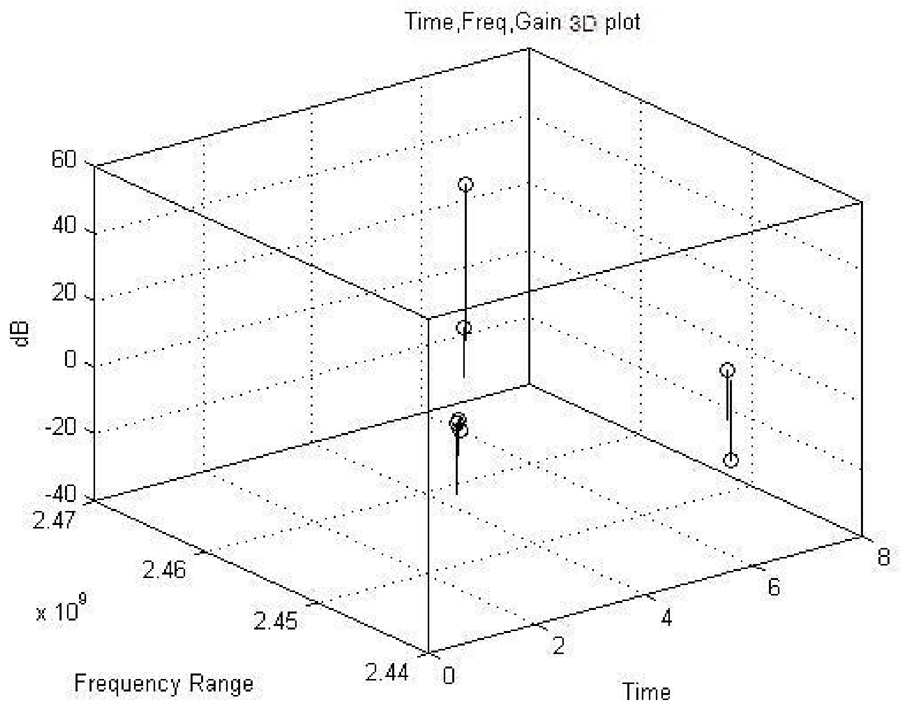


Figure 6.14: Time, Frequency, Gain 3D plot at microwave oven frequency

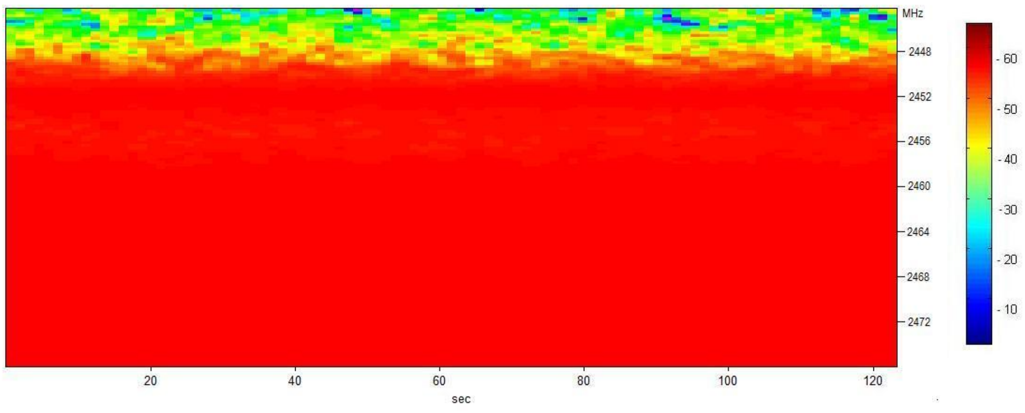


Figure 6.15: Spectrogram for microwave oven frequency

.

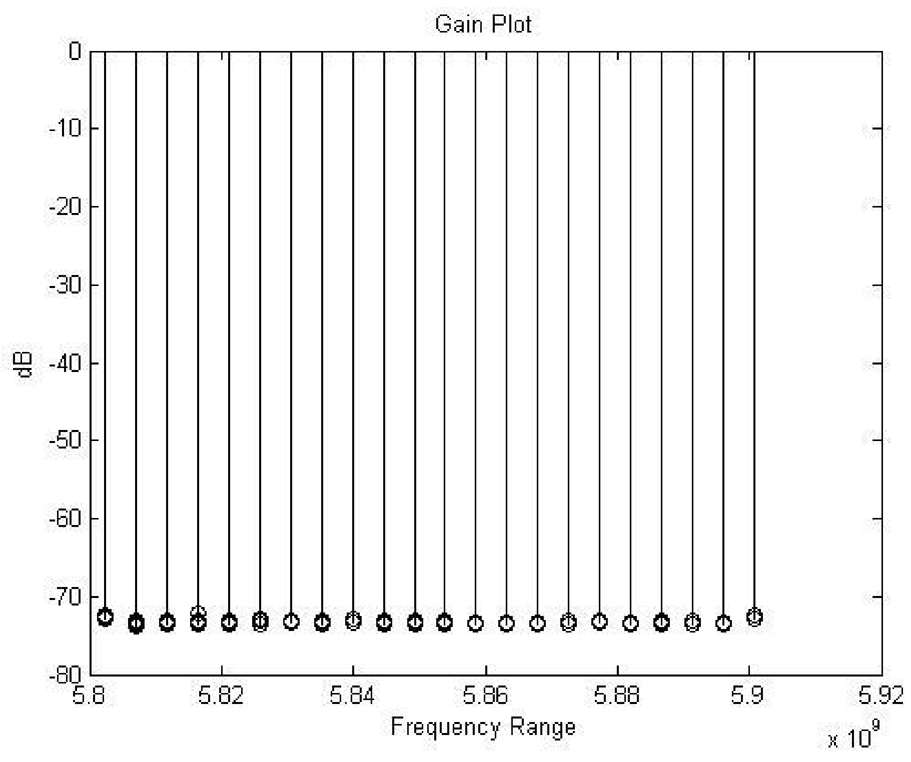


Figure 6.16: Frequency vs. gain plot for 5.8 GHz ISM band

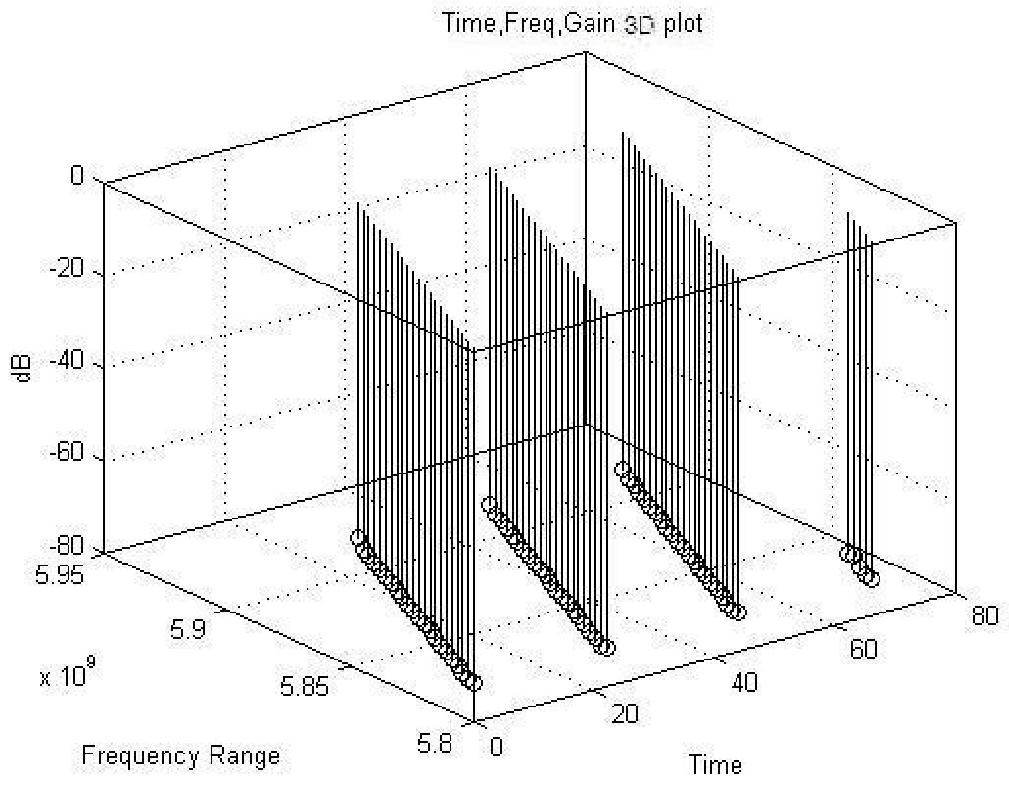


Figure 6.17: Frequency, Time, Gain plot for 5.8 GHz ISM Band

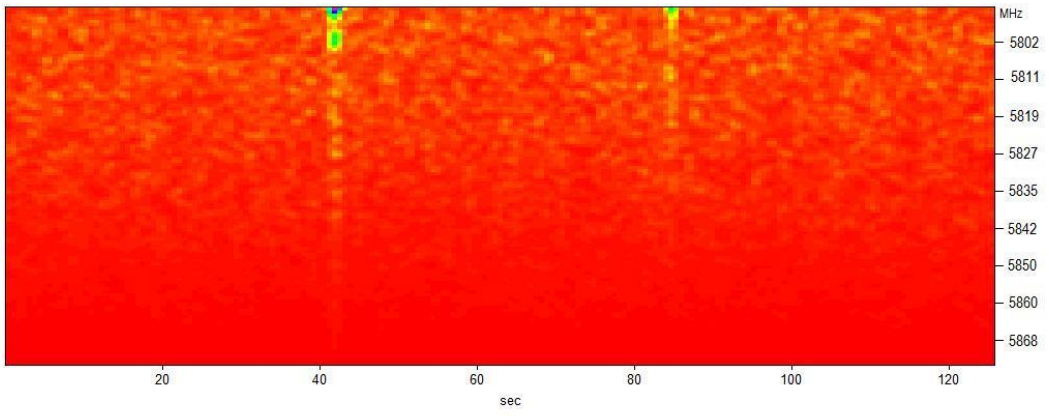


Figure 6.18: Spectrogram for 5.8 GHz ISM band

Final experiment was conducted over 5.8 GHz ISM band. Since 5.8 GHz frequency band is not used inside the campus, the whole spectrum range appears as white space or underutilized. It can be observed by looking at the Figure 6.17-6.18. A few colored spots in the spectrogram are due to the thermal noise or other hardware noise figure.

CHAPTER 7

Conclusion and Future Work

7 CONCLUSION AND FUTURE WORK

The report presents implementation of energy detection and wavelet based spectrum sensing implementation and analysis of different spectrum sensing techniques. Different experiments were performed to find out the spectrum holes in the occupied 2.4 GHz ISM band. The raw data collected by USRP2_spectrum.py is plotted using different plotting techniques to find the spectrum holes. The qualitative analysis of different spectrum sensing methods shows that energy detection is the most reliable and authentic method for the spectrum sensing as it has the lowest computational cost and requires no a prior information about the spectrum. The raw data collected in the form of FFT bins is the most efficient way of collecting data for spectrum sensing as it requires just a few signal processing operations. Rest of the work is performed inside the USRP2, This is closest one can get to the hardware for precise results. The results obtained proved that it is possible to find the underutilized bandwidth in a spectrum without having prior knowledge of PU and SU. Spectrograms though often used in astronomy and acoustics can be used to identify the spectrum holes as shown in the result by plotting spectrograms. The threshold level for any given spectrum of frequencies depends upon several factors such as receiver sensitivity and the number of energy transmitting/receiving nodes and the dimensions taken under consideration.

7.1 Future work

Cognitive radio is relatively new area of research as compared to the rest of communication theory. There are several other methods of spectrum sensing which need to be explored. Energy detection method results can be improved through the use of the Multiple Input Multiple Output (MIMO) approach. This can be done by connecting multiple USRP2 together and sensing the spectrum for a large area. Another way to sense the spectrum is by using cooperative communication and by taking antenna diversity and other factors under consideration. This study could be extended by repeating the same experiment for licensed frequency bands and the results obtained can be compared with ISM band results to get the better understanding of the area of research.

Bibliography

[1] J.Mitola. “Software radios-survey, critical evaluation and future directions”, *IEEE National Telesystems Conference*, pages 13/15- 13/23, 19-20 May 1992.

[2] Matt Ettus. Universal software radio peripheral. http://www.ettus.com.

[3] T. Yucek and H. Arslan, “A survey of spectrum sensing algorithms for cognitive radio applications,” *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, First Quarter 2009.

[4] M. Sarijari, A. Marwanto, “Energy detection sensing based on GNU radio and USRP, An analysis study”, *2009 IEEE 9th Malaysia International Conference on Communications (MICC)*, no.15-17, pp.338-342, Dec. 2009.

[5] Shent, B.; Huang, L.; Zhao, C.; Zhou, Z. & Kwak, K. “Energy Detection Based Spectrum Sensing for Cognitive Radios in Noise of Uncertain Power”, Proc. Int. Symp. Communications and Information Technologies ISCIT 2008, 2008, 628-633

[6] End to End efficiency [E^3] white paper, “Spectrum Sensing”, Nov. 2009.

[7] S. J. Shellhammer “Spectrum Sensing in IEEE 802.22,” Qualcomm Inc. 5755 Morehouse Drive San Diego, CA 92121, 2009.

[8] Pooyan Amini, Daryl Wasden, Arash Farhang, Ehsan Azarnasab, Peiman Amini, Behrouz Farhang-Boroujeny, “Cognitive spectrum assignment,” *2008 Software Defined Radio Technical Conference and Product Exhibition*, Oct. 26-30, 2008, Washington D.C., Paper # 1.3-5. Conference Paper, published, 2008.

[9]. D. Cabric, A. Tkachenko, R. Brodersen, “Experimental study of spectrum sensing based on energy detection and network cooperation”, Proc. of the first international workshop on Technology and policy for accessing spectrum,2006.

[10] Khaled Ben Letaief. "Cooperative Spectrum Sensing", Cognitive Wireless Communication Networks, 2007

[11] S. Haykin, “Cognitive radio: Brain-empowered wireless communications,” IEEE Journal on Selected Areas in Communications, February 2005.

[12] F. Penna, C. Pastrone, M. A. Spirito, and R. Garello, “Energy detection spectrum sensing with discontinuous primary user signal,” IEEE Proc. ICC, Jun. 2009.

[13] D. Cabric, A. Tkachenko, R. Brodersen, “Experimental study of spectrum sensing based on energy detection and network cooperation”, Proc. of the first international workshop on Technology and policy for accessing spectrum,2006.

[14] H. Urkowitz, “Energy detection of unknown deterministic signals,” *Proc. of IEEE*, pp. 523–531, Apr. 1967.

[15]. R. Tandra, A. Sahai, “SNR Walls for Signal Detection”, *IEEE Journal of Selected Topics in Signal Processing*, vol.2, no.1, pp.4-17, Feb. 2008.

[16] *GNURadio website, http://www.gnu.org/software/gnuradio/.*

[17] Y. Zeng and Y. C. Liang, “Spectrum-sensing algorithms for cognitive radio based on statistical covariances,” *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, May 2009.

[18] J. Mitola and G. Q. Maquire, “Cognitive radio: making software radios more personal,” *IEEE Pers. Commun.*,Vol. 6, pp. 13–18, Aug. 1999.

[19] *K. Arshad and K. Moessner, “*Impact of User Correlation on Collaborative Spectrum sensing for Cognitive Radio,*”* in proc. ICT Mobile Summit 2009, 10-12 June ‘09, Satander, Spain.

[20] D.Cabric.S.M., Mishra.R.W., Brodersen, “Implementation Issues in Spectrum Sensing”, In Asilomar Conference on Signal, Systems and Computers, November 2004.

[21] Zhi Yan, Zhangchao Ma, Hanwen Cao, Gang Li, and Wenbo Wang, “Spectrum Sensing, Access and Coexistence Test bed for Cognitive Radio Using USRP”, 4th IEEE International Conference on Circuits and Systems for Communications, 2008. ICCSC 2008, 26-28 May 2008.

[22] GNURadio Wiki, http://gnuradio.org/redmine/wiki/gnuradio

[23] Wikipedia, http://en.wikipedia.com

[24] SCA based open source software defined radio, http://ossie.wireless.vt.edu/

[25] High Performance Software Defined Radio (HPSDR), http://hpsdr.org/

[26] Flex Radio, http://www.flex-radio.com/

Appendix A

Code of USRP2_spectrum.py is written in python language. The modified part of the code is in italics. The rest of the code is taken from usrp_spectrum_sense.py which is a standard routine available in GNU Radio, and works for USRP first release only. There might be some indentation errors in the routine which can easily be fixed by copying the code in any standard python editor e.g. IDLE.

```
#!/usr/bin/env python
# Copyright 2005,2007 Free Software Foundation, Inc.
# This file is part of GNU Radio


from gnuradio import gr, gru, eng_notation, window
from gnuradio import usrp2
from gnuradio.eng_option import eng_option
from optparse import OptionParser
import sys
import math
import struct


class tune(gr.feval_dd):
    """
    This class allows C++ code to call back into python.
    """
    def __init__(self, tb):
        gr.feval_dd.__init__(self)
        self.tb = tb

    def eval(self, ignore):


      try:

          new_freq = self.tb.set_next_freq()
          return new_freq

        except Exception, e:
          print "tune: Exception: ", e


class parse_msg(object):
    def __init__(self, msg):
        self.center_freq = msg.arg1()
        self.vlen = int(msg.arg2())
        assert(msg.length() == self.vlen * gr.sizeof_float)

        t = msg.to_string()
        self.raw_data = t
        self.data = struct.unpack('%df' % (self.vlen,), t)


class my_top_block(gr.top_block):

    def __init__(self):
        gr.top_block.__init__(self)

        parser = OptionParser(option_class=eng_option)
        parser.add_option("-e", "--interface", type="string", default="eth0", help="Select
ethernet interface. Default is eth0")
        parser.add_option("-m", "--MAC_addr", type="string", default="", help="Select
USRP2 by its MAC address.Default is auto-select")
        parser.add_option("-a", "--start", type="eng_float", default=1e7, help="Start
ferquency [default = %default]")
        parser.add_option("-b", "--stop", type="eng_float", default=1e8,help="Stop
ferquency [default = %default]")
        parser.add_option("", "--tune-delay", type="eng_float", default=10e-1,
metavar="SECS", help="time to delay (in seconds) after changing frequency
[default=%default]")
        parser.add_option("", "--dwell-delay", type="eng_float",default=100e-1,
metavar="SECS", help="time to dwell (in seconds) at a given frequncy
[default=%default]")
        parser.add_option("-g", "--gain", type="eng_float", default=None,help="set gain in dB
(default is midpoint)")
        parser.add_option("-s", "--fft-size", type="int", default=256, help="specify number of
FFT bins [default=%default]")
        parser.add_option("-d", "--decim", type="intx", default=16, help="set decimation to
DECIM [default=%default]")
        parser.add_option("-i", "--input_file", default="", help="radio input file",
metavar="FILE")

        (options, args) = parser.parse_args()

        if options.input_file == "":
            self.IS_USRP2 = True
        else:
            self.IS_USRP2 = False

        self.min_freq = options.start
        self.max_freq = options.stop

        if self.min_freq > self.max_freq:
            self.min_freq, self.max_freq = self.max_freq, self.min_freq   # swap them
            print "Start and stop frequencies order swapped!"

        self.fft_size = options.fft_size



        # build graph

        s2v = gr.stream_to_vector(gr.sizeof_gr_complex, self.fft_size)

        mywindow = window.blackmanharris(self.fft_size)
        fft = gr.fft_vcc(self.fft_size, True, mywindow)
```

```
                        power = 0
        for tap in mywindow:
            power += tap*tap

        c2mag = gr.complex_to_mag_squared(self.fft_size)
                        #log = gr.nlog10_ff(10, self.fft_size, -20*math.log10(self.fft_size)-
10*math.log10(power/self.fft_size))


        # modifications for USRP2
        if self.IS_USRP2:
            self.u = usrp2.source_32fc(options.interface, options.MAC_addr)
            self.u.set_decim(options.decim)
            samp_rate = self.u.adc_rate() / self.u.decim()
        else:
            self.u = gr.file_source(gr.sizeof_gr_complex, options.input_file, True)
            samp_rate = 64e6 / options.decim

         self.freq_step =0.75* samp_rate
        self.min_center_freq = self.min_freq + self.freq_step/2
        nsteps = math.ceil((self.max_freq - self.min_freq) / self.freq_step)
        self.max_center_freq = self.min_center_freq + (nsteps * self.freq_step)

        self.next_freq = self.min_center_freq

        tune_delay  = max(0, int(round(options.tune_delay * samp_rate / self.fft_size)))  # in
fft_frames
        dwell_delay = max(1, int(round(options.dwell_delay * samp_rate / self.fft_size))) # in
fft_frames

        self.msgq = gr.msg_queue(16)
        self._tune_callback = tune(self)        # hang on to this to keep it from being GC'd
        stats = gr.bin_statistics_f(self.fft_size, self.msgq, self._tune_callback, tune_delay,
dwell_delay)


        self.connect(self.u, s2v, fft,c2mag,stats)

        if options.gain is None:
            # if no gain was specified, use the mid-point in dB
            g = self.u.gain_range()
            options.gain = float(g[0]+g[1])/2

    def set_next_freq(self):
        target_freq = self.next_freq
        self.next_freq = self.next_freq + self.freq_step

        if self.next_freq >= self.max_center_freq:
            self.next_freq = self.min_center_freq

        if self.IS_USRP2:
            if not self.set_freq(target_freq):
                print "Failed to set frequency to ", target_freq, "Hz"
        return target_freq
    def set_freq(self, target_freq):
        return self.u.set_center_freq(target_freq)

    def set_gain(self, gain):
         self.u.set_gain(gain)

def main_loop(tb):
    while 1:

        m = parse_msg(tb.msgq.delete_head())

        print m.center_freq
        print m.data

.
if __name__ == '__main__':
    tb = my_top_block()
    try:
        tb.start()              # start executing flow graph in another thread..
        main_loop(tb)

    except KeyboardInterrupt:
        pass
```

Appendix B

Raw data format

First few samples of data collected through USRP2 and GNU radio using USRP2_spectrum.py. The values in the parenthesis are the FFT magnitudes at the center frequency mentioned at beginning of brackets.

```
2402343750.0
(0.078479491174221039,  0.040066912770271301,  0.0055190813727676868,
0.0021366067230701447,  0.0011074023786932 23,  0.0019328504567965865,
0.0016706101596355438, 0.0020320124458521605, 0.0034777612891048193,
0.0014919996028766036,  0.0013331314548850 06,  0.0016256794333457947,
0.0018371485639363527, 0.0019112494774162769, 0.0021049873903393745,
0.0038519951049238443, 0.0029116699006408453, 0.0028225337155163288,
0.0045247073285281658, 0.0043828110210597515, 0.0036936171818524599,
0.0043287002481520176,  0.0051233791746199131,  0.004700744990259409,
0.0052246819250285625, 0.0079239737242460251, 0.0079878270626068115,
0.0072058727964758873,  0.0083901183679699898,  0.012011573649942875,
0.010452025569975376,   0.011384587734937668,   0.010718739591538906,
0.010121340863406658,  0.0098102204501628876,   0.023751826956868172,
0.028328873217105865,   0.016683906316757202,   0.021510237827897072,
0.029274577274918556,    0.02120569534599781,   0.018909499049186707,
0.017359867691993713,   0.021779812872409821,   0.028562052175402641,
0.060994364321231842,   0.085704058408737183,   0.062144704163074493,
0.07691742479801178,   0.087469212710857391,   0.061169750988483429,
0.045997627079486847,   0.036451626569032669,   0.047624539583921432,
0.078885406255722046,     0.2981133759021759,     0.5619933605194091 8,
0.68123906850814819,      0.5712742805480957,      1.022626519203186,
1.815961480140686,       3.1547107696533203,       2.9904580116271973,
1.7410080432891846,       1.8966439962387085,       1.0949727296829224,
0.92414122819900513,     0.65502303838729858,     0.88575261831283569,
1.2561960220336914,       1.0833464860916138,       1.6493371725082397,
1.3696602582931519,        1.6419686079025269,        2.719914436340332,
2.2194290161132812,       1.4921739101409912,       1.3443087339401245,
1.8912926912307739,      1.1128083467483521,      0.46113380789756775,
0.67372077703475952,      0.99898803234100342,      1.1562153100967407,
1.8998949527740479,        1.767426609992981,       2.2555630207061768,
2.9818150997161865,       4.2489080429077148,       4.9478602409362793,
3.6183938980102539,       1.8145967721939087,       1.5198602676391602,
1.1647709608078003,      0.6286810040473938,      0.62178975343704224,
0.80838441848754883,       1.0932257175445557,       2.4948527812957764,
2.5064244270324707,        1.6184374094009399,        1.942987322807312,
2.6447768211364746,       1.5150642395019531,       1.6680817604064941,
0.85805624723434448,      0.80695647001266479,      0.5647236704826355,
0.77822285890579224,      0.97723042964935303,      1.0299559831619263,
1.066877007484436,       1.6649191379547119,       2.0907723903656006,
1.385168194770813,      1.2304015159606934,      0.62880885601043701,
0.77945518493652344,     0.60114175081253052,     0.75301861763000488,
0.52458691596984863,     0.36764374375343323,     0.44105544686317444,
0.81339508295059204,     0.50622117519378662,     0.48039990663528442,
0.57239365577697754,     0.62805533409118652,     0.63090991973876953,
0.44689875841140747,     0.17851966619491577,     0.14225435256958008,
0.077040821313858032,     0.13759393990039825,     0.16485799849033356,
0.15660923719406128,     0.11775496602058411,     0.26699408888816833,
0.35701039433479309,     0.25196456909179688,     0.11670123040676117,
0.05526901036500930 8,   0.04523083940148353 6,   0.031556162983179092,
```

```
0.055269010365009308,   0.045230839401483536,   0.031556162983179092,
0.012226605787873268,  0.009931655600666999 8,  0.0070943846367299557,
0.0052436715923249722, 0.0052291429601609707, 0.0065299035049974918,
0.0034304608125239611, 0.0031522943172603846, 0.0027866777963936329,
0.0019167417194694281,  0.001906857592985034,  0.0013906561071053147,
0.0012801015982404351, 0.0010967451380565763, 0.0025738335680216551,
0.011629209853708744,   0.021526094526052475,   0.011421794071793556,
0.0024552359245717525,                         0.00081654638051986694,
0.00074491609120741487,                        0.00087696971604600549,
0.00078586529707536101,                        0.00071032048435881734,
0.0006499909795820713, 0.00068799924338236451, 0.000849866250064224,
0.00079919432755559683,                        0.00075856858165934682,
0.00078854116145521402,                        0.00071194040356203914,
0.00067059422144666314,                        0.00084492011228576303,
0.00076157570583745837,                         0.0007594208000227809,
0.00067484361352398992,                        0.00076539855217561126,
0.00088167772628366947,                        0.00084834126755595207,
0.0008001718670129776,                         0.00080491398693993688,
0.00095775781664997339,                        0.00075492350151762366,
0.00084168644389137626,                         0.0007911412394605577,
0.0012192637659609318,                          0.0012803474674001336,
0.00072430918226018548,                        0.00075210718205198646,
0.00084508676081895828,                         0.0011267503723502159,
0.0014678185107186437,                          0.0011142620351165533,
0.00083363440353423357,                         0.0008323927759192884,
0.0007781044696457684,                         0.00071245571598410606,
0.0008046915172599256,                         0.00094067084137350321,
0.00068867631489410996,                        0.00078109797323122621,
0.00084381789201870561,                        0.00072479399386793375,
0.00095712661277502775,                         0.0008555956301279366,
0.00076716899638995528,                        0.00076011696364730597,
0.00097077444661408663,                         0.0008456491632387042,
0.00081668398343026638,                        0.00072018272476270795,
0.0010230715852230787,                         0.00076872180216014385,
0.00068192993057891726,                        0.00083688297308981419,
0.00079831457696855068,                         0.0010027461685240269,
0.00083487335359677672,                        0.00078005483373999596,
0.00081469607539474964,                         0.0007705554598942399,
0.00089630088768899441,                        0.00071952160215005279,
0.00078048795694485307,                        0.00071756489342078567,
0.00082102115266025066,                        0.00071246037259697914,
0.0012075777631253004,                          0.0011661506723612547,
0.00073863635770976543,                        0.00073408539174124599,
0.00086960755288600922,                          0.000859142339322716,
0.00086273468332365155,                        0.00085836776997894049,
0.0010911953868344426,                         0.00086930743418633938,
0.0008030780591070652,                         0.00081013055751100183,
0.0008304059156216681, 0.0013898427132517099, 0.0014285683864727616,
0.00084563024574890733,                        0.00078523502452298999,
0.00087360222823917866,                         0.0013947460101917386,
0.0014321762137115002, 0.0013469539117068052, 0.0011599337449297309,
0.0011492453049868345, 0.003844299353659153, 0.037321273237466812)
2407031250.0
(3.1974740028381348,      4.031559944152832,      3.217695951461792,
2.0523378849029541,      2.2461001873016357,      3.2930624485015869,
2.9177732467651367,      3.1102924346923828,      2.8934581279754639,
3.1583716869354248,      5.5028438568115234,      3.4998223781585693,
3.728071928024292,      2.8119258880615234,      4.1116776466369629,
4.5282888412475586,      6.4161314964294434,      5.9056835174560547,
5.3880019187927246,      5.1316804885864258,      3.1722424030303955,
3.2820501327514648,      3.5894057750701904,      5.1031947135925293,
5.9944367408752441,        4.7254638671875,        5.943519115447998,
4.9885139465332031,      3.4352085590362549,      3.8861455917358398,
3.0611209869384766,      4.1641387939453125,      4.6584477424621582,
3.3806934356689453,       3.6395854949951172,      4.581965446472168,
4.0355067253112793,       6.711235523223877,       9.182032585144043,
5.950444221496582,      5.0542440414428711,       4.2773265838623047,
4.3682861328125,        4.2800970077514648,        6.594294548034668,
6.3907628059387207,       7.563817024230957,      6.2335686683654785,
6.3701944351196289,      6.1517653465270996,      7.1196331977844238,
7.0584292411804199,      5.2938923835754395,      3.7152411937713623,
6.152287483215332,      8.7384729385375977,       9.4443597793579102,
13.17851448059082,      11.978425979614258,       10.985051155090332,
10.31685733795166,      6.4902448654174805,       6.4896612167358398,
6.7908363342285156,      4.5423426628112793,      3.9046883583068848,
4.681190013885498,      5.1105408668518066,       5.5710554122924805,
6.555757999420166,      7.9897575378417969,       8.5431900024414062,
7.2733640670776367,      7.8920340538024902,      6.0835604667663574,
4.9511837959289551,      5.4244108200073242,      5.5524086952209473,
7.0115170478820801,       7.6162238121032715,      7.339911937713623,
7.5073418617248535,      6.3882355690002441,      6.5758600234985352,
5.7620663642883301,      5.3743562698364258,      7.7268595695495605,
10.491754531860352,      8.4772605895996094,      8.8215827941894531,
8.545628547668457,      7.9041132926940918,       7.3362040519714355,
7.1336016654968262,      11.792904853820801,      10.975196838378906,
10.993707656860352,      11.894734382629395,      16.446784973144531,
14.538098335266113,       12.99225902557373,      11.927684783935547,
8.4462566375732422,      7.3576827049255371,      7.5367612838745117,
4.5616464614868164,       3.7589983940124512,       4.76129150390625,
5.8710637092590332,      5.2951393127441406,      6.4392209053039551,
6.3135218620300293,      4.8167567253112793,      6.2896203994750977,
6.3264636993408203,       3.5456728935241699,      3.480363130569458,
3.255685567855835,      3.4054141044616699,       3.8168199062347412,
4.3830351829528809,      3.6322879791259766,      3.2404756546020508,
2.7397422790527344,      2.3923275470733643,      2.0473501682281494,
1.7573552131652832,      2.1140027046203613,      2.6542408466339111,
1.76539146900177,       1.568912148475647,        1.4443670511245728,
1.5097736120223999,      1.4458366632461548,      1.0355499982833862,
0.98314499855041504,    0.94674640893936157,     0.99487966299057007,
1.4055907726287842,      2.5000534057617188,      1.6392966508865356,
1.4201083183288574,     0.81627821922302246,     0.66750556230545044,
0.41280713677406311,     0.36113214492797852,     0.7205967903137207,
0.97726327180862427,      1.1921614408493042,     1.4042747020721436,
1.6822177171707153,      2.2844400405883789,      3.5576410293579102,
4.3758697509765625,      2.656505823135376,      0.94340842962265015,
1.0595099925994873,     0.99709200859069824,     0.83766782283782959,
0.49527463316917419,     0.95600581169128418,     1.4233869314193726,
2.0502135753631592,      2.1766338348388672,      1.9012550115585327,
2.7550060749053955,       2.7532544136047363,      1.422130823135376,
1.1020327806472778,     1.4120019674301147,      0.93049532175064087,
1.9735193252563477,      2.9511003494262695,      1.4158855676651001,
1.5492707490921021,      2.7556734085083008,      2.8711717128753662,
2.4999613761901855,      2.2758064270019531,      2.3133275508880615,
0.8397904634475708,     0.90137410163879395,      1.2856817245483398,
0.92863726615905762,      1.0676389932632446,     0.9720306396484375,
1.8714859485626221,      1.8402211666107178,      2.3330845832824707,
2.4703466892242432,      2.1704912185668945,      2.3740203380584717,
3.1873459815979004,      2.3636045455932617,      1.2329649925231934,
1.166181206703186,      0.81706392765045166,      1.5437220335006714,
2.7041969299316406,      2.4289414882659912,      2.2527797222137451,
5.9352045059204102,      8.9075870513916016,      7.1699471473693848,
3.3433778285980225,      2.0493607521057129,      2.8909635543823242,
2.3855693340301514,       2.389784574508667,        1.5883164405822754,
0.86046326160430908,      0.69971579313278198,      1.2471919059753418,
2.473766565322876,       2.5478525161743164,        2.6224849224090576,
2.5137460231781006,       2.0163238048553467,       2.9764595031738281,
2.3626995086669922,       1.7347713708877563,       1.3939838409423828,
1.1714987754821777,       1.5879503488540649,       1.2804248332977295,
1.8526248931884766,       2.1849963665008545,       2.5127692222595215,
3.3121376037597656,       3.2454426288604736,       2.8538851737976074,
1.9961237907409668,       1.9406003952026367,       3.7038774490356445,
4.7797470092773438,       2.3262240886688232,       1.3763785362243652,
2.8627450466156006,       3.8520331382751465,       3.0856180191040039,
3.3258025646209717,       2.3263769149780273,       3.5551505088806152,
2.6326093673706055,       2.5623633861541748,       1.7378360033035278,
1.7956358194351196,       1.9629738330841064,       2.2246809005737305,
2.2838873863220215,       2.4823381900787354,       3.0893852710723877,
5.9786229133605957,       10.225701332092285,       9.9261703491210938,
5.7646760940551758)
```

Appendix C

MATLAB2010 scripts to import and plot the data collected through USRP2_spectrum.py

1) Script for making frequency, magnitude plot

```
load rec.dat
x= rec(:,1);
y= rec(:,2);
[n,p]=size(rec)
%b=200;
%z=filter(x,b,y)
figure (1)
plot(x,y);
legend('cycle1','cycle2','cycle3',2)
xlabel('Frequency Range')
ylabel('|FFT|')
title('FFT Magnitude')
%figure (2)
%plot(x,z,'k')
y=y/256;
y=sqrt(y);
y=20*log(y);
figure (2)

plot(x,y);
xlabel('Frequency Range')
ylabel('dB')
title('Gain Plot')
```

% rec.dat is the raw data file.

2) Script for making time, frequency and magnitude 3D plots

```
load ad58.dat
x= ad58(:,1);
y= ad58(:,2);
%[n,p]=size(twenty)
z=1:1:71
b=256;
%z=filter(x,b,y)
y=y/256;
y=sqrt(y);
y=20*log(y);
stem3(z,x,y,'k')
title('Time,Freq,Gain 3D plot')
xlabel('Time')
ylabel('Frequency Range')
zlabel('dB')
```