2016

Implementation and evaluation of a software defined radio based radiometer

Matthew Erik Nelson
Iowa State University

Implementation and evaluation of a software defined radio based radiometer

by

Matthew Erik Nelson

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:

Phillip H Jones, Major Professor

John Basart

Brian K. Hornbuckle

Iowa State University

Ames, Iowa

2016

DEDICATION

I would like to dedicate this thesis to my wife Jennifer who has stood by me through this long journey towards my Masters. She put up with my countless afternoons that I was gone working on my thesis and also helped edit my thesis.

I would also like to thank my parents, family and friends without whose support and encouragement enabled me to complete this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Phillip Jones for his guidance, patience and support throughout this research and the writing of this thesis. I would also like to thank Dr. John Basart who patiently spent many hours with me going over both the fundamentals and details with radiometers. I would like to thank Dr. Brian Hornbuckle for his continued support and for allowing me to utilize his radiometer, lab and students towards my research. Finally I would like to thank Dr. Mani Mina who not only was instrumental in my undergraduate career but continued his support into my graduate path as well. If it was not for Dr. Mina hiring me as the Chief Engineer of the Spacecraft Systems and Operations Lab, I would not be here today.

ABSTRACT

This thesis explores using a software defined radio (SDR) to create a SDR-based radiometer that is capable of performing the same operation as a traditional radiometer and offers additional capabilities not found in traditional radiometers. Traditional radiometer requires careful design to ensure correct operation, by digitizing the RF signal as soon as possible and processing this signal in software, the hardware design of the radiometer can be simplified.

Digital radiometers have been explored before, but often use customized components. Software defined radio technology has become more widespread, and affordable Commercial Off The Shelf (COTS) SDRs are now available with high performance. This thesis leverages a COTS SDR technology to implement and evaluate a SDR-based radiometer. This will lower the cost of the radiometer and help make radiometers more accessible to a wider audience. The mapping of the functionality of a traditional radiometer to our proposed SDR-based radiometer is examined. Then an experimental evaluation of the performance between a traditional and SDR-based radiometer is conducted. Additionally this thesis explores how the implemented SDR-based radiometer can help mitigate radio frequency interference.

CHAPTER 1. INTRODUCTION

This thesis explores using software defined radio (SDR) technology to develop a radiometer that has performance on par with that of traditional radiometer. In addition to demonstrating a SDR-based radiometer can achieve similar performance, in terms of sensitivity and stability, as a traditional radiometer, it is shown that the inherent flexibility of SDR technology allows implementing functionality beyond what a traditional radiometer typically provides. Furthermore, by reducing cost and increasing flexibility, SDR technology may become an attractive path for broadening the accessibility of radiometry to the general research community.

Motivation. Remote sensing refers to recording, observing, and perceiving objects or events that are far away (i.e. remote)[Weng (2012)]. Since the object of interest is remote, we cannot physically interact with it using local measurement methods such as placing sensors or probes on the object. Remote sensing can be accomplished in a variety of ways and the following lists the basic approaches:

1. Visible light: Photography and Photogrammetry,
2. Thermal, Far infrared: Thermal Infrared Radiometry
3. Laser distance measurement: Lidar,
4. Radio Frequency (RF): Radiometry.

Each of these methods has an associated set of pros and cons, thus remote sensing often uses a combination of methods to paint a complete picture of an object. This thesis focuses on the apparatus used in radiometry to capture RF signals, called a radiometer.

Radiometry can be broken into two methods; active or passive. An active system is one in which a radio frequency signal (or pulse) is generated and transmitted to the object of interest. The reflection of this RF signal, or in some cases lack of reflection, gives us information about the object. A passive system is one in which no RF signal (or pulse) is generated. Instead, this type of radiometer simply listens to the RF energy that is naturally generated by the object, or that may be reflected from another source, such as the Sun. Radiometers have been focused on Earth for such purposes as helping scientists better understand its water cycle by monitoring ocean salinity [Hardy et al. (1974)] and soil moisture [Liu et al. (2013)]. Radiometers such as these are already in service on satellites such as the Soil Moisture and Ocean Salinity (SMOS) satellite [McMullan et al. (2008)] launched by the European Space Agency (ESA). Additional applications of radiometry include: assessing vegetation health and observing celestial objects[Ulaby and Long (2014)].

Problem Statement. While radiometers have proved to be an excellent tool for remote sensing and have been used in research applications for over fifty years, they have not made it into wide spread use. This is due to the fact that many traditional radiometers have the following hurdles:

1. They are expensive,
2. They require advanced knowledge to implement and use,
3. They are typically built and designed for a custom application.

This thesis aims to help address each of these hurdles as follows. The use of commercial off the shelf (COTS) parts and solutions will help reduce cost and the need for a custom design. Software will be used to define key components of the radiometer that have traditionally been implemented in hardware, and to ease adaptation to multiple applications. A user friendly graphical user interface (GUI) will help reduce the advanced knowledge required to implement and use the radiometer.

Contributions. The primary contributions of this thesis are:

1. Development of a software defined radio-based radiometer
2. Python based scripts for analyzing data generated by the radiometer
3. A Radio Frequency Interference (RFI) mitigation technique implemented using software defined radio technology

Organization. The remainder of this thesis is organized as follows. Chapter 2 gives a discussion of related works. Chapter 3 gives background on traditional radiometers and on software defined radios. Chapter 4 presents details on how software defined radio technology was used to implement a radiometer. Chapter 5 describes the experimentation setup used to verify and evaluate the operation of the implemented radiometer. Chapter 6 examines the results obtained from our performance evaluation experiments. Chapter 7 concludes this thesis and outlines avenues of future work.

CHAPTER 2. RELATED WORKS

Three areas closely related to the work in this thesis are: digital radiometers, software defined radio based radiometers, and radio frequency interference (RFI) mitigation. This chapter first presents two classifications of digital radiometers (hybrid and direct sampling). Next, software defined radio based radiometers are discussed in their own section, as a third classification of digital radiometer. This chapter concludes with a brief overview of the topic of RFI, which is important to consider when deploying radiometers.

2.1 Digital Radiometers

A digital radiometer replaces portions of a traditional radiometer's analog components with digital components[Ruf and Gross (2010)]. Two types of digital radiometers include: hybrid and direct sampling.

Hybrid. A hybrid radiometer uses a mixture of analog and digital components[Skou and Vine (2006)]. Often the analog voltage output from the diode of a square law detector, which is used to indicate the total power observed, will be digitized.

Direct Sampling. A direct sampling radiometer can be considered a type of hybrid radiometer that directly samples the incoming RF signal and then uses digital signal processing techniques to extract total power information. As an example, Iowa State University (ISU) owns a 1.4 GHz, dual polarization, correlating radiometer that uses direct sampling. It was built by the University of Michigan and put into service at ISU in 2006 [Erbas et al. (2006)]. This radiometer takes the RF signal and using analog

components amplifies and filters the signal, and then sends it to an analog to digital converter. The radiometer is designed to operate in the protected spectrum of 1400 MHz to 1420 MHz. Unlike other radiometers discussed, this radiometer samples the incoming RF signal at 1400 MHz. This method allows for power information to be extracted, however the full signal can not be recreated due to Nyquist's theorem [Fischman (2001)].

Both hybrid and direct sampling radiometers are designed to retain only the total power information contained in a RF signal. While measuring total RF power is the primary purpose of a radiometer, as it will be discussed later, retaining phase and frequency information can be useful as well.

2.2 Software Defined Radio Based Radiometers

Software defined radio based radiometers can be considered a relatively new subclass of digital radiometer. With the advent of software defined radios that are now available, there has been increasing interest in applying this technology to radiometers. This section discusses two works that have explored using software defined radio technology in radiometry from the Shirley's Bay Radio Astronomy Consortium and from Grand Valley State University.

Shirleys Bay Radio Astronomy Consortium. The Shirleys Bay Radio Astronomy Consortium (SBRAC) made use of a software defined radio to restore a radio telescope used for radio astronomy. They attached a software defined radio to their eighteen meter radius dish to obtain astronomical information by observing the hydrogen line located at 1420.4058 MHz in the RF spectrum[Leech and Ocame (2007)]. Marcus Leech, who headed SBRAC, contributed software to GNURadio specifically to support radio astronomy applications. This branch of GNURadio was used as the software base used in this thesis [Leech (2006)].

While parts of the GNURadio software used in this thesis were derived from Marcus Leech's work, additional features were added such as offending signal detection, offending signal mitigation, and a software implemented noise generator. Additionally, elements of the graphical user interface (GUI) were enhanced to aid in data visualization and analysis data. For example, a waterfall display of a signal spectrum over time was implemented.

Grand Valley State University. In 2013, the University of Illinois and Grand Valley State University built a software defined radio based radiometer to listen to emissions from Jupiter[Behnke et al. (2013)]. They custom built the hardware portion of their software defined radio using an Analog Devices analog to digital converter (AD9460) and a Xilinx (Spartan-3E-500) FPGA. They also implemented a RF front end to filter and amplify the incoming RF signal. The software side of their radiometer was composed of: 1) GNURadio for low-level communications with their software defined radio and, 2) Python script and WxGUI to implement a higher level user interface. The students reported that their SDR based radiometer worked well at a low price point. One aspect that differentiates the work in the thesis from Grand Valley State University's work is that they build their own custom hardware for their software defined radio, while in this work off the shelf components were used with an aim of making radiometers more wildly accessible to the research community.

2.3 Radio Frequency Interference (RFI) Mitigation

When an RF signal generated by a source other than the object (or phenomena) of interest interferes (i.e. masks or contaminates) with the RF signal of interest this is referred to as radio frequency interference (RFI). Radio frequency interference is a common problem with nearly all radiometers because they are highly sensitive receivers, thus even small unwanted signals can have a large negative impact on a radiometer based

experiment. It is for this reason certain frequency bands have been designated by the international community as "protected frequencies" for radiometer use. However, not all entities abide by these standards. For example, the satellite radiometer used by the Soil Moisture Ocean Salinity (SMOS) mission has had numerous issues with RFI [Kerr (2012)] skewing their data and in some cases making the data unusable for soil moisture measurements [Richaume (2012)].

The area of RFI detection and mitigation is still an active field of research [Forte et al. (2013)]. With respect to RFI detection, since radiometers typically do not retain spectral frequency information, statistical methods have been explored that look at variations in the received power to determine when RFI is occurring. One such method is the kurtosis statistic method[De Roo et al. (2007)] while another method is the polarization signature method. With respect to RFI mitigation, mechanical filters are used to selectively filter out the offending signals [Misra et al. (2012)].

While mechanical filters are an effective means of RFI mitigation, they add both weight and complexity to the radiometer. For example, multiple filters would be required to isolate and remove frequency bands that contain the offending signal(s). One idea this thesis explores is making use of frequency information and software-based digital filters for RFI detection and mitigation.

CHAPTER 3. BACKGROUND

This chapter gives background information on the operation of traditional radiometers and on software defined radios. We begin by looking at the major components of a radiometer and how it measures power. Next the metrics used to measure the performance of a traditional radiometer are discussed. Then an overview of how software defined radios operate is given. Finally, we review the tools used to develop a software defined radio based radiometer.

3.1 Radiometer Basics

A radiometer is a device designed to measure the power in electromagnetic emission by a material media due to the electron agitation within the material [Ulaby et al. (1981) in respect to remote sensing. This electromagnetic emission is the thermal noise of the object and can be correlated to the physical temperature of the object[Nyquist (1928)]. Because of this correlation, the amount of noise received is called the noise temperature and it is measured in Kelvin.

There are six stages common to all radiometers. They are:

1. Source (antenna or T_A)
2. Bandwidth (β),
3. Amplification (power gain or G),
4. Power detection (X^2),

5. Data smoothing,

6. Output (Voltage, rQ, Kelvin).

Figure 3.1 illustrates how a signal propagates through a radiometer. First, the signal from the source enters the antenna, T_A. Next the signal is filtered to a set bandwidth, β. This filtered signal is then amplified by Low Noise Amplifiers (LNAs) by a power gain of G. The power information is then extracted from the signal using a square-law detector, X^2. This device is a diode that operates in the square law region. While the diode is operating in this region, the output voltage is proportional to the square of the input voltage and is therefore proportional to the input power [Leinweber (2001)]. The voltage output from the square-law detector is then smoothed using an integrator with an integration time τ. Finally, the integrated voltage signal is then measured and recorded.

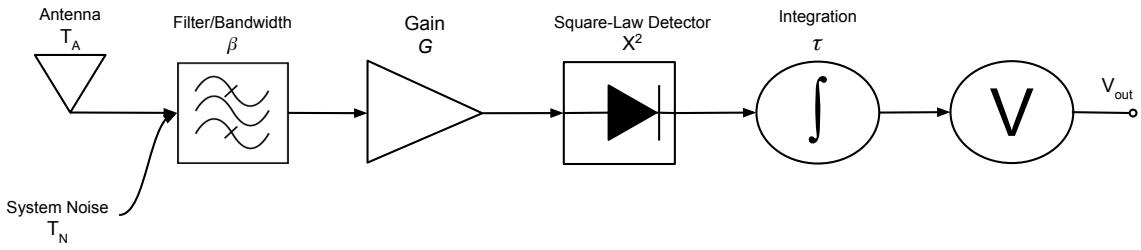

Figure 3.1 A total power radiometer block diagram

A non-physical object that is present in all radiometers is system noise, represented as T_N. System noise is noise that is generated from within the radiometer due to thermal agitation. A radiometer is designed to reduce system noise as much as possible by using low-loss components and amplifiers that are low noise, such as Low Noise Amplifiers (LNAs).

3.1.1 Power measurement

The voltage from a radiometer can be expressed as the sum of the noise voltages which the amplitude distribution of which is nearly Gaussian as defined by the central limit theorem. This results in a voltage mean of zero, but results in the root mean square (rms) value of the voltage to be greater than zero in a band limited sample. Because our voltage is an RMS value, our power and subsequently our noise power is a RMS value.

As shown in Equation 3.1, the power measured by an ideal radiometer is equal to the product of the thermal noise from the source (T_A), the power gain (G), the bandwidth (β) of the radiometer, and the Boltzmann constant ($k = 1.38 \times 10^{-23} J/K$).

$$P = k * \beta * G * (T_A)\ watts \tag{3.1}$$

While the components of an ideal radiometer do not contribute noise power (T_N) to the system, they do in a real radiometer. The impact of this internally generated and unwanted noise on the power measured by the radiometer is captured by Equation 3.2 where T_N is the system noise referred to the input of the system.

$$P = k * \beta * G * (T_A + T_N)\ watts \tag{3.2}$$

Power gain (G) and bandwidth (β) are important design parameters of a radiometer. While a large power gain is desired to amplify the source signal (T_A), the system noise is also amplified. The design of these amplifiers are then crucial to amplify the signal while minimizing contributions of the system noise.

The bandwidth of the source signal is typically wide (large) to maximize the sensitivity of the radiometer. The two primary limiting factors to bandwidth are: 1) hardware limitations (e.g. LNA operating limits), and 2) unwanted signals located at a number of frequencies (e.g. radio communication signals).

Low Noise Amplification. The method used by most radiometers to mitigate the system noise contributed during signal amplification is daisy chaining (i.e. cascading) devices called Low Noise Amplifiers (LNAs). The total amount of amplification we can expect from n LNAs that are cascaded is the sum of the power gain of each LNA shown in equation 3.3 where G_n is the power gain value for the nth amplifier.

$$G_{total} = G_1 + G_2 + G_3 + \cdots + G_{n-1} \quad dB \tag{3.3}$$

A performance metric of an LNA is the noise figure (NF). The noise figure gives us the difference between the actual noise output and an ideal amplifier with the same power gain and bandwidth attached to a matched load at the standard noise temperature (290 K). Another metric used is the noise factor (F). The noise factor is the ratio of the output signal to noise ratio to the input signal to noise figure. The noise factor (F) is related to the noise figure (NF) as shown in Equation 3.4.

$$NF = 10 * \log_{10}(F) \quad dB \tag{3.4}$$

For devices that are cascaded, the total noise factor is found by the Friis formula and results in equation 3.5 where F_n is the noise factor for the nth device.

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \frac{F_4 - 1}{G_1 G_2 G_3} + \cdots + \frac{F_n - 1}{G_1 G_2 G_3 \cdots G_{n-1}} \tag{3.5}$$

The key implication is that the first LNA in the cascade contributes the most system noise. As a consequence, it is critical that the first LNA has the smallest noise factor, while the remaining LNAs provide a majority of the signal power gain.

3.1.2 Radiometer performance metrics

Two criteria used to determine how well a radiometer performs are: 1) Sensitivity and 2) Stability. These criteria determine the smallest change in signal noise temperature (i.e. power) the radiometer can detect, and the amount of drift power measurements have over an extended period of time, respectively..

Sensitivity. Sensitivity of a radiometer is the smallest change in power that can be detected. A radiometer must be able to differentiate between signal noise received by the antenna (T_A) and the system generated noise (T_N).

Two methods to quantify the sensitivity of a radiometer are: 1) experimentally, and 2) analytically. Experimentally, sensitivity can be computed as the standard deviation of the measured power (assuming a stable radiometer). Analytically, sensitivity can be computed as a function of radiometer properties defined in Equation 3.6. The term Noise Equivalent Delta (Δ) Temperature ($NE\Delta T$) is often used interchangeably with sensitivity. As can be seen in Equation 3.6, sensitivity improves (i.e. becomes smaller) as the bandwidth (β) and/or integration time (τ) of the radiometer increases[Ulaby et al. (1981)].

$$NE\Delta T = \frac{T_A + T_N}{\sqrt{\beta * \tau}} \tag{3.6}$$

The following example illustrates the impact system generated noise has on a radiometers ability to detect changes in signal noise. Lets assume we want a sensitivity of 1 K. If there is no system generated noise (i.e. $T_N = 0$) and the received signal at the antenna is 200 K (i.e. $T_A = 200$), we can then calculate our receiver sensitivity by using Equation 3.6. Lets assume a bandwidth of 10 MHz (i.e. $\beta = 10x10^6$) and that our integration time is 40 milliseconds (i.e. $\tau = 0.04$). We can now take these values and put them in Equation 3.6 which results in Equation 3.7.

$$NE\Delta T = \frac{200 + 0}{\sqrt{10x10^6 * 0.04}} = 1K \tag{3.7}$$

Equation 3.7 gives us a result of 1 K for our sensitivity, which meets our goal. Because we do not have an ideal radiometer (i.e. $T_N \neq 0$), lets assume our system noise is 800 K (i.e. $T_N = 800$). Assuming that our bandwidth (β), our integration time (τ) and our antenna signal (T_A) is the same, we can now apply Equation 3.6, which results in Equation 3.8. This results in a sensitivity of 5 K, which is five times higher than our ideal sensitivity of 1 K.

$$NE\Delta T = \frac{200 + 800}{\sqrt{10x10^6 * 0.04}} = 5K \tag{3.8}$$

As can be clearly seen, system noise makes the job of detecting changes in signal noise more difficult[Skou and Vine (2006)]. Section 6.2.2 uses both the experimentation and analytical methods of finding sensitivity as a cross validation of the correctness of our SDR based radiometer.

Stability. Stability of a radiometer is a measure of the degree to which fluctuations we see are a result of the source and not a change occurring within the radiometer. Lets examine Equation 3.2, which defines the total power the radiometer receives. If our bandwidth (β), power gain (G), system noise (T_N), and Boltzmann constant (k) are constant, then the system is stable. If we can assume that our bandwidth is fixed and that our system noise (mean value) is constant, then this leaves our power gain (G) being the uncertain variable that may cause unwanted fluctuations[Evans and McLeish (1977)].

Radiometer instabilities are due to Low Noise Amplifier (LNAs) power gain fluctuations. Two factors cause these gain fluctuations: 1) fluctuations in the LNA's voltage supply, and 2) fluctuation in the LNA's physical temperature.

The impact these power gain fluctuations have on stability is given by Equation 3.9. Where:

- ΔT_G is the noise temperature fluctuation,
- ΔG is the LNA power gain fluctuation,
- G is the ideal power gain of the LNA and,
- T_{sys} is the combined antenna source (T_A) and system noise (T_N).

$$\Delta T_G = T_{sys}\left(\frac{\Delta G}{G}\right) \tag{3.9}$$

These power gain fluctuations can be controlled by closely monitoring and controlling both the voltage supply and temperature of the LNAs. However, this adds levels of complexity to the radiometer that may be impractical in some cases. As an alternative, modifications to the basic radiometer given in Figure 3.1 have been developed to compensate for these fluctuations. There are three common types of radiometers designed to account for power gain fluctuations. They are: Dicke, Noise injection, and Polarimetric (or Correlating) radiometers.

Dicke Radiometer. Figure 3.2 shows the block diagram of a Dicke radiometer, which switches between measuring the source signal (T_A) and a known reference signal (T_R)[Dicke (1946)]. By quickly switching between the source and reference signal at a frequency of F_S, a Dicke radiometer can reduce the impact of power gain fluctuations on its stability.

While a Dicke radiometer improves stability, it does so at the cost of not seeing the object of interest while it is measuring the reference signal. This reduces the sensitivity of the radiometer by a factor of 2, since half the time is used to observe the source signal. This changes the equation for the sensitivity of a Dicke radiometer as shown in Equation 3.10.

$$NE\Delta T = \frac{2(T_A + T_N)}{\sqrt{\beta * \tau}} \tag{3.10}$$

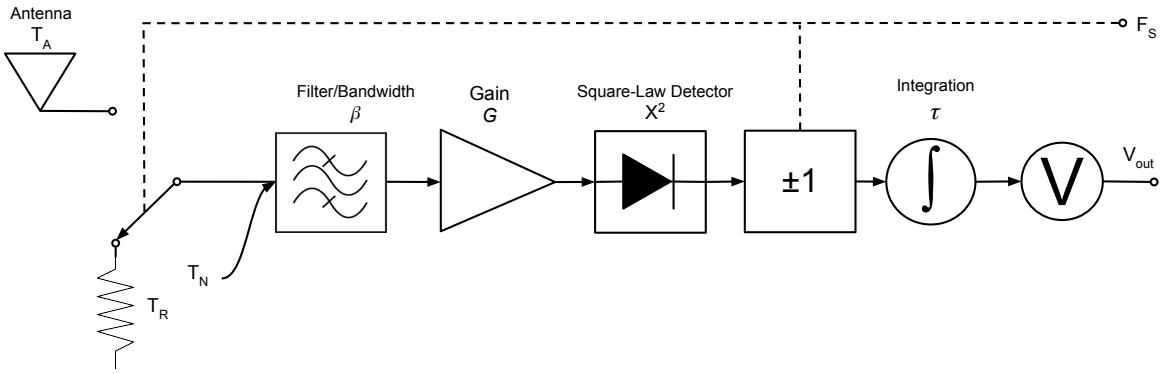

Figure 3.2 A block diagram of a Dicke radiometer

Noise Injection Radiometer. A noise injection radiometer is a variation of the Dicke radiometer, where a variable noise signal (T_I) is injected into the RF chain, as shown in Figure 3.3. The injected noise signal power is adjusted so when added to the source signal their sum equals the reference signal power. This improves stability by eliminating power gain fluctuations, but increases system noise (T_N). Much like the Dicke radiometer discussed earlier the sensitivity is reduced by a factor of 2 and the sensitivity is found to be similar to Equation 3.10 except our T_A is now equal to our known noise constant which we can define as T_R.

Polarimetric Radiometer. A polarimetric (or correlating) radiometer uses two polarized signals, referred to as vertically polarized (V-Pol) and horizontally polarized (H-Pol). In order to accomplish this, an antenna with dual polarization is used [Fujimoto (1964)]. Each polarized signal is fed into the radiometer and correlated. Because the source noise signal has polarization and the power gain fluctuations do not, the power gain fluctuations can be eliminated. This reduces power gain fluctuations to increase stability, while helping maintain sensitivity. The sensitivity of a correlating radiometer is shown in Equation 3.11, where $NE\Delta T_{sys}$ is the $NE\Delta T$ of both the vertical and horizontal receivers of the radiometers assuming that both receivers are identical in design and function. This

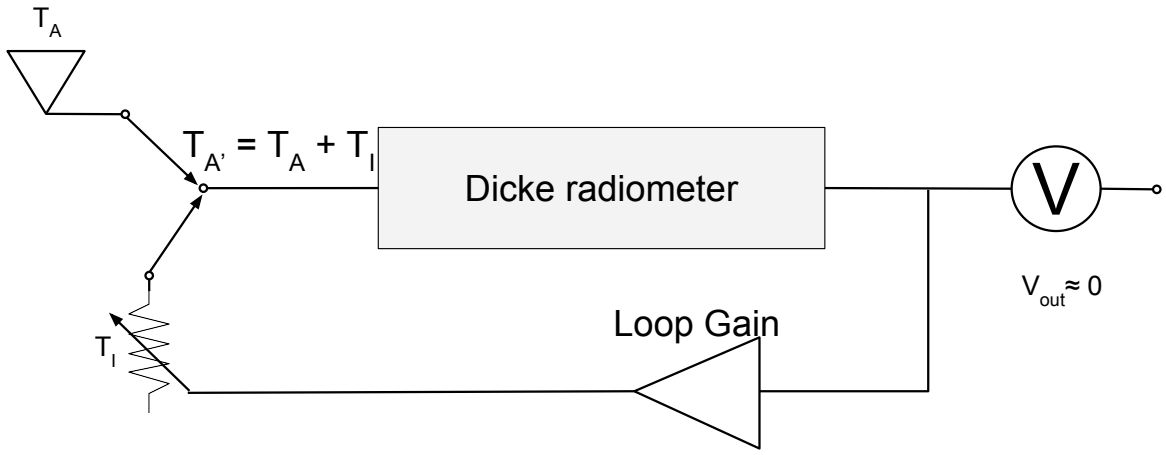

Figure 3.3 A block diagram of a Noise Injection radiometer

type of radiometer does increase the cost and complexity of the radiometer, since two identical receivers (one for each polarization) are required.

$$\frac{NE\Delta T_{sys}}{\sqrt{2}} = \frac{NE\Delta T_{V-Pol}}{\sqrt{2}} = \frac{NE\Delta T_{H-Pol}}{\sqrt{2}} \tag{3.11}$$

3.2 Software Defined Radios Basics

A Software Defined Radio (SDR) is a device that digitizes a received RF signal as soon as possible, and processes the digital representation of the signal using a computer, FPGA, or dedicated System on Chip (SoC). A canonical software defined radio architecture consists of a power supply, antenna, analog to digital converter, and a processing unit to carry out radio functions in software [Mitola (1995)]. An ideal software defined radio block diagram is shown in Figure 3.4.

A SDR can perform certain hardware functions in software (e.g. filtering), which provides flexibility over hardware solutions. Changes can be made by simply uploading new software or firmware to the system. This has a cost benefit as certain hardware components are no longer needed, and changes made in software do not require hardware

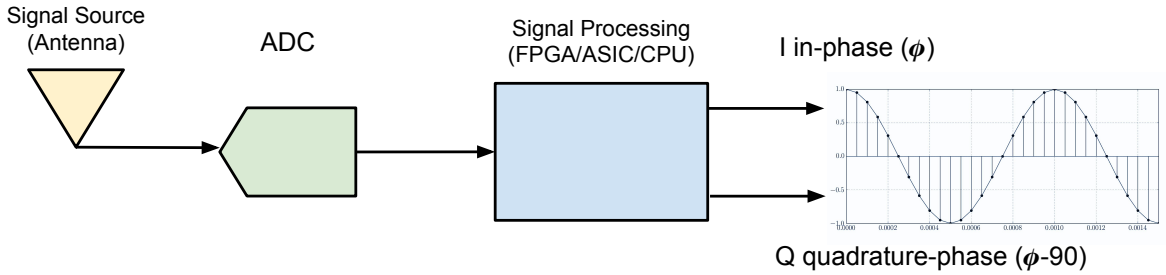

Figure 3.4 An ideal software defined radio

to be added, removed or modified.

A more realistic software defined radio is shown in Figure 3.5, where two items have been added. First, an amplification of the signal (power gain) is added to ensure the signal can be detected. Second, a mixer is often used to down-convert the high frequency RF source signal to a lower frequency so a less expensive analog to digital converter can be used. However, if the source signal frequency is already within the range of the analog to digital converter, then the mixer may be omitted.

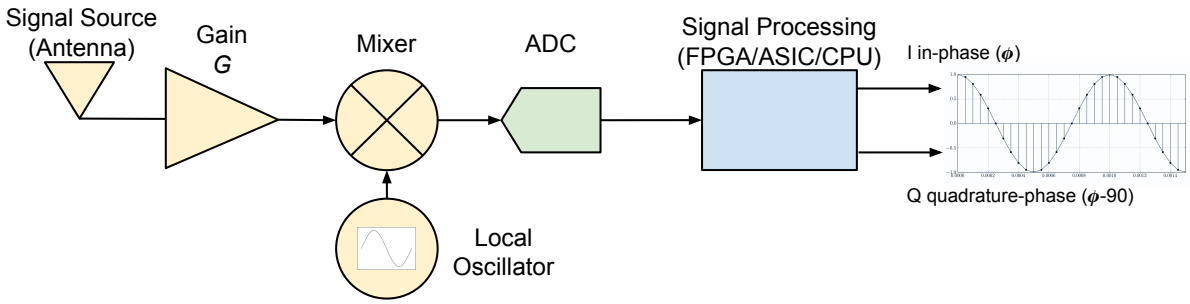

Figure 3.5 A block diagram of a typical software defined radio

Software Defined Radio Applications. Software Defined Radios (SDRs) can be used for a variety of applications, but have been primarily used in the area of communications. Some examples of these applications include: mobile communications, wireless local area networks, personal area networks, and digital broadcast. They appeal to applications where having the ability to change a modulation scheme or filter on the fly is desirable. In these respects, SDRs often outperform traditional hardware-only radios because of their ability to easily change their operations through software.

Early SDRs were expensive due to the cost of high speed analog to digital converters (ADCs) and high-end Field Programmable Gate Arrays (FPGA) that were required. In recent years, the cost of SDRs has decreased due to the cost of these key components decreasing. Also, while the cost of SDRs has gone down, their performance has increased. This has led to SDRs becoming feasible for use in many new ways [Jondral (2005)].

3.3 Software Defined Radio Development Platform

This section discusses the platform used to develop a software defined radio based radiometer. The hardware and software tools used are off the shelf. First, the hardware platform will be introduced, followed by the software platform.

3.3.1 Hardware platform

The hardware platform selected for this work is the Ettus Research Group N200 SDR shown in Figure 3.6. Its flexible architecture and ability to support a large bandwidth made the N200 an ideal hardware platform for software defined radio based radiometer development. The N200 has the following features that made it desirable for our specific application:

- Dual 14-bit ADC,
- 20 MHz bandwidth per channel,

- Modular daughter-board system for RF front end.

Figure 3.6 The USRP N200 from Ettus Research (Image from Ettus Research Website - www.ettus.com)

When selecting the hardware for this thesis, the requirements defined in Section 4.1 were examined. The bandwidth requirement was a deciding factor for hardware selection. It was decided that a minimum bandwidth of 20 MHz was desired. The 14-bit ADC was not a defined requirement, but was deemed to provide an adequate level of resolution. The N200 has a flexible architecture through the use of daughter-boards.

Figure 3.7 shows the overall architecture of the N200 SDR. A daughter board directly receives the RF signal and then outputs analog I (in-phase) and Q (quadrature phase) signals that are then sampled by the N200 14-bit A/D converter.

DBSRX2 Receiver. The daughter board selected for this work was the DBSRX2, shown in Figure 3.8. This daughter board is receive only, and operates between 800 MHz and 2400 MHz, which includes the frequency band required for this work (1400 MHz - 1425 MHz).

The DBSRX2 has the RF hardware needed to transform the received RF signal into in-phase and quadrature-phase outputs. This includes a programmable gain amplifier, a direct-conversion converter, a mixer, and finally a band-pass filter. Figure 3.9 provides a block diagram of the DBSRX2. First, the signal received is amplified by the Pro-

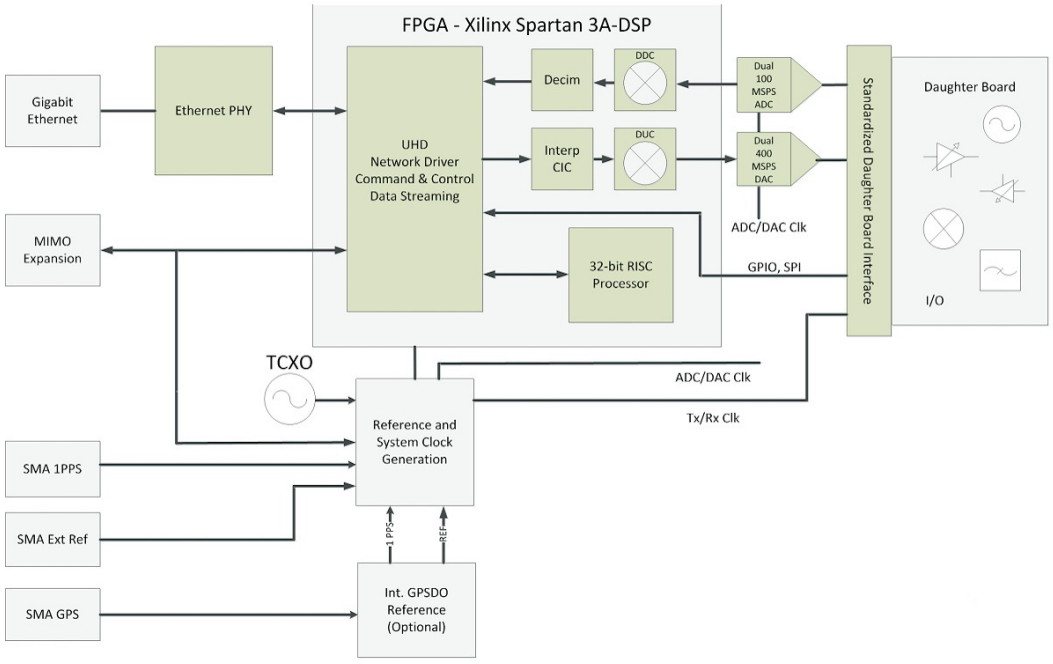

Figure 3.7 A block diagram of the Ettus N200 SDR. (Image from Ettus Research Website - www.ettus.com)

grammable Gain Amplifier (PGA). The PGA can be configured by software. Next, the signal goes into a direct-conversion integrated circuit (a Maxim 2112). This integrated circuit directly converts the RF signal into analog I (in-phase) and Q (quadrature phase) signals and is composed of an integrated mixer and band-pass filter.

These analog I and Q signals are then sent to the analog to digital converter to be digitized. The IQ signals are transmitted as differential signals to minimize noise. Once digitized, the digital I-Q values are sent to a FPGA to be processed, and then sent to the PC for software based signal processing.

3.3.2 Software platform

There are two pieces of software that are used with the software defined radio: 1) the firmware that is used in the FPGA of the N200, and 2) the software running on the host PC.

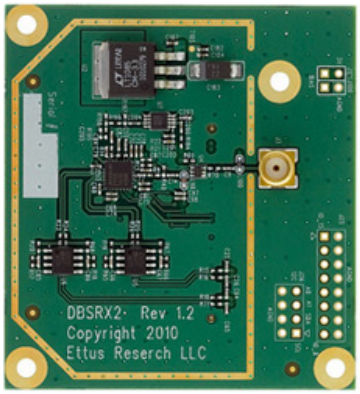

Figure 3.8 The DBSRX2 daughter board from Ettus Research (Image from Ettus Research Website - www.ettus.com)

The firmware provides low level processing of the signal before being sent to the software located on the PC. It also provides a link for controlling key aspects of the software defined radio, such as power gain, bandwidth and the center frequency. This firmware comes pre-loaded into the FPGA by Ettus Research, and can be upgraded using tools provided by Ettus Research.

The software on the host PC performs signal processing on the I/Q data. GNU-Radio was selected to be this software. GNUradio is an open source software package designed for software defined radio application development. It provides a GUI framework to create an interactive environment for the user, and is well supported by the N200 hardware.

GNURadio uses a combination of Python and C++, where Python handles the high level interface and C++ is used to implement drivers and low level interfaces to the hardware. This combination allows for a system that is easy to use, but still meets the performance required for handling large amounts of data.

GNURadio also has a rapid development tool called GNURadio companion (GRC). GRC is a simple to use graphical system for designing and building radio components in software. An example of GRC is shown in Figure 3.10.

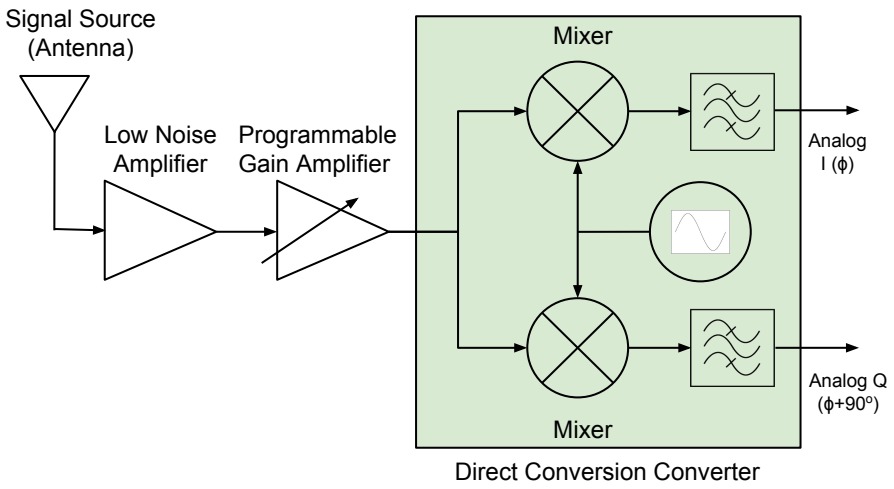

Figure 3.9 Block diagram of the DBSRX2 daughter board.

GNURadio Companion provides common functions, such as signal sources, signal processing and signal sinks, as blocks that can be picked and placed on the screen. Once placed, the blocks can be wired up, much like in LabView, and the flow of data can be controlled in this fashion. GNURadio Companion also includes blocks that allow for building a GUI interface, which can be used to display data and control the software defined radio.

If a block does not exist, it can be created. Because GNURadio uses Python, users can use this powerful and flexible language to build new blocks that can be imported into GRC.

Using GNURadio and GNURadio Companion, a software defined radio can be rapidly built with little programming experience. The excellent support of the hardware selected, and the ease of use made it an ideal development tool for building the necessary software for our SDR-based radiometer.

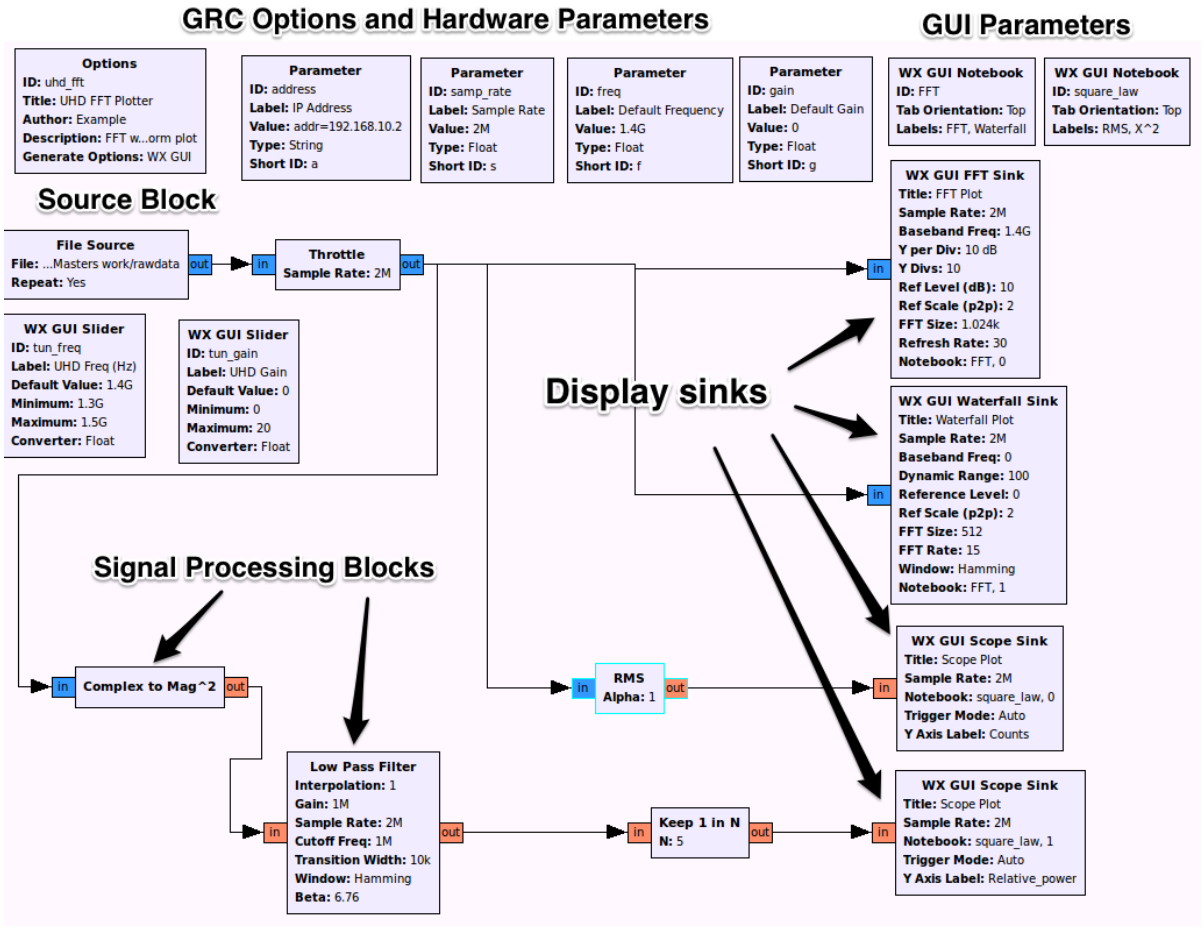

Figure 3.10 A screenshot of the GNURadio Companion editor program. Source: GNURadio

CHAPTER 4. SOFTWARE DEFINED RADIO BASED RADIOMETER IMPLEMENTATION

This chapter examines the implementation of a software defined radio based radiometer. First we will examine requirements of both a traditional radiometer and a Software Defined Radio (SDR) based radiometer. Next we will map the traditional radiometer components discussed in chapter 3 to their digital counterparts. Finally an overview of the controls and how data is displayed of a SDR-based radiometer is discussed.

4.1 Requirements

This section discusses the hardware and software requirements used to drive the selection of the hardware and software platforms use to implement our software defined radio based radiometer.

Hardware Requirements. The capabilities of existing traditional radiometers were the primary driving force in setting the requirements of our hardware development platform. Dr. Brian Hornbuckle from the Electrical and Computer Engineering and Agronomy department at Iowa State University was consulted with respect to key specifications of his radiometer. Additionally, the specifications of other radiometers were examined.

Table 4.1 Required Radiometer performance

Parameter	**Value**	**Units**
Minimum bandwidth	20	MHz
Operational frequency	1400 - 1420	MHz
$NE\Delta T$ (sensitivity)	1	Kelvin

Table 4.1 summarizes the specification of three parameters that were decided upon for selecting our hardware development platform, based on our investigation. This lead to the selection of the N200 software defined radio platform with a DBSRX2 daughter board from Ettus Research as our hardware platform. Section 3.3 provides more in depth information on the hardware used and why it was selected.

Software Requirements. Since an objective of this work was to help make radiometers more widely accessible to the general research and education community, a requirement of the development software was ease of use. Additionally, the user interfaces developed with these software tools needed to be easy to use, while providing sufficient computing efficiency for the signal processing required for radiometry.

GNURadio met the stated requirements. It includes a supplemental software package called GNURadio Companion (GRC), which uses a graphical interface for creating a radio environment. GNURadio and GRC are discussed in greater detail in Section 3.3.2.

4.2 Mapping Traditional Radiometer Functions to a Software Defined Radio Based Radiometer

The use of a software defined radio (SDR) to implement a radiometer requires mapping components of a traditional radiometer to SDR-based technology. This section presents the mapping of three such components for implementing our SDR-based radiometer. These components are: 1) power measurement, 2) data smoothing, and 3) bandwidth limiting and filtering.

4.2.1 Power measurement

A traditional radiometer uses a device called a square law detector to measure power. This device is a diode that operates in the square law region. While the diode is operating in this region, the output voltage (V_{out})is proportional to the square of the input voltage

(V_{in}) and is therefore proportional to the input power (P_{in}), where n is the constant of proportionality. This relationship is shown in Equation 4.1 [Department (2013)]. Figure 4.1 shows a simple circuit diagram and input and output voltages from this circuit.

$$V_{out} = nV_{in}^2 = nP_{in} \tag{4.1}$$

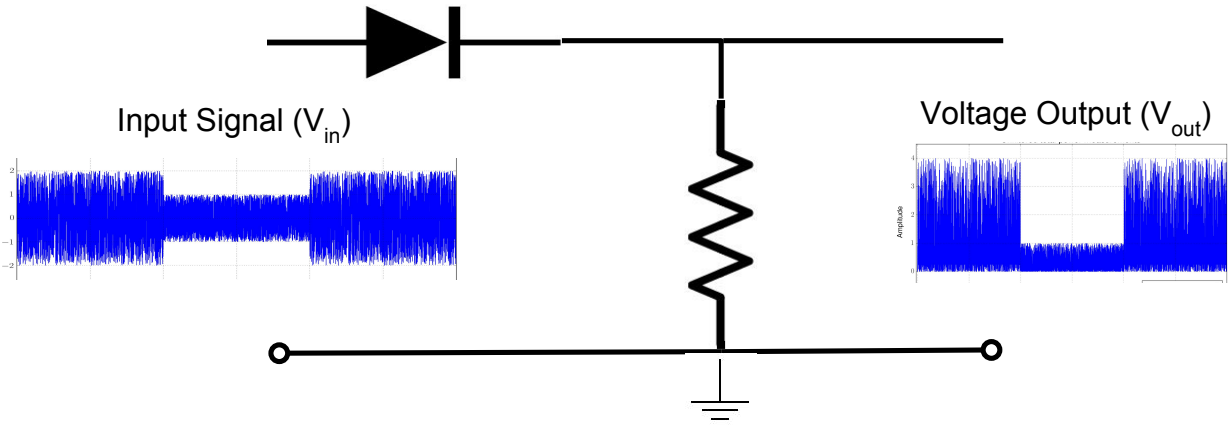

Figure 4.1 A simple diagram of a square-law detector.

The software defined radio based radiometer operates in a similar manner as a traditional radiometer. With the SDR-based radiometer, our signal is converted to I (in-phase) and Q (quadrature-phase) binary data. Adding the I and Q data allows us to recreate the signal. The I and Q data contains the magnitude of the in-phase and quadrature-phase information respectively. When summed this represents the magnitude of the original signal. To determine the total power, the magnitude is squared to produce an output that represents the total power of the signal (P_{out}). Equation 4.2 gives the mathematical representation of a software defined radio based radiometer implementation of a square law detector[Rashid et al. (2011)].

$$I^2 + Q^2 = P_{out} \tag{4.2}$$

Figure 4.2 shows the block within GNURadio Companion that performs the function of total power measurement. In Figure 4.2, the block labeled as A performs mathematically the power detection as shown in Equation 4.2. Block C decimates the data to reduce the sample size and thus reduce the file size of the data.

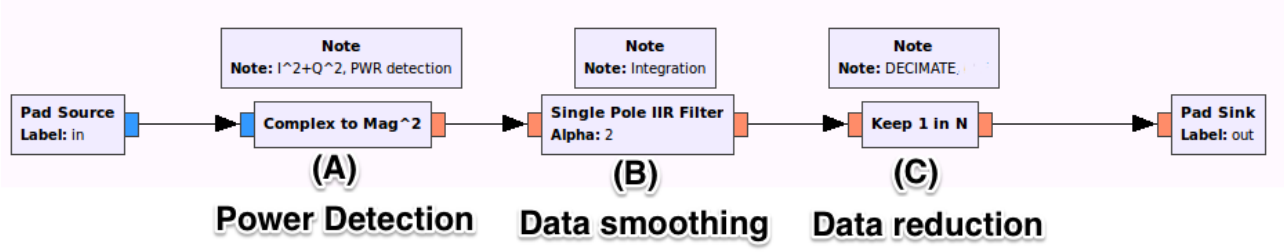

Figure 4.2 A block diagram of the power detection, low pass filter and decimation block used for total power measurements.

For both a traditional radiometer and a SDR-based radiometer, the output of this total power information will fluctuate rapidly. To smooth this signal we send it through a low pass filter, which is shown in Figure 4.2 as block B. This smoothing will be discussed in the next section.

4.2.2 Data smoothing

Output from the power detection system is noisy, which makes detecting small changes in power difficult (i.e. hinders sensitivity). Figure 4.3 shows an example of this noisy output from a square-law detector.

A traditional radiometer will use an integrator, which is equivalent to a low pass filter, to smooth the output from the square-law detector. This low pass filter is implemented as a simple RC circuit as shown in Figure 4.4. This low pass filter allows us to reduce the noise and this results in Figure 4.5.

To understand how a low pass filter works, we can look at the step response to this filter. Figure 4.6 shows the response in the top left and top right graph. We can now

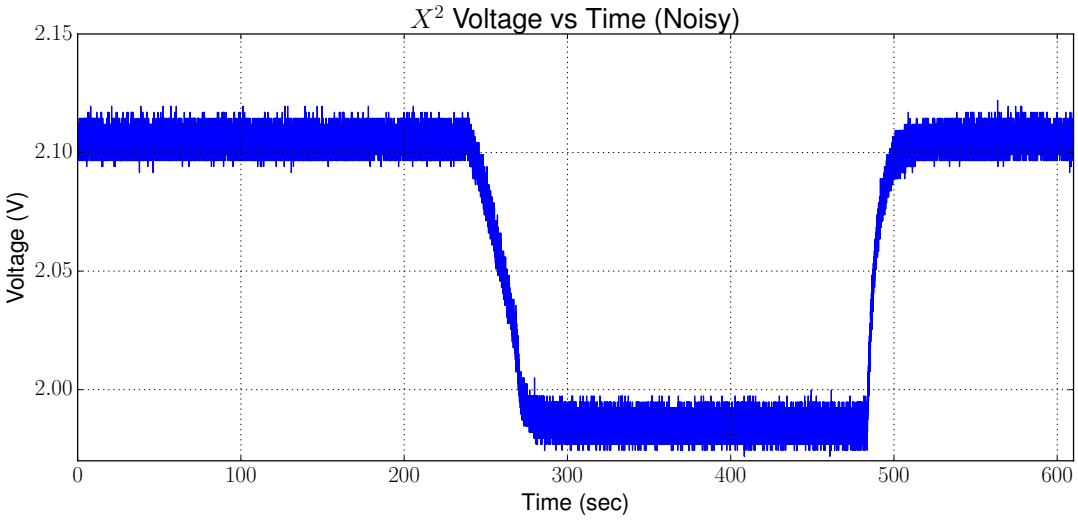

Figure 4.3 Power measurements from a square law detector before filtering

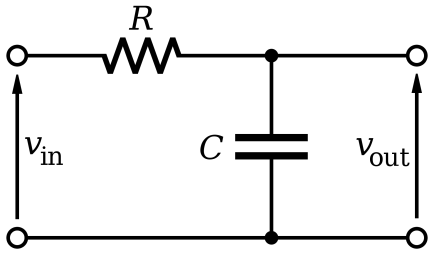

Figure 4.4 A simple RC circuit.

compare this analog low pass filter to our digital filter that will be discussed in more detail.

To implement this type of filter in our SDR-based radiometer an Infinite Impulse Response (IIR), also known as a recursive filter, is used. IIR filters are ideal for this application as they are well suited for achieving a long impulse response without having to perform a long convolution. The IIR filter that was created is known as a single pole low-pass filter.

The mathematical expression used to describe this filter is a recursive function shown in Equation 4.3. In Equation 4.3, our discrete output ($y[n]$) is defined as the input signal

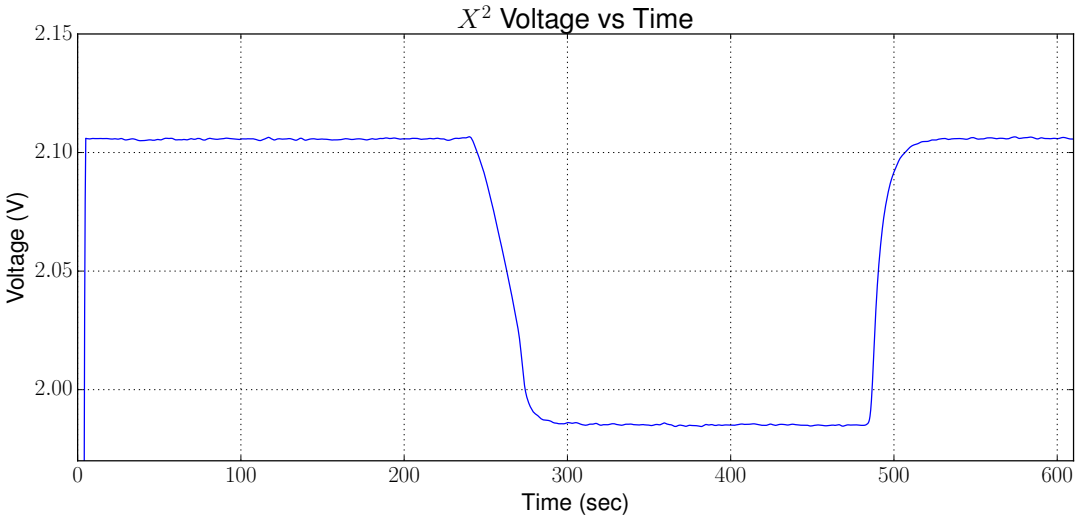

Figure 4.5 Power measurements from a square law detector after filtering.

($x[n]$) added to the previous sample calculated. The coefficients, a_0 and b_1, then define the response of the filter. The discrete input and output of this filter is shown in the lower left and right of Figure 4.6. This response is calculated using Equation 4.3 and the coefficients for a low pass filter, a_0 and b_1, are determined by Equation 4.4 and $0 \leq b_1 \leq 1$ [Smith et al. (1997)].

$$y[n] = a_0 * x[n] + b_1 * y[n-1] \tag{4.3}$$

The coefficients a_0 and b_1 are determined by Equation 4.4 for a low pass filter, where $0 \leq b_1 \leq 1$.

$$a_0 = 1 - b_1 \tag{4.4}$$

In Figure 4.4, the input is V_{in}, the resistance value is R, the capacitance value is C and the output is V_{out}. This circuit can be represented by Equation 4.5.

$$\frac{V_{in} - V_{out}}{R} = C\frac{dV_{out}}{dt} \tag{4.5}$$

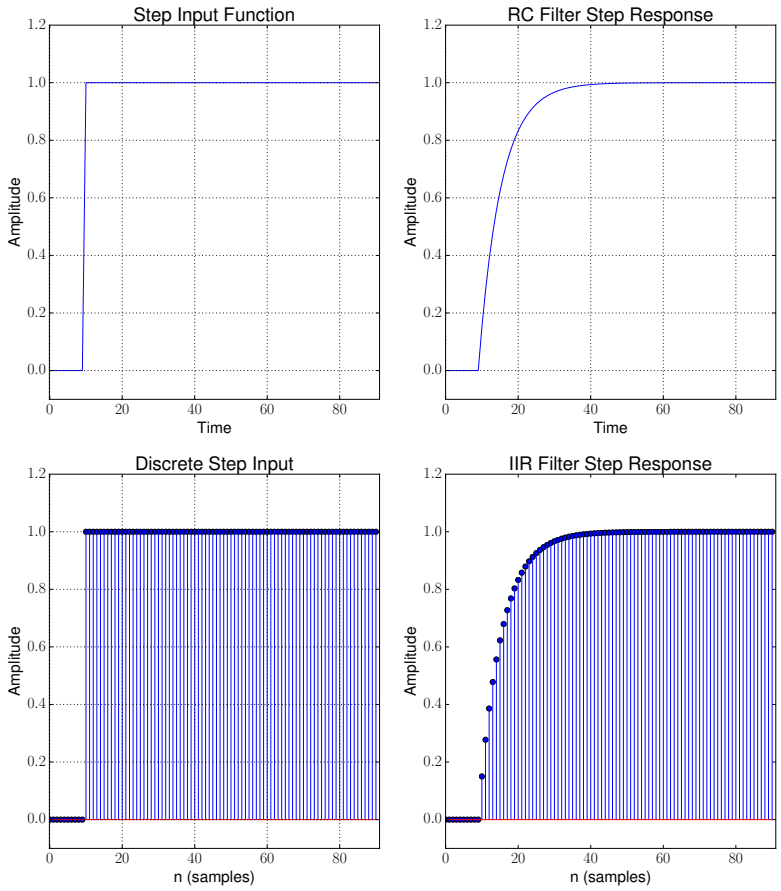

Figure 4.6 Graphs representing the analog and discrete response of a RC low pass filter to a step input.

Equation 4.5 represents the differential equation relating the input voltage V_{in} to the output voltage V_{out}. We can substitute the input to the RC circuit (V_{in}) as the input to Equation 4.3, x_n. The output of our RC circuit (V_{out}) can also be expressed as the output in Equation 4.3 which is y_n. However, in order to do this we must move from the continuous domain to the discrete domain that our digital filter operates in. This is done by showing the relationship between our sampling frequency f_s and our period, the time between samples, T and is shown in Equation 4.6.

$$T = TimeBetweenSamples = \frac{1}{f_s} \tag{4.6}$$

Next we rewrite our differential equation by substituting x_n and y_n into Equation 4.5, which results in an approximated finite difference equation shown in Equation 4.7.

$$\frac{x_n - y_n}{R} = C\frac{y_n - y_{n-1}}{T} \tag{4.7}$$

We can now solve for y_n algebraically and this results in our final Equation 4.8.

$$y_n = \frac{T}{T + RC}x_n + \frac{RC}{T + RC}y_{n-1} \tag{4.8}$$

Equation 4.8 shows an IIR filter that has a frequency response that closely approximates an RC circuit. The approximation improves as T approaches zero.

$$a_0 = \frac{T}{T + RC} \tag{4.9}$$

$$b_1 = \frac{RC}{T + RC} \tag{4.10}$$

To design the filter we need to look at what our desired cut-off frequency needs to be. For a RC filter, our resistance and capacitance define our cutoff frequency (f_c) and has the relationship shown in Equation 4.11.

$$f_c = \frac{\sqrt{3}}{2\pi RC} \tag{4.11}$$

Given our desired cutoff frequency, we can determine our combined RC value by rearranging Equation 4.11 algebraically to Equation 4.12.

$$RC = \frac{\sqrt{3}}{2\pi f_c} \tag{4.12}$$

The RC value is also referred to as the time constant of the circuit and we do not need to find the individual R and C values. An example of how we can find the coefficients of our IIR filter, given a desired cutoff frequency of f_c, is shown next.

Assume, for a low pass filter, we want a cutoff frequency (f_c) of 1000 Hz. Given this and using Equation 4.12 we can determine our time constant to be 2.757 x 10^{-4} seconds. If our sample rate is 1 MHz, then our T value is 1 x 10^{-6} second. Plugging these values into Equation 4.8 results in the coefficient $a_0 = 0.0036$ and $b_1 = 0.9964$.

GNURadio includes a program that allows us to specify our filter, and it generates the required coefficients and taps. Chapter 5 goes into more depth on this program.

4.2.3 Bandwidth limiting and filtering

Bandwidth limiting. Bandwidth limiting is the process of defining the frequency band over which a radiometer measures power. For a traditional radiometer, this is typically accomplished using analog filters. Within a SDR-based radiometer, bandwidth is controlled by the sampling rate of the Analog to Digital Converter (ADC). However, in the N200 SDR used in this thesis, the ADC clock is fixed at 100 MHz. Therefore the FPGA decimates the data to reduce the bandwidth from the ADC, but only by integer values of the divisor.

Filtering. There are instances when the frequency being observed requires additional filtering (i.e. RFI mitigation, see section 6.3). Traditional radiometers deploy analog band-pass or band-reject filters to accomplish this, while in this thesis we show similar functionality can be obtained by implementing software defined filters.

Next we will examine on how we control key functions in our SDR-based radiometer. These key functions have an impact on the performance of the SDR-based radiometer. We will also look at how we display the data collected from the SDR-based radiometer.

4.3 Software Defined Radio Based Radiometer GUI

A traditional radiometer will be designed with fixed parameters such as bandwidth and integration time. While changes can be made, it requires changes to the radiometer's

physical hardware. A SDR-based radiometer allows us to control several functions that defines how it operates as a radiometer. These changes can be made while the radiometer is operating since these operations happen in the digital domain and are defined by software. The following items are controlled by software in our SDR-based radiometer via the GUI interface: 1) Center frequency, 2) Bandwidth (β), 3) Integration time (τ), and 4) Power gain.

The GUI for the SDR-based radiometer was designed knowing that these parameters such as frequency, bandwidth and integration time can be changed. Because these controls impact both the performance of the radiometer and in what frequency range the radiometer operates in, it was important to have these controls clearly marked. Figure 4.7 shows a screen-shot of the GUI.

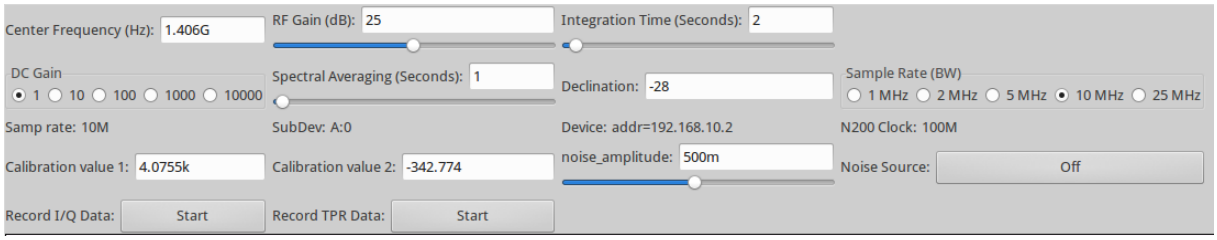

Figure 4.7 A screenshot of the interface made for communication with and controlling the software defined radio

The bandwidth, β, and our integration time, τ, both impact our sensitivity or $NE\Delta T$ of the radiometer as defined in Equation 3.6. The bandwidth can be changed by changing the sample rate of the SDR. The sample rate effectively controls the bandwidth in which the SDR is operating. This also gives us a band-pass filter as well, since the SDR will not respond to frequencies outside of this bandwidth. The integration time parameter is set by the user through the GUI and allows us to change the integration time in seconds. This directly controls the time constant for the IIR filter used to smooth the data.

The SDR-based radiometer in this thesis uses Low Noise Amplifiers (LNAs) to increase the power gain of the system. The DBSRX2 hardware, described in Section 3.3.1,

also has a Programmable Gain Amplifier (PGA) that is controlled by software. This allows additional power gain to be added or removed as needed. Next we will also look at how the data is displayed and stored in the software defined radiometer.

SDR-based radiometer data display. The information from the software defined radio can be displayed through GNURadio to show relevant information to the user. Currently, our SDR-based radiometer can display spectrum information and total power information. Spectrum information is used to verify the signal is clean and free of interference. Total power information is displayed as both un-calibrated values and calibrated values, assuming the correct calibration terms have been provided. This includes a display in a "ticker tape" graph and as a bar graph. Figure 4.8 shows a screen-shot of the total power display.

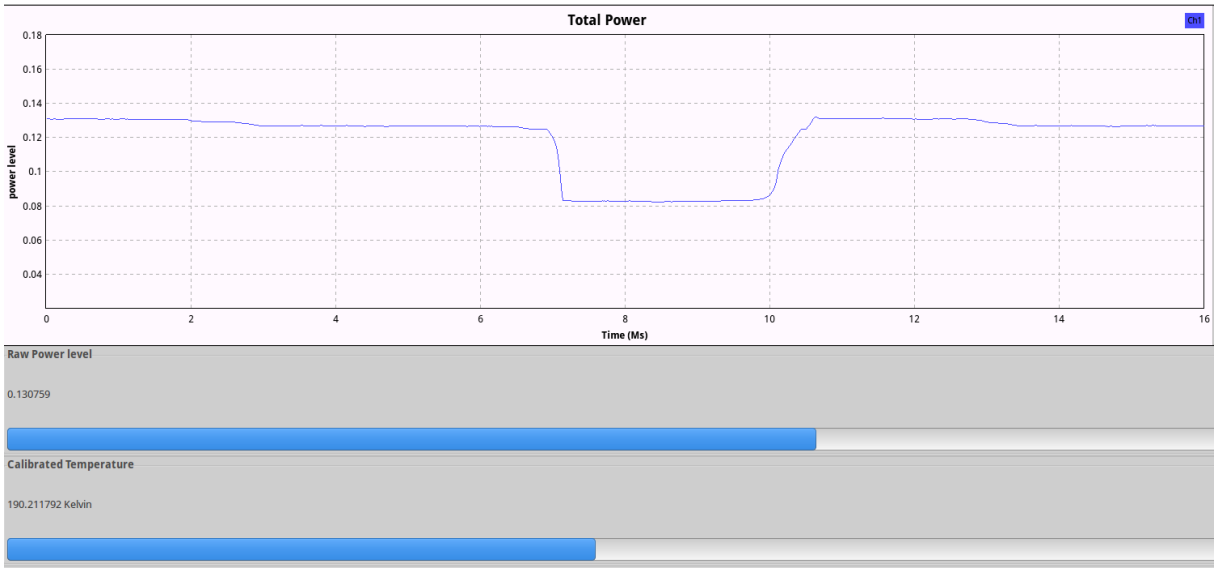

Figure 4.8 A screenshot showing the ticker tape display for the total power readings. In addition, raw and calibrated noise temperature is shown below.

CHAPTER 5. EVALUATION SETUP AND EXPERIMENTAL DESIGN

The experiments outlined in this chapter are used to demonstrate and verify that a software defined radio (SDR) base radiometer can perform on par with a traditional radiometer. In addition, these experiments are designed to demonstrate that a SDR based radiometer can provide functionality not typically found in traditional radiometers.

Four experiments are described. Experiment one verifies our SDR-based radiometer by comparing its operation to a square-law detector, a device typically used within a traditional radiometer. Experiment two evaluates the sensitivity and stability of our SDR-based radiometer. Experiment three evaluates our SDR-based radiometer's ability to mitigate an interfering signal in comparison to the behavior of a traditional radiometer in the presence of an interfering signal. Experiment four's purpose is to further examine the impact of our frequency notching approach for Radio Frequency Interference (RFI) mitigation on radiometer sensitivity.

5.1 Experiment I - Software Defined Radiometer Verification and Calibration

This experiment is designed to verify our SDR-based radiometer functions as expected. Its behavior is compared against an analog square-law detector, which is commonly used in traditional radiometers.

To verify the results of the information that the software defined radio is obtaining a square-law detector is used to measure the power of the incoming signal in parallel to the SDR-based radiometer. This signal is split using a power divider so that the information will be the same to both devices. This power divider ideally would divide the signal so that the resulting signal is 3 dB lower, plus the insertion loss, and is equal between both ports. The splitter used in this thesis was tested and verified that it does split the signal with a slight difference between port one and port two. This difference was measured to be .1 dB which is well within the .3 dB specified for this power splitter. This allows us to verify the software defined radio with a proven system.

5.1.1 Experimental setup

Figure 5.1 shows a block diagram of the experimental setup. A matched load is used to simulate our source signal. This matched load is then submerged in temperature baths. These baths use Liquid Nitrogen (LN2), which is known to boil at 77 Kelvin, and an ice water bath which is known to be at 273.15 Kelvin. The temperature of these baths were monitored with a thermometer that had an accuracy of ± 1 degree Celsius. The load was submersed in each bath for a minimum of 2 minutes to allow it to reach the same temperature as the bath. The physical temperature of this matched load is then the noise temperature the radiometer sees and can be used to calibrate the radiometer.

The radiometer RF front end provides the amplification needed for our experiments. Figure 5.2 shows an image of the RF front end, with the LNAs and band-pass filters marked. After the signal has been amplified, we divide the signal between the square-law detector (ADL5902) and the software defined radio (N200). The N200 is then connected to a personal computer running XUbuntu Linux and GNURadio.

The following hardware was used and has also been marked on Figure 5.1:

1. N200 Software Defined Radio with DBSRX2 Daughter-board

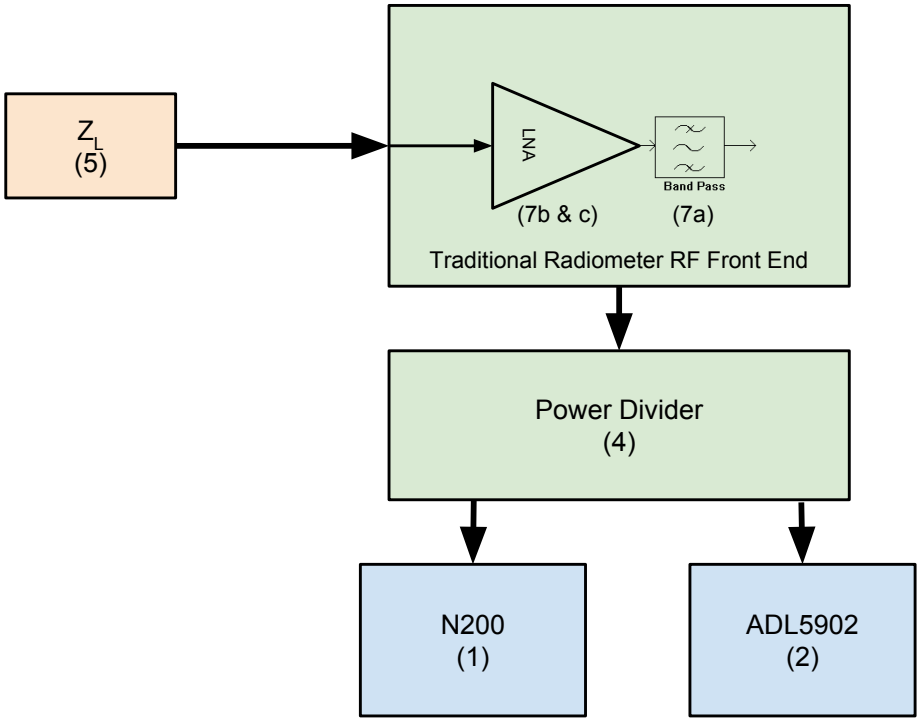

Figure 5.1 Block diagram of Experiment 1 setup.

2. ADL5902 Square-law detector

3. National Instruments USB-6009 Data Acquisition Unit (not shown in Figure 5.1)

4. ZN2PD-20-S+ Power Divider

5. 50-ohm matched load (Z_L)

6. Rigol DP832 Power Supply (not shown in Figure 5.1)

7. Radiometer RF Front End

   (a) 4 x Integrated Microwave Bandpass filters (1400 - 1425 MHz)

   (b) 2 x Miteq AMF-3F-01400147-30-10P LNA

   (c) 1 x Miteq AMF-2F-01400147-04-10P LNA

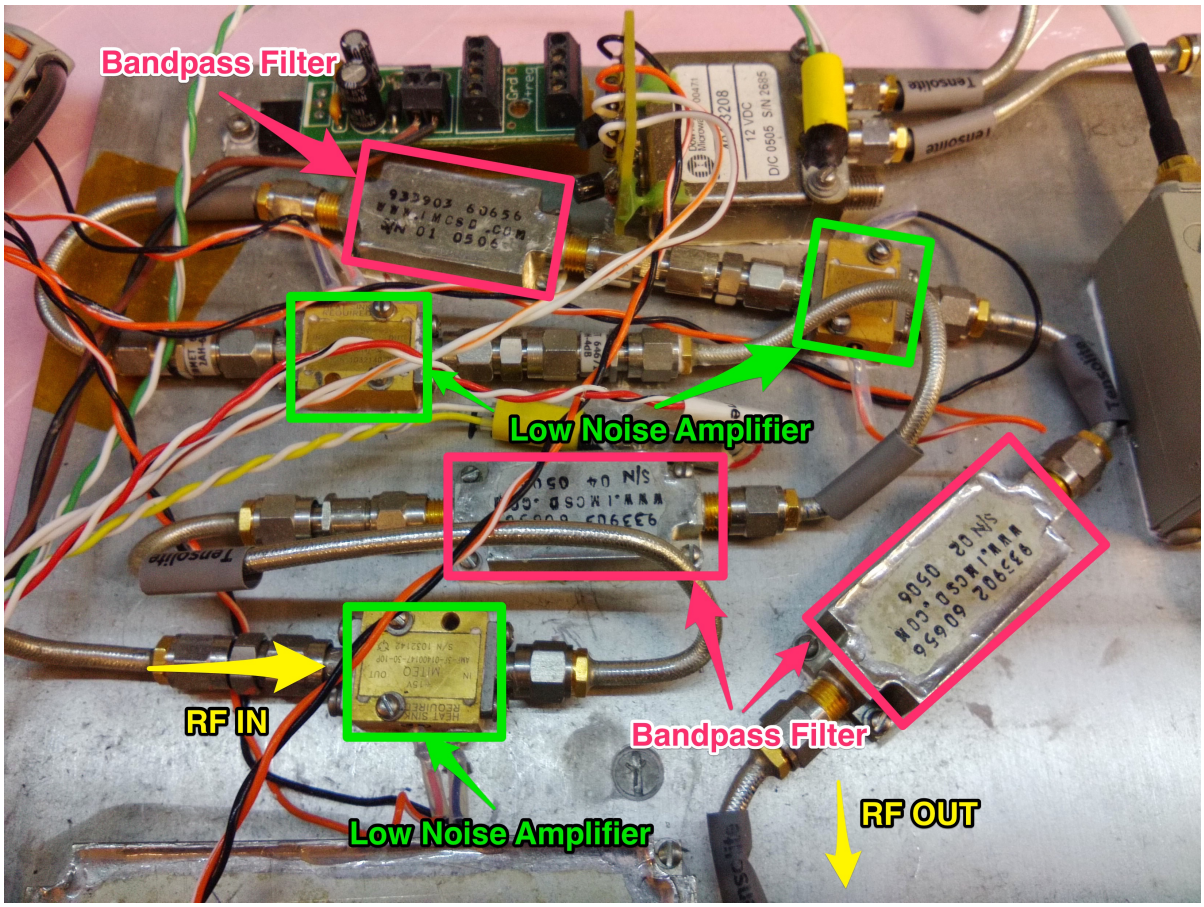

Figure 5.2 The radiometer RF front end, with LNAs and band-pass filters used in the experiments.

5.1.2 Data collection

Two sets of data are produced with this experiment. First, data is generated from the software defined radio using GNURadio. Second, data is generated from a data acquisition device (USB-6009) that is attached to the square-law detector. Data from each is stored to a local computer running the appropriate software.

Software Defined Radio Data. The software defined radio based radiometer is configured with a bandwidth of 10 MHz and an integration time of two seconds. It is centered on a frequency of 1.406 GHz which allows it to operate within the mechanical band-pass filters.

The data from the software defined radio is stored in files generated from GNURadio. GNURadio uses a sink block to output the data to either a screen, socket connection such as TCP/IP, or a file. As shown in Figure 5.3, a file sink block is used to output the data to a file. The flow of data to this sink is controlled by a valve block. This allows the user to turn on and off recording of the data.

Figure 5.3 The File Sink block used in GNURadio. Source: GNURadio

There are two types of files the SDR generates. The first type of file is the I (in-phase) and Q (quadrature phase) data points. This file is stored in little-indian format as complex values. Due to the sampling rate, it is not uncommon for this file to grow quite large, usually several gigabytes of data for a 10-15 minute run. However, this file can then be fed back through GNURadio later to be played back if needed, and it contains the information needed to completely recreate the signal.

The second file type is the total power values generated from the total power block in GNURadio. A diagram of this block can be found in Appendix A (Figure A.1), and its source code can be found in Appendix A. This file also uses a little-endian format, however this file only has real values. This file is also much smaller than the file that contains the I and Q data points due to decimation of the data. A typical file size is 50 - 100 kB for a 10 to 15 minute run.

Square-law detector data. The square-law detector used (ADL5902) outputs total power information as an analog voltage that is linearly proportional to the RF power measured. The Analog Devices ADL5902 is a single Integrated Circuit (IC) that contains a square-law detector and necessary amplification for the output signal and is shown in

Figure 5.5 . This device operates from 50 MHz to 9 GHz and can detect power as low as -60 dBm. The output voltage from the square-law is amplified to a range from zero to five volts with a calibrated output of 53.7 mV/dB.

Testing was done on the square-law detector to verify linearity and proper operation of the square-law detector. This test involved sending a known signal into the square-law detector and changing the amplitude at set intervals. This was then graphed and is shown in Figure 5.4. This graph shows that as we changed the input power in linearly, the output measured from the square law detector also changed linearly.

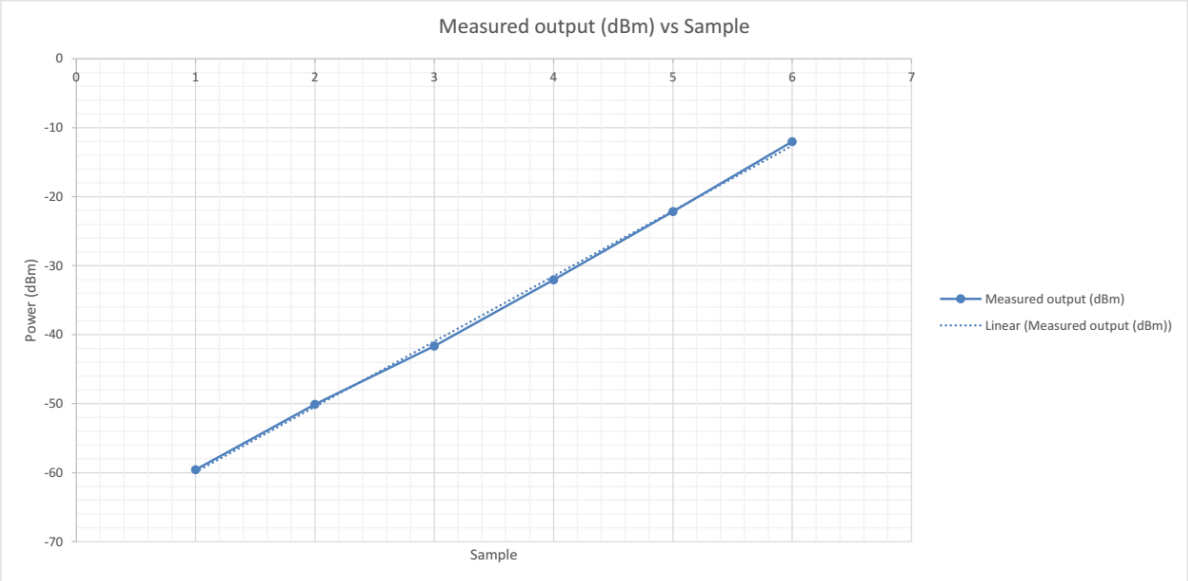

Figure 5.4 A graph of the power output of the square law detector.

To capture the voltage output from the ADL5902, a data acquisition unit (DAQ) is used. The National Instruments USB-6009 DAQ unit was selected as it met the requirements for an easy to use yet high enough resolution to obtain accurate information. The USB-6009 unit has 8 analog inputs that can sample at 48 KSPS with a resolution of 14-bits.

To use the USB-6009, a fairly simple Lab View program was created to obtain, display and store the data from the ADL5902. This program retrieved information from the

USB-6009 and stored the data in both Labview's binary format and in a more human friendly ASCII format. A GUI program, shown in Figure 5.6, is used to control and display the data. This made obtaining the data and using the device straightforward.

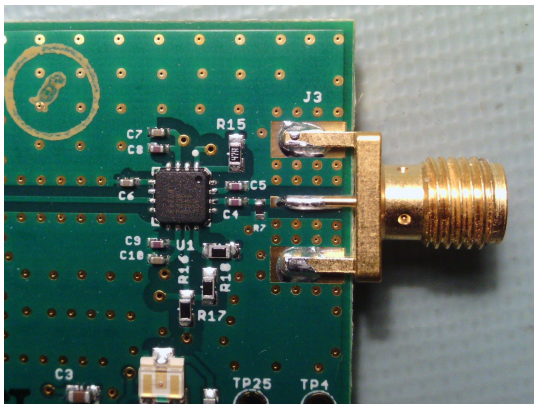

Figure 5.5 The ADL50902 IC on a demonstration board.

For rapid development, the National Instruments DAQ assistant was used to quickly configure and setup the USB-6009. Labview also includes blocks that allows us to easily record the data to a file and to use a low pass filter. The filter used is then configured to have a cutoff frequency that is equivalent to an integration time of 2 seconds to match the software defined radio. These blocks made up most of the program and resulted in a program that was quickly made. Figure 5.7 shows the blocks used and the wiring of the blocks.

5.2 Experiment II - Sensitivity and Stability Evaluation

In this experiment, we evaluate the sensitivity and stability of a SDR-based radiometer. This is evaluated both experimentally and statically.

5.2.1 Experimental setup

The experimental setup used for experiment two is the same as outlined in section 5.1.1. A longer soak time is used to verify the stability of the radiometer.

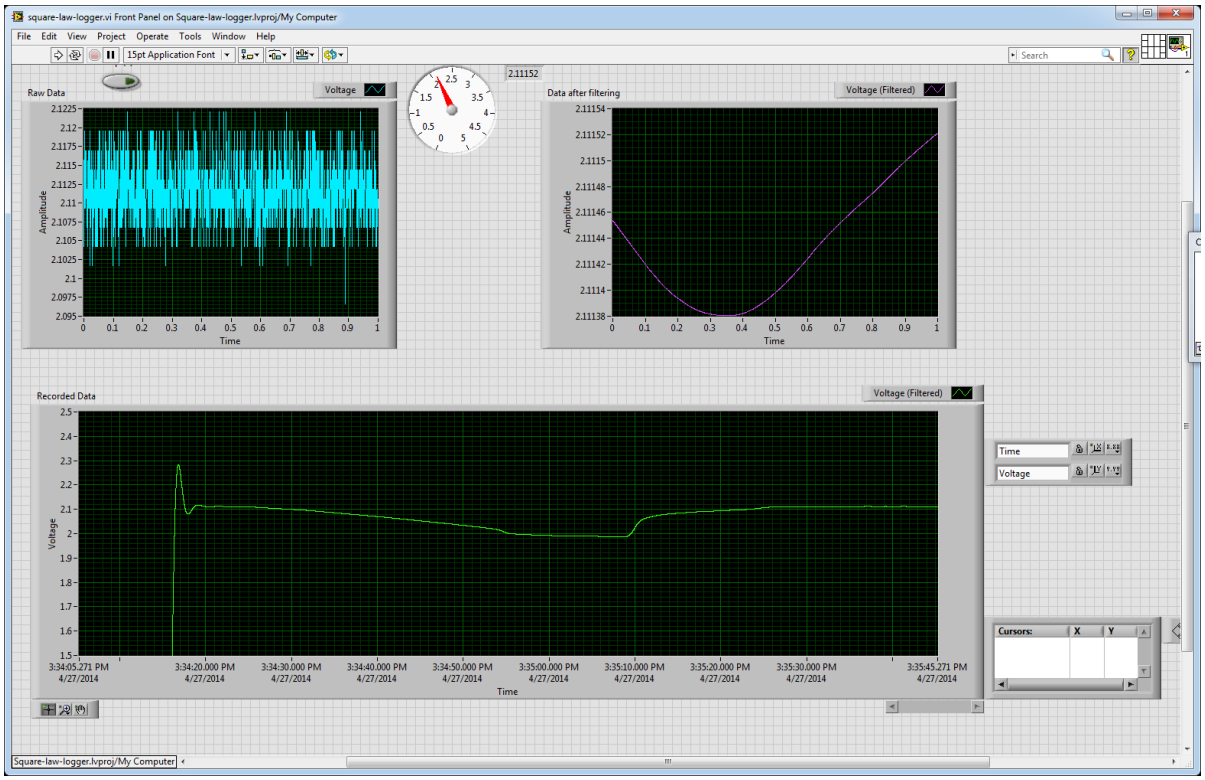

Figure 5.6 A screenshot of the Labview GUI interface.

5.2.2 Data collection

The data collected for this experiment were the total power measurements made from the SDR-based radiometer. These measurements uses the same method as outlined in section 5.1.2. A longer soak time was used for this experiment where the matched load was in LN2 for approximately five hours before the LN2 boiled out.

5.3 Experiment III - Interfering Signal Mitigation

In this experiment, we generate an interfering signal and then mitigate the signal using a software defined filter. A square-law detector is hooked up in parallel to measure the same signal, but is not provided with a mitigation mechanism. We then compare the two signals to verify that the SDR-based radiometer can mitigate the interfering signal,

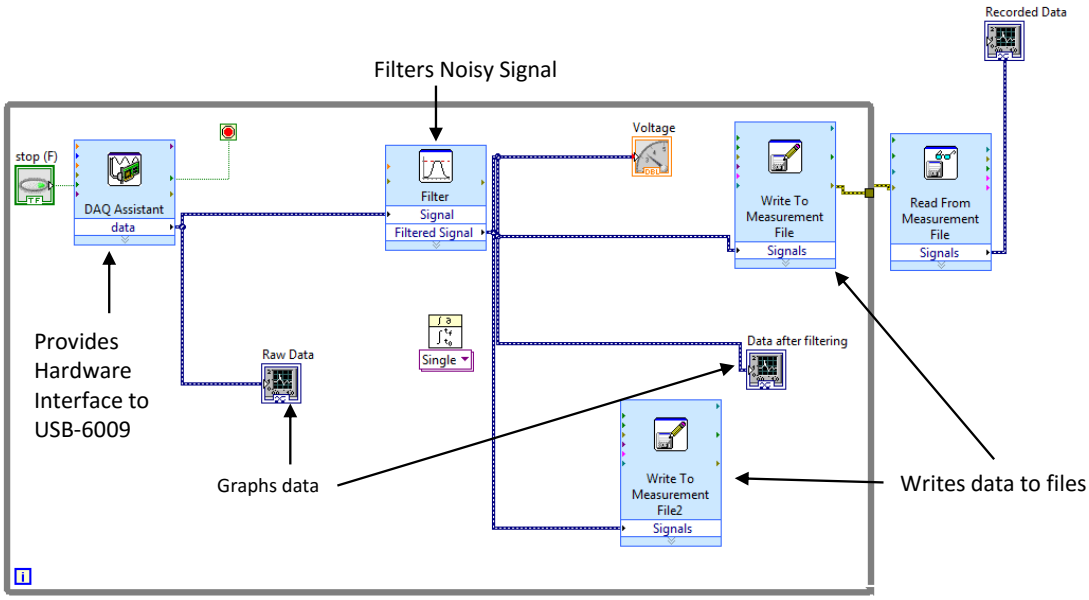

Figure 5.7 A screenshot of the Labview block diagram.

while continuing to make useful total power measurements.

This experiment was designed to determine whether or not a SDR-based radiometer can cope with an interfering signal. This test injects a known signal at 1.406 GHz to interfere with the normal operation of the radiometer. The amplitude of this signal is then incremented and decremented at various times during the test. This was done to reflect a possible real world scenario and to make it easy to identify the interfering signal with the square-law detector, which only measures power.

In order to mitigate the offending signal, a filter was designed to remove the offending signal. The design of the filter used a program that is part of the GNURadio software package, called the GNU Radio Filter Design tool. Figure 5.8 shows a screen shot of this tool when designing the band-reject filter for this application.

This tool generates filter values (also called taps) that GNURadio will use for defining the filter. The GUI program, shown in Figure 5.8, allows us to interactively create a filter. Because this tool is part of the GNURadio package, it also includes a command

line interface to the program. This allows us to call the program from within GNURadio to integrate this functionality into our SDR-based radiometer.

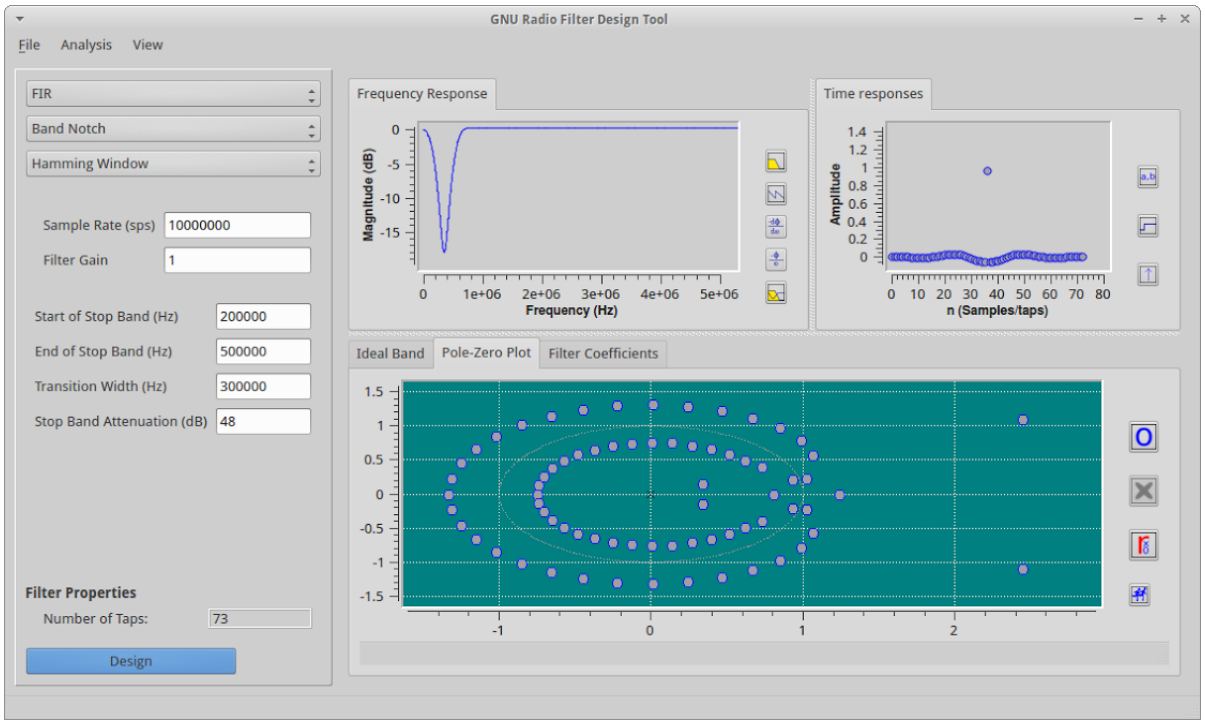

Figure 5.8 Image of the GNU Radio Filter Design tool

5.3.1 Experimental setup

The setup for this experiment is similar to the setup outline in section 5.1. In addition, a second software defined radio was added to inject an offending signal into the RF signal chain. This software defined radio was configured to operate as a signal generator to create the offending signal.

Our signal generator is the HackRF One (or just HackRF), shown in Figure 5.9. This SDR is cheaper, and has lower specifications than the Ettus Research N200 used as the SDR-based radiometer. However, it allows for transmission of a signal and it suits our needs for the purpose of a signal generator. The signal the HackRF generates is a

sinusoidal signal centered at 1.406 GHz. The amplitude of this sinusoidal signal is then adjusted accordingly.

As stated, the quality of this SDR is lower than the Ettus Research N200 SDR. As such the output signal is not as clean as one might expect. This can be seen in Figure 6.14 where two smaller signals are present. These are harmonics generated due to the Local Oscillator (LO) and the Intermediate Frequency (IF) used in the setup. However, this did not present an issue as the filter was designed to filter the main signal at 1.406 GHz and the two smaller harmonic peaks.

Figure 5.9 Image of the HackRF used to generate the offending signal. (Image from Great Scott Gadgets - www.greatscottgadgets.com)

The HackRF generates a sinusoidal wave at a fixed frequency of 1.406 GHz and can be seen in Figure 6.13. In this experiment, the signal amplitude is changed at different times during the experiment. Because the signal from this device is relatively large, 26 dB of attenuation is inserted between the HackRF and the first LNA to avoid over-powering the LNAs.

The amplitude is controlled from a program called *osmocom_siggen*. Osmocom was originally developed to communicate with OsmocomSDR hardware. However, it has been

expanded to include the HackRF and Ettus Research hardware. The *osmocom_siggen* program provides a GUI to set the frequency, amplitude and type of the signal generated.

5.3.2 Data collection

The data collected for this experiment includes both the total power measurements from the SDR-based radiometer and the square-law data. Section 5.1.2 explains in detail the setup and configuration of the equipment used to collect this data.

5.4 Experiment IV - Performance Impact of Interfering Signal Mitigation

In this experiment we examine the impact of filtering an interfering signal on the sensitivity of the SDR-based radiometer and how reducing our overall bandwidth affects the total power received by the radiometer.

5.4.1 Experimental setup

In this experiment, we set up our experiment as outlined in Section 5.3.

5.4.2 Data collection

The data collected for this experiment is the total power reading measurements obtained from the software defined radio. The data collection method is identical to the method used in Section 5.1.2.

CHAPTER 6. RESULTS AND ANALYSIS

This chapter presents the results obtained from the experiments outlined in Chapter 5. We discuss the analysis of the results and what was learned from each experiment. This chapter concludes with a trade off analysis between an SDR-based radiometer and a traditional radiometer in terms of cost and weight.

6.1 Experiment I - Software Defined Radio Based Radiometer Verification and Calibration

As outlined in Chapter 5, Section 5.1, this experiment verifies the operation of a software defined radio based radiometer. This is done by performing experiments that are similar to the verification and calibration methods used for a traditional radiometer. We compare our results to those of a square-law detector receiving the same signal.

6.1.1 Data collected

For this experiment, total power measurements were collected from the software defined radio based radiometer and the square-law detector. This data is then calibrated using the known physical temperature of the matched load. Table 6.1 shows the values collected during the experiment and is used to calibrate the raw total power measurements.

Table 6.1 Total Power calibration data points

rQ Value	X^2 Voltage (V)	Temperature (K)
.1139	1.9846	77
.1730	2.1065	271.65

6.1.2 Data analysis

To analyze the results, iPython Notebook is used to read our data and generate the graphs used in this thesis. This tool uses Python along with HTML and Markdown code to generate a virtual notebook for each experiment.

Software Defined Radio Data. The SDR records the total power measurements in a binary file that either Matlab or Python can read. This file format is explained in section 5.1.2. We begin by looking at the raw or uncalibrated total power readings. This power information is information collected after the total power radio block in GNURadio and is explained in Chapter 4. Because the values are uncalibrated total power readings, there are no units.

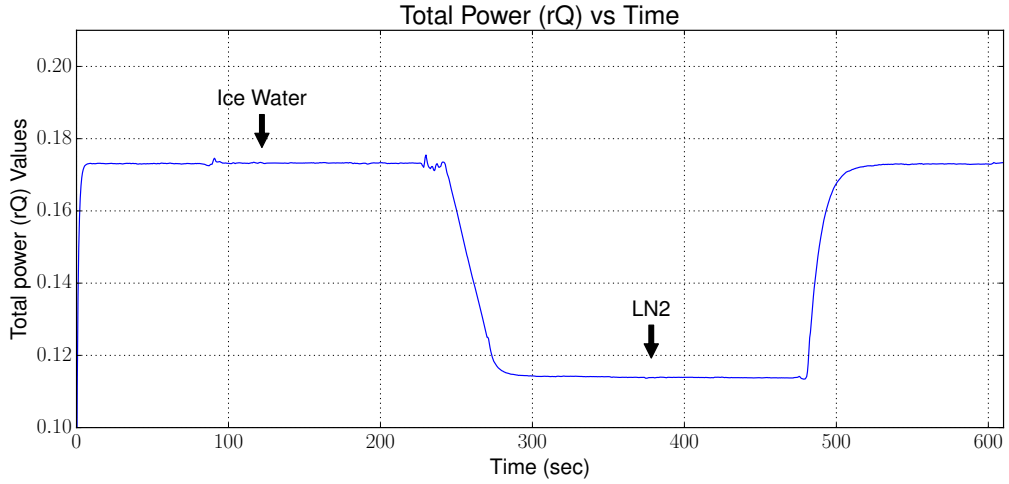

Figure 6.1 Graph of the uncalibrated SDR total power values of Experiment I

Figure 6.1 shows the total power reading versus time. Figure 6.1 has been annotated to show which medium the matched load has been placed in (i.e. Ice water and LN2).

Since we know the temperatures of the ice water and LN2, we can calibrate these readings to a noise temperature reading. This is done by reading a calibration file we have stored in a csv format and solving for the slope of a line. This was accomplished using the following code written in Python.

```
a=numpy.array([[rQ_val[0],1.0],[rQ_val[1],1.0]],numpy.float32)
b=numpy.array([temp_values[0],temp_values[1]])
z=numpy.linalg.solve(a,b)
```

Now that we have our calibration points, we can re-graph this data as calibrated noise temperature as shown in Figure 6.2. Figure 6.2 is also colorized to represent "warmer" to "cooler" noise temperatures.

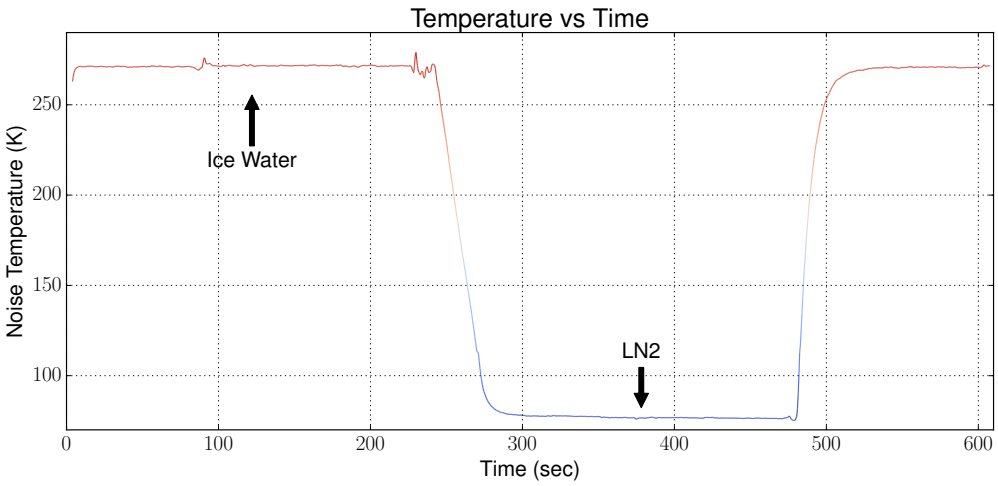

Figure 6.2 Graph of the SDR calibrated noise temperature for Experiment I

Square Law Detector Data. Now that we have looked at the software defined radio data, we want to examine the square-law detector data and then compare the two. The square-law detector gives us power information as a voltage. This voltage corresponds to the received power by the square-law detector

In order to compare the square-law detector to the SDR data we will also calibrate it as a noise temperature. We can do that using the same method as the SDR and calibrate

the voltages to the known temperature references. It can be see in Figure 6.3 that the data from the square-law detector is very noisy. Therefore, we use a filter to smooth out the data.

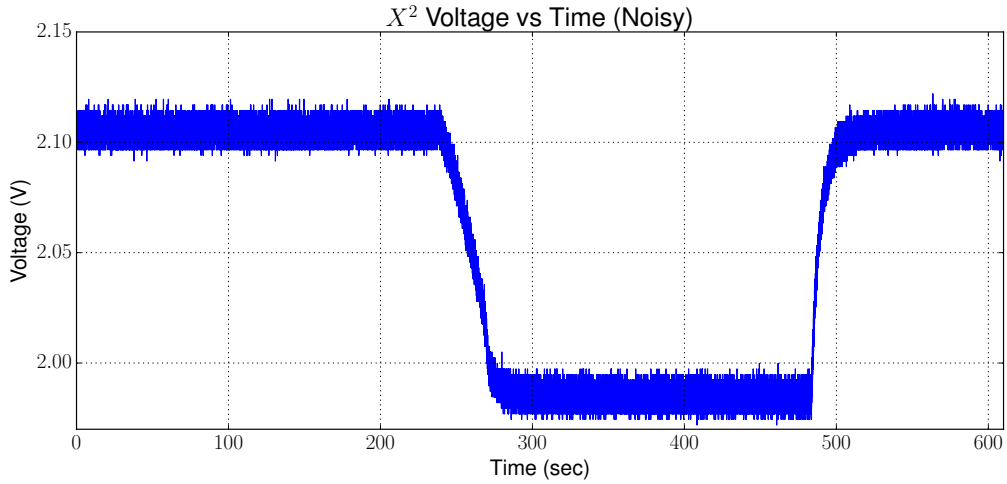

Figure 6.3 Unfiltered data from the square-law detector collected in Experiment I

To filter the data, Python in conjunction with SciPy is used. For this experiment, we use a low pass filter to smooth out the signal. The following code applies a Finite Impulse Response (FIR) low pass filter that filters the data.

```
from scipy import signal
N=100  # Number of taps
Fc=40  # Cutoff Frequency
Fs=1600  #Sample Frequency
h=scipy.signal.firwin(numtaps=N, cutoff=Fc, nyq=Fs/2)
x2_filt=scipy.signal.lfilter(h,1.0,x2_voltage)
```

Figure 6.4 shows our data after being filtered by the low pass filter. Using the same technique as earlier, we can now calibrate the raw voltages from the square-law detector to the noise temperature. As with thte data collected from the SDR-based radiometer, the data from the square-law detector is calibrated to the physical temperature that our

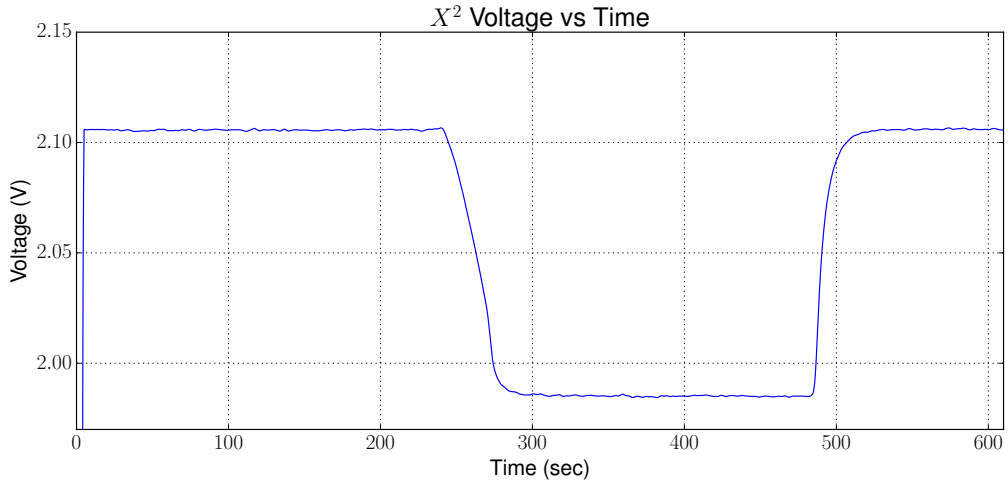

Figure 6.4 Filtered data from the square-law detector used in Experiment I

matched load is placed in. Figure 6.5 shows the calibrated data from the square-law detector. This now allows us to directly compare the square-law detector to the software defined radio data since we have a common reference point.

SDR-based radiometer vs square-law detector. We now compare the Software Defined Radio Based Radiometer with the square-law to make sure they match. Because both the SDR-Based Radiometer and the square-law are now calibrated to a noise temperature, we can graph both sets of data and compare them to each other.

We can see in Figure 6.6 that both the software defined radio and the square-law detector match up and this was analyzed to be within 0.002 Kelvin. This number was determined by taking the difference between the average values of the software defined radio and the square law detector over the same fifteen second period. This shows that both the square-law detector and the software defined radio agree when properly calibrated. This verifies that the software defined radio can indeed operate as a total power radiometer and the data we obtain from this setup agrees with an analog and more traditional radiometer.

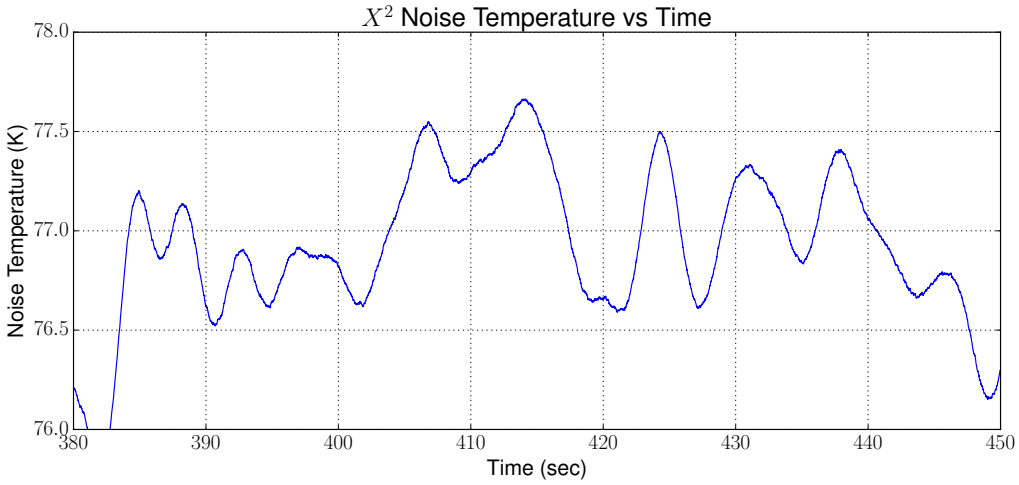

Figure 6.5 Calibrated data from the square-law detector used in Experiment I

6.2 Experiment II - Evaluation of Sensitivity and Stability

6.2.1 Data collected

This experiment examines the sensitivity and stability of a SDR-based radiometer. The data used for the sensitivity is data that was collected to confirm the stability of the radiometer. This data is the total power data collected from the SDR-based radiometer over a period of approximately five hours, however only a fifteen minute block was used for final analysis. For both sensitivity and stability, the data is calibrated using Table 6.2. A graph of the data showing the long duration soak time including indicating when the LN2 boiled out of the vessel is shown in 6.7.

Table 6.2 Total Power calibration data points for the stability experiment

rQ Value	Temperature (K)
.1132	77
.1770	271.65

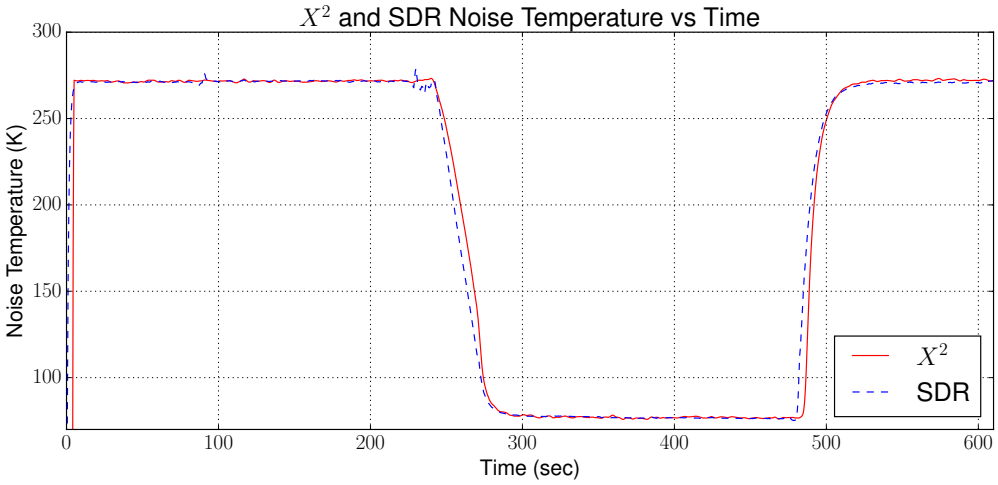

Figure 6.6 Figure showing both the SDR and square-law noise temperature data in Experiment I

6.2.2 Data analysis

Sensitivity. Sensitivity for a radiometer is the $NE\Delta T$ that was covered in chapter 3 and is found in Equation 3.6. The sensitivity can also be measured as the standard deviation of the data that we collect from the software defined radiometer.

Table 6.3 Experimental parameters for experiment one.

Bandwidth (β)	**Integration Time** (τ)	$T_A + T_N$
10 MHz	2 sec	462 K

Python is used to calculate the $NE\Delta T$ using the data from Table 6.3. Here T_A is the physical temperature of the matched load and T_N is the system noise temperature of the radiometer. The python code listed below is used to calculate the theoretical $NE\Delta T$.

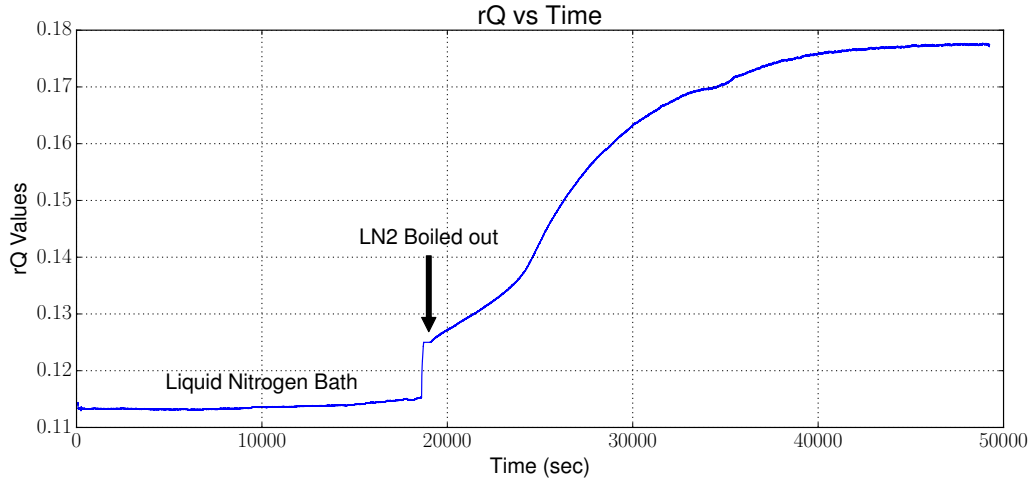

Figure 6.7 Graph of the un-calibrated total power from Experiment II while the matched load is submerged in LN2 between 4000 and 4900 seconds.

```
tau = 2 # Integration Time
BSDR = 10e6 # Bandwidth
TN = 385 # System noise temperature
TA = 77 # Antenna noise temperature
NEAT_SDR = (TA+TN)/sqrt(BSDR*tau)
```

This gives us 0.1 Kelvin for the expected $NE\Delta T$. Since the sensitivity is related to the standard deviation of the graph, we can use Python to give us the standard deviation of the data graphed in Figure 6.8. Figure 6.8 uses the data graphed in Figure 6.7 but zoomed in from 4000 to 4900 seconds.

Python is used to determine the standard deviation of this range of data. The code used is shown below and gives a result of .08 Kelvin. We can now plot both the expected sensitivity and the actual sensitivity which is shown in Figure 6.9.

```
stdsdr = numpy.std(g)
print stdsdr
```

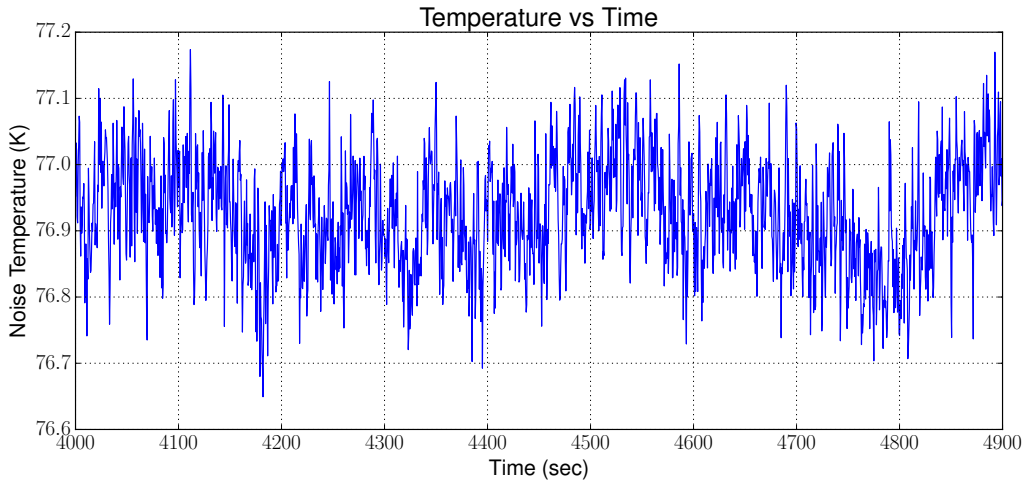

Figure 6.8 Graph of the calibrated total power from Experiment II while the matched load is submerged in LN2 between 4000 and 4900 seconds.

While .08 Kelvin is lower than the calculated sensitivity, it is still quite acceptable. As discussed in chapter 4 and shown in Table 4.1, our target $NE\Delta T$ is one Kelvin or less. Therefore our actual performance of .08 Kelvin still meets our radiometer requirement.

Stability. To verify stability of the radiometer, we look to see how much change the radiometer records over a relatively long period of time. To test this, a matched load was submerged in a liquid nitrogen bath for an extended period of time, in this case for fifteen minutes. The readings were then examined to study the trend of the data. The data is graphed in Figure 6.8.

We can use Python to calculate a second order polynomial to create a line to fit the data in Figure 6.8. An ideal line would be flat to show no overall change with the data. In Figure 6.10 we can see that the line and the data does not change more than 0.05 Kelvin over this fifteen minute period.

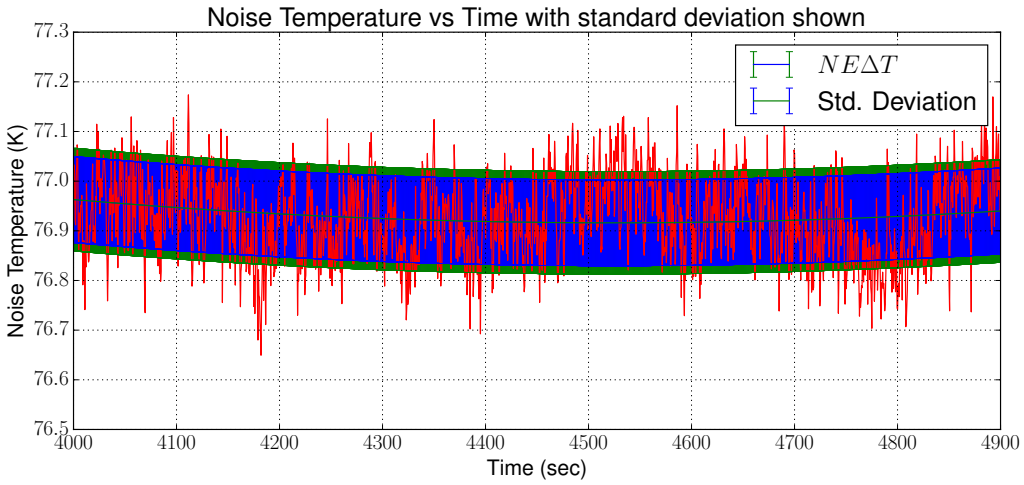

Figure 6.9 Graph of the calibrated total power with expected and actual sensitivity.

6.3 Experiment III - Interfering Signal Mitigation

The addition of an unwanted interfering signal has an adverse effect on how a radiometer operates[Ellingson et al. (2003)]. This is a growing problem for all radiometers, but is a greater issue for radiometers used in orbiting spacecraft to observe the Earth as they see large areas that could contain interfering signal sources [Misra et al. (2012)]. Even though the band we are working in (i.e. 1.4 GHz) is an internationally protected frequency, there have been both intentional and unintentional signal sources in this band detected by current space borne radiometers that have caused interference [Forte et al. (2013)].

6.3.1 Data collected

The data collected for Experiment III is the total power values from the SDR-based radiometer and the voltage data from the square-law detector. These values are calibrated using the data points provided in Table 6.4. These values differ from the previous tables because the filter is turned on. This changes the performance of the SDR-based

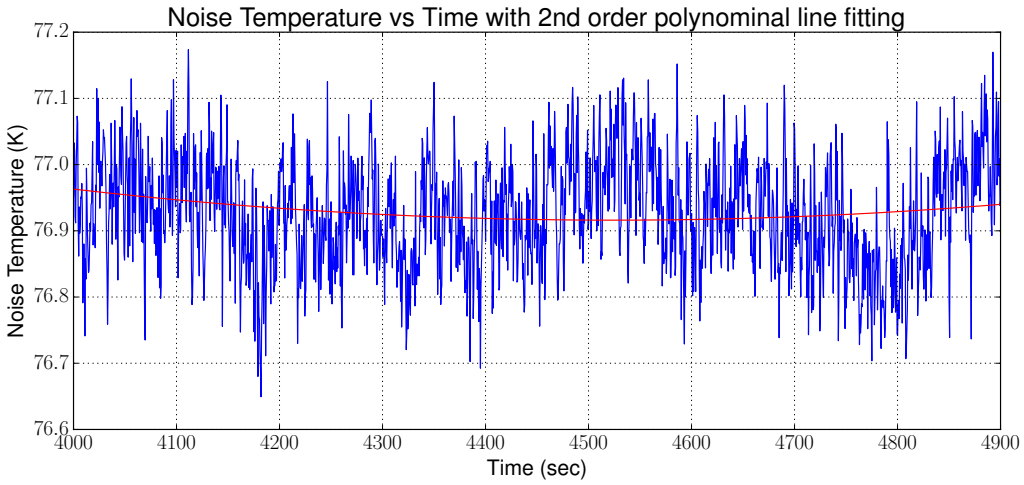

Figure 6.10 Graph of the calibrated total power over a period of 15 minutes with a best fitted line.

radiometer and results in different rQ values. The voltages are higher due to the presence of the offending signal.

Table 6.4 Total Power calibration data points

rQ Value	X^2 **Voltage (V)**	**Temperature (K)**
.0361	2.1234	77
.0623	2.1872	271.65

6.3.2 Data analysis

We begin by looking at what happens to our total power readings when no RFI mitigation is used. As stated in Section 5.3.1, the frequency of the offending signal will not change, but the amplitude will. This results in clear indications of the total power changing as the amplitude of the offending signal changes.

It can be seen that there are pulses that occur in Figure 6.11 that correspond to changes in the amplitude of the offending signal that affect our total power readings. These same pulses can be seen in the square-law detector data shown in Figure 6.12.

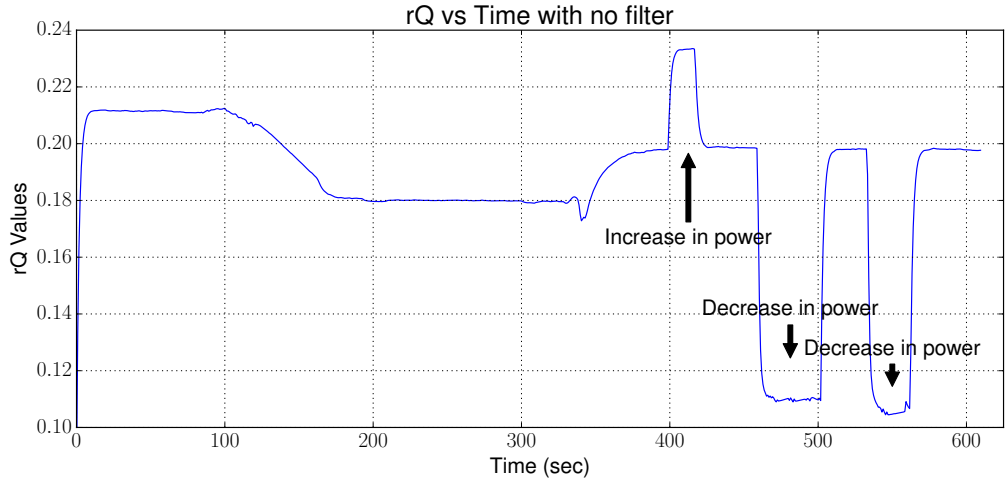

Figure 6.11 Graph showing the unfiltered total power measurements of the software defined radio

We can see in both the software defined radio and the square-law detector that there is an interfering signal. If we now look at the spectrum view of the software defined radio, shown in Figure 6.13, we can see the signal in question occurs at 1.406 GHz. The square-law detector has no frequency information, so our only method to detect an interfering signal is by looking at the total power readings. In Figure 6.12 we can see the spikes in the square-law data, however, we do not know where in the spectrum the offending signal is located.

Since we know where the offending signal is located for the SDR-based radiometer, we can design a filter to remove this signal. In GNURadio, we can specify both the frequency and the bandwidth that we desire for this band-reject filter. Ideally we want to keep the bandwidth of the filter as tight as possible to the offending signal, while making sure our filter is effective in removing the signal. Figure 6.14 shows the spectrum display of the software defined radio while filtering the offending signal.

Since we have now removed the offending signal, we want to re-run our experiment and once again compare the difference between the software defined radio and the square

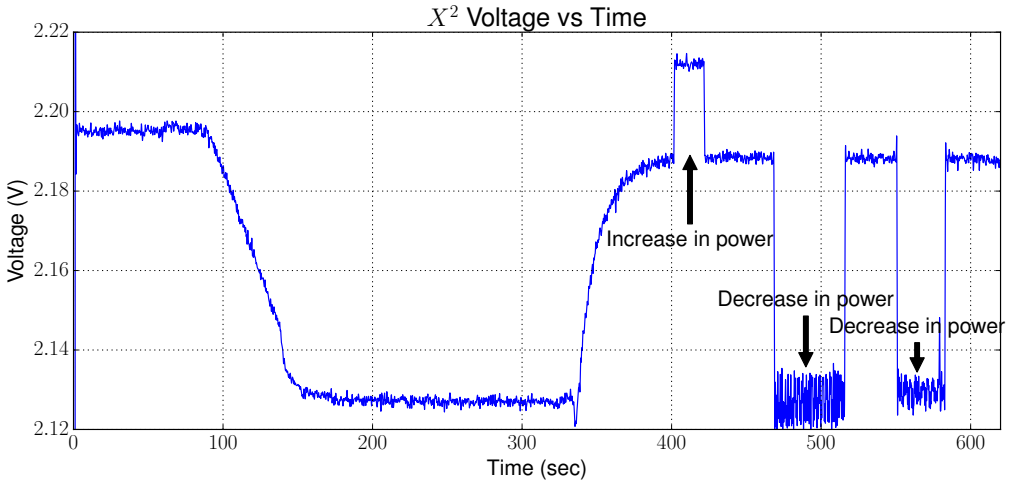

Figure 6.12 Graph showing the raw total power read from the square-law detector with an interfering signal.

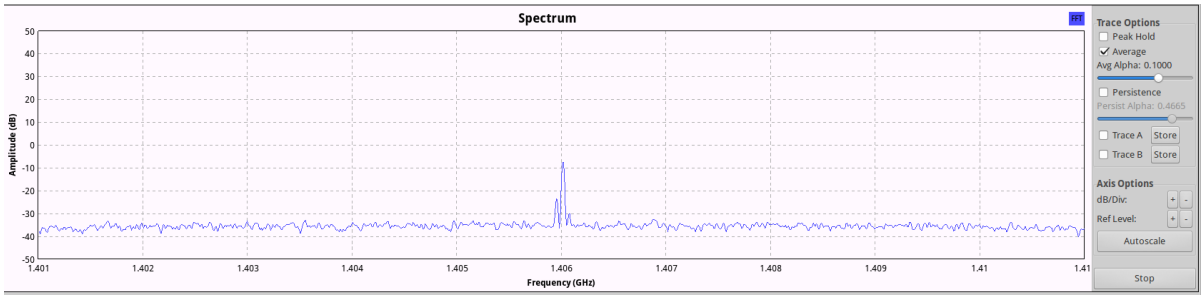

Figure 6.13 Image showing the spectrum view of the SDR-based radiometer with no RFI mitigation.

law detector. We can begin by looking at the software defined radio total power readings. Figure 6.15 shows a calibrated graph of the noise temperature seen by the software defined radio. This graph is similar to graphs we expect from a total power radiometer. However, we want to compare this to our square-law detector as well.

Figure 6.16 shows both the software defined radio and the square-law detector calibrated total power readings. In this graph you can see that the software defined radio is able to make normal readings where the square-law detector still shows changes in amplitude corresponding to offending signal pulses, which would make both calibration

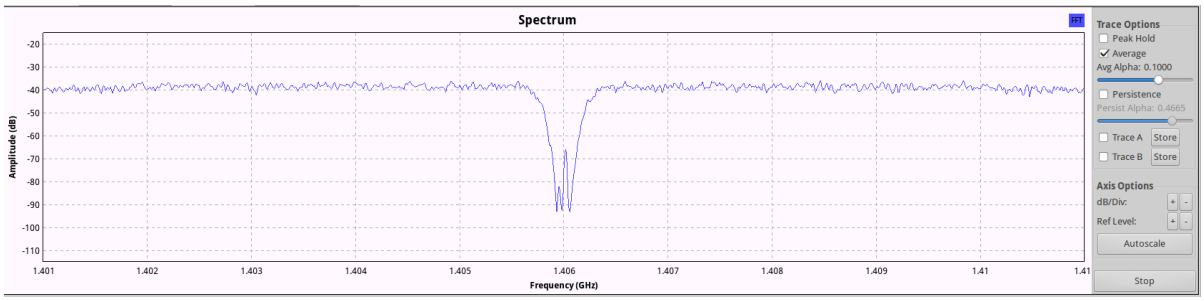

Figure 6.14 Image showing the spectrum view from the SDR-based radiometer filtering the offending signal

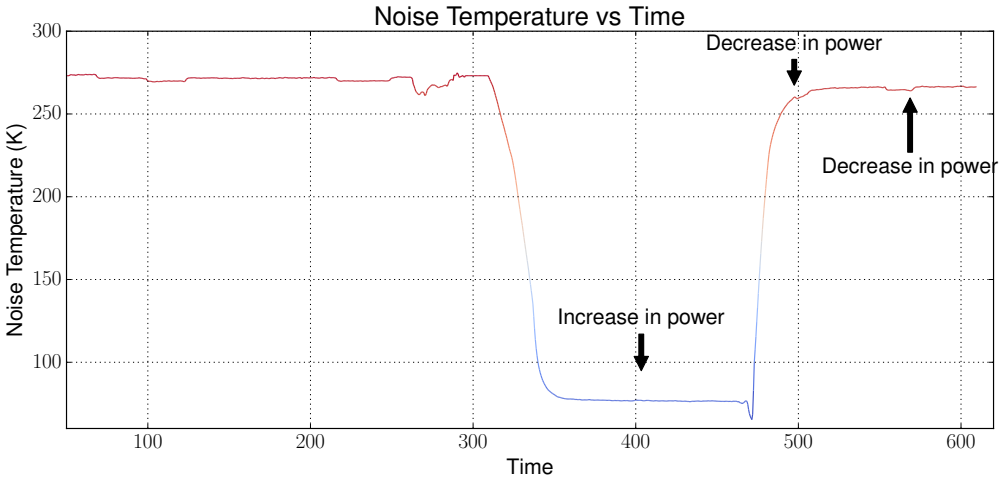

Figure 6.15 Graph showing the calibrated total power readings with the filter removing the offending signal

and obtaining useful data difficult.

Filter delay. It should be noted that when doing data analysis, that there will be some delay due to two factors. First, there is a delay in the software defined radiometer due to the integrator that is used. This creates a time delay as the integrator accumulates information and then settles. We use fairly large integration times, usually in seconds so this can add a significant delay. Second we do have a smaller delay in the decimation and low pass filter also used in the software defined radiometer. These are Finite Impulse Response (FIR) filters and thus have a delay given in Equation 6.1, where N is the

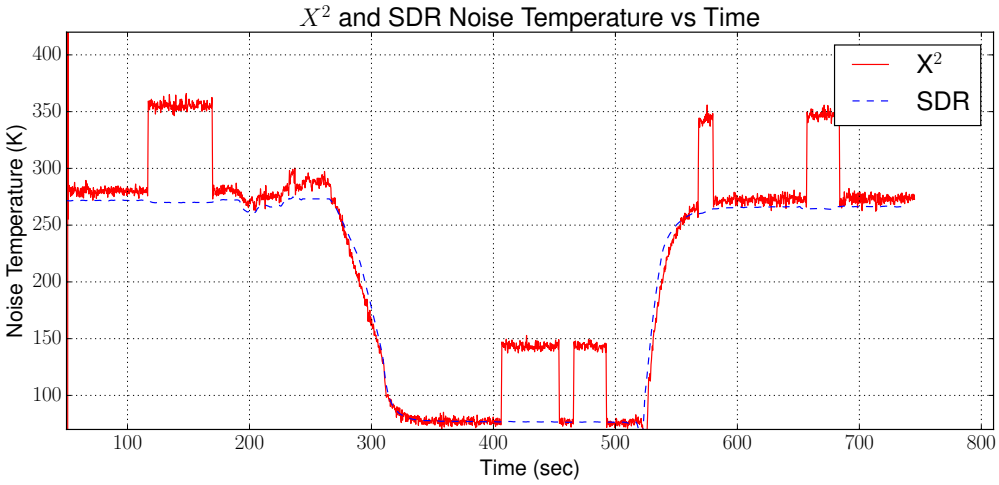

Figure 6.16 Image of the offending signal being filtered out by the SDR. It can be seen that the signal is no longer visible.

number of taps generated and F_s is our sampling frequency, in our case 10 MHz.

$$\frac{(N-1)}{(2*F_s)} \tag{6.1}$$

The taps value is generated by Python using the filter design program. For this experiment, the number of taps generated was 18,181. Taking this and our sampling rate into account, our FIR filter only delays the signal by 9 milliseconds.

A final note on aligning the square-law detector and the software defined radiometer data. Both systems have a record function and must be started manually by the user. They also run on separate computers. Therefore, there is a human error that also gets introduced to the system as well. This is usually no more than 1 or 2 seconds. But in this experiment it was noted that there was a slightly longer delay between the two of about 4 seconds.

6.4 Experiment IV - Performance Impact of Interfering Signal Mitigation

The goal of this experiment is to examine what affect adding a filter has on the performance of the radiometer. While we have demonstrated we can filter an offending signal, this comes at a cost. This experiment examines the impact that filtering out bands has on the measurement of total power and radiometer sensitivity.

6.4.1 Data collected

The data collected for this experiment is shown Table 6.5 and Table 6.6, and the remaining data is generated from Equations 6.2 and 6.3. This data will be examined next.

Table 6.5 Measured sensitivity and Bandwidth of Filter

NEΔ T (K)	**Bandwidth (MHz)**
.139	.125
.141	.250
.143	.500
.147	1
.153	2
.166	3
.181	4
.195	5
.234	6
.252	7
.318	8
.450	9
1.45	10

6.4.2 Data analysis

In Experiment IV, we examine the performance of a software defined radiometer when a filter is used. Recalling the equation for $NE\Delta T$ from Equation 3.6, our sensitivity

Table 6.6 Measured Total Power and Bandwidth of Signal

Total Power Value (rQ)	**Bandwidth (MHz)**
.003	.125
.006	.250
.012	.500
.025	1
.050	2
.071	3
.101	4
.125	5
.140	6
.170	7
.200	8
.230	9
.250	10

is a function of the amount of noise from both the antenna (T_A) and the addition of system noise (T_N). Sensitivity is also a function of the radiometer's bandwidth (β) and integration time (τ).

Our integration time is controllable, and can be set using the GUI panel of the software defined radio. In a typical radiometer, we often do not have any control of the bandwidth. It is often set by the mechanical band-pass filters that are in place and the square-law detector circuit to measure the noise power. In a SDR radio, we have more control over bandwidth as we can change our sampling rate which in turns controls our bandwidth. There is a limit as larger sampling rates require higher performance ADCs and greater computing speed.

Recall from Experiment III, in section 6.3, where we filtered out an offending signal. While we were able to filter out the offending signal and resume total power measurements, it comes at the cost of reducing the overall bandwidth available for power detection. In this experiment, we examine how this impacts sensitivity based on Equation 6.2.

$$NE\Delta T = \frac{T_A + T_N}{\sqrt{(\beta - \beta_{filter})\tau}} \tag{6.2}$$

Because we are notching out a portion of the bandwidth in order to remove the offending signal, this also removes that bandwidth for total power detection. As the bandwidth of the filter (β_{filter}) increases, the more bandwidth that is subtracted from the total bandwidth (β) available. Equation 6.2 accounts for this loss by subtracting the filter bandwidth (β_{filter}) from the total bandwidth (β).

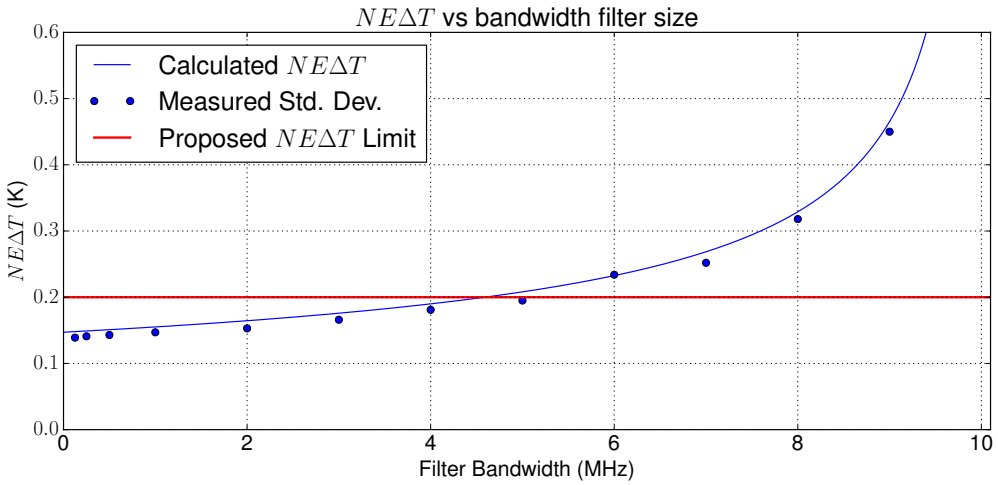

Figure 6.17 Graph of the calculated $NE\Delta T$, the proposed limit for the sensitivity of the radiometer, and the measured standard deviation with repsect to filter bandwidth.

We can graph the expected response of the $NE\Delta T$ by adjusting the values of β_{fitler} to range from a narrow-band filter, in our example 125 kHz, all the way to 9.99 MHz or nearly all of the bandwidth. Figure 6.17 shows the expected exponential response of $NE\Delta T$ as the size of the filter increases.

Figure 6.17 shows measured standard deviation points for the SDR-based radiometer for different filter sizes. These filter sizes and collected data are found in Table 6.5. It can be seen in Figure 6.17 that there is a good correlation between the expected sensitivity of the radiometer and the measured sensitivity of the radiometer.

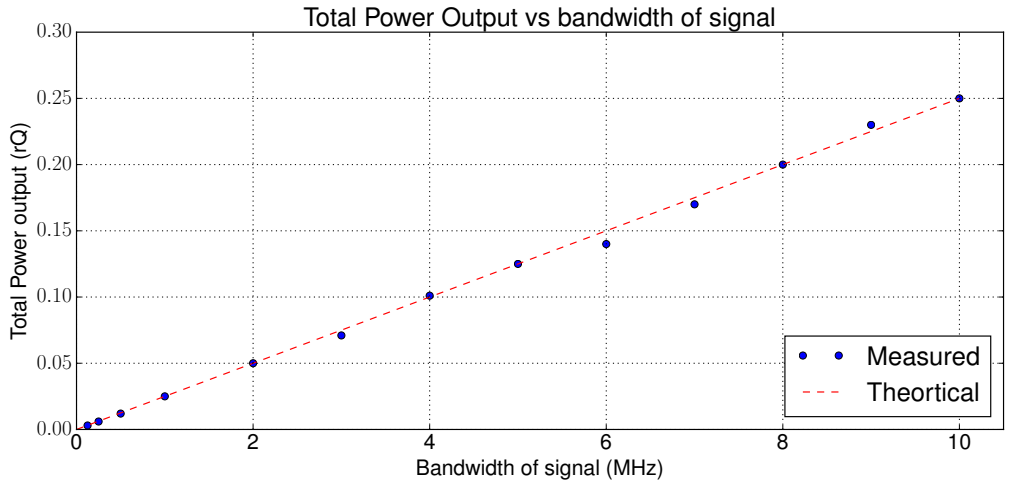

Figure 6.18 Graph of the total power measured and theoretical power versus the bandwidth of the measured signal.

Finally a line is added to Figure 6.17 to show a possible limit of when we may have filtered too much. In this example a $NE\Delta T$ of 0.2 Kelvin is used. In Figure 6.17 it can be seen that at about 5 MHz, our $NE\Delta T$ exceeds our threshold of 0.2 Kelvin. This would mean that to meet this performance criteria we would need to not exceed 5 MHz for our filter size. This is assuming our integration time (τ) and the bandwidth (β) is held constant.

We now look at the relationship of the total power received as the bandwidth decreases. Figure 6.18 shows both the measured total power received and the expected total power received as the bandwidth of the filter increases.

The total power is calculated from Equation 3.2 and is based on the system noise (T_N), the antenna (T_A), the bandwidth (β) and the power gain (G) of the amplifiers used. Our power gain and noise temperatures are fixed, and in this experiment we use a system gain of 30 dB. This represents the power gain that we see with the three LNAs used minus any losses or attenuation placed in the RF chain.

The change in this experiment is the amount of bandwidth seen by the SDR-based radiometer. Again, we can modify Equation 3.2 by subtracting the filter bandwidth (β_{filter}) from the total bandwidth available and this is shown in Equation 6.3.

$$P_{out} = k(\beta - \beta_{filter})G(T_A + T_N) \tag{6.3}$$

To compare this theoretical power to the actual power measured, we collected the rQ values by once again creating different filter sizes and then measuring the rQ values. These values can be found in Table 6.6 and are added as dots to Figure 6.18. This also shows a very good correlation between the expected total power received and the measured total power received for Experiment IV.

6.4.3 Usage scenario

Application with Soil Moisture Readings. A common application of radiometers is in the measurement of soil moisture. All materials naturally emit RF energy due to the random excitation by the electrons in the object. The amount of noise that gets generated varies by temperature and the amount that reaches the antenna varies by the amount of moisture in the soil. If we calibrate the radiometer to two known soil conditions, then we can measure the various levels of soil moisture in the soil. We will assume the soil is at a constant temperature during the observation. We will look at the percentage of moisture in the soil, which will vary from zero percent or dry soil to one hundred percent or very wet soil. The drier the soil, the more thermal noise we receive and the "warmer" the noise temperature. Wet soil on the other hand attenuates the thermal noise and shows up as a "cooler" noise temperature.

Using the SDR-based radiometer, we can configure it to observe a soil sample area. Since we do not have an antenna hooked up, we will simulate this by using a matched

load submerged in two reference temperatures. We will use the ice water bath and LN2 bath that was used in previous experiments for our reference temperatures.

Figure 6.19 shows the data collected from Experiment I. If we assume that our LN2 is dry soil and that our ice water bath is wet soil, we can now interpolate the data to this scale and show the information we obtained in Experiment one as both a noise temperature and soil moisture.

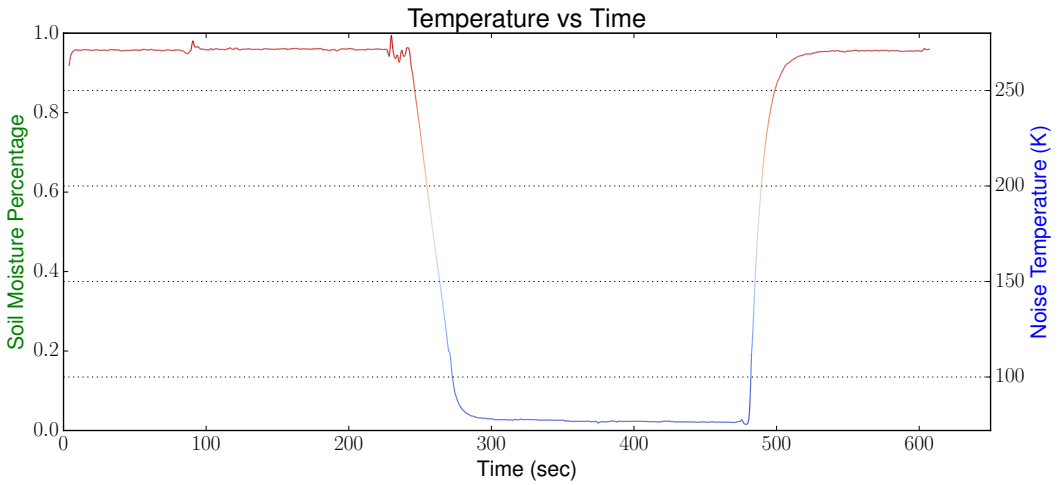

Figure 6.19 Plot of the noise temperature of Experiment I with soil moisture percentage added.

While this demonstrates that we can calibrate our total power readings with a soil moisture percentage, we would use actual field tests to calibrate the radiometer. In addition, we could also calibrate to soil moisture content instead of a percentage if desired. Both methods have been done with traditional radiometers[Jonard et al. (2011)][Shi et al. (2003)].

Mitigating interference. Experiment III demonstrates that an offending signal will impact and cause the total power received to increase. This increase in total power will then invalidate our calibration of the SDR-based radiometer. This will result in a spike as shown in Figure 6.11. This spike shows a higher than normal total power which will

result in a higher noise temperature. A higher noise temperature will then cause the SDR-based radiometer to show drier soil conditions than what is actually present.

By mitigating the interfering signal the SDR-based radiometer is able to make calibrated soil moisture measurements even with the signal present. Figure 6.15 demonstrates the SDR-based radiometer making total power measurements with an offending signal present.

Soil moisture measurements can continue with an offending signal present, however there is an impact on the system when filtering out this signal. This is covered in Section 6.4. Because of this impact, re-calibration of the SDR-based radiometer still needs to take place.

Example experiment. An example of how a SDR-based radiometer would mitigate a signal for a soil moisture reading is outlined next. The SDR-based radiometer is configured to record both total power measurements and the in-phase and quadrature-phase (i.e. I and Q) data. The recorded data is then analyzed after the experiment is over and it is observed that the total power readings have erratic readings. Displaying the frequency and magnitude information that is generated from the I/Q data shows an interfering signal is present. A band-reject filter is then designed to notch the offending signal out from the data presented. The experiment can now be re-played with the filter active and new total power measurements are now recorded with out the offending signal.

As discussed in Section 6.4, the performance (i.e. sensitivity) of the radiometer will change depending on the bandwidth of the filter used for the signal. Figure 6.18 however, shows that the relationship between the bandwidth of the signal and the total power received is a linear relationship and can therefore be predicted. We can therefore compensate for the filter to maintain the calibration of the radiometer.

6.5 Advantages of a Software Defined Radio Based Radiometer

A study was conducted on what benefits a software defined radio based radiometer would have over a more traditional radiometer. We focused on three main areas; cost, weight and size, and the value a SDR-based radiometer can add over traditional radiometers.

6.5.1 Cost benefits

Software defined radios have become more commonplace in recent years and this has generated a number of Commercial Off The Shelf (COTS) solutions. A COTS solution is often a lower cost solution due to the mass manufacturing that takes place. This has driven the cost of many SDRs to under one thousand dollars, while still having excellent performance characteristics. The N200 SDR purchased for this research cost fifteen hundred dollars and the daughter-board approximately cost one hundred and fifty dollars. Other software defined radios however have come out on the market since then. Ettus, for example, has some that are below one thousand dollars and the author has also obtained the HackRF One SDR that now sells for three hundred dollars. The main difference between software defined radios on the market is in their resolution and the bandwidth they support.

As seen in Table 6.7, even the higher cost Ettus research equipment is a lower cost option than the custom built radiometer in use at Iowa State University, and even a comparable off the shelf radiometer. It should be noted that the radiometer in use at Iowa State University is also a dual polarization radiometer so there are two RF front ends and two ADCs that feed into a FPGA board. It would be quite easy to add dual polarization to the Ettus N200 SDR as it does support two daughter-boards. This would increase the cost to $2,179 for the additional LNAs and daughter-board.

Table 6.7 Cost Analysis

Device	Quantity	Cost
SDR Solution		
N200 SDR	1	$1515
LNA at $60 ea.	3	$180
DBSRX2 Daughter-board	1	$152
GNURadio	1	$0
Total		$1847
ISU Radiometer		
LNA, FPGA, ADC, Microcontroller and power supplies	1	$10,000
Commercial Off the Shelf Unit		
Spectracyber 1420 MHz Hydrogen Line Spectrometer	1	$2,650

The largest cost benefit is that key components that you find in a radiometer, the filters and square-law detector, can be performed in software instead of needing additional equipment. The system is also much more frequency agile, which means it can work over a broader range of frequencies than most traditional radiometers, with very little change in hardware and in some cases may require no change in hardware. The Ettus N200, for example, uses daughter-boards to provide the RF interface. While these boards provide a high quality RF signal, it does come at a cost and are usually designed for certain bands of frequencies. Other low cost SDRs, however, also support a wide range of frequencies. The HackRF, for example, works from 10 MHz to 6 GHz, but does so at the cost of lower resolution, less power gain in its front-end, and supports a lower bandwidth.

6.5.2 Weight and component size benefits

A typical radiometer has many components that are involved in its design. This includes filters, LNAs, and the power detection (i.e. square-law detector) used. These components add weight, size and costs to a radiometer. A software defined radio however digitizes the signal and we are able to replace the filters and square-law detector with their software equivalent. While a software defined radio does add an ADC and usually a FPGA to perform signal processing, advances in semiconductor technology have con-

tinued to shrink these components. These components are also lighter than the filters often used in traditional radiometers. Table 6.8 displays the mass for the SDR-based radiometer built and the ISU Radiometer. The ISU Radiometer is a representation of most traditional radiometers.

Table 6.8 Weight Analysis

Device	**Mass**
SDR Solution	
N200 SDR	1.2 kg
LNA at .03 kg ea	.09 kg
DBSRX2 Daughter-board	.1 kg
Total	1.39 kg
ISU Radiometer	
LNA, FPGA, ADC, Microcontroller and power supplies	22.7 kg
Commercial Off the Shelf Unit	
Spectracyber 1420 MHz Spectrometer (estimated)	6 kg

Size is another benefit, since semiconductor technology has continued to shrink components. Again, since items like the filters and square-law detector are replaced with software implementations, this helps to reduce the overall size.

6.5.3 Value added benefits

A SDR-based radiometer adds additional value for two reasons. One, it is able to work with both frequency and magnitude information, where a traditional radiometer does not. This allows for additional analysis on the signal and can help identify issues such as an interfering signal, which was demonstrated in this thesis.

Second, we are able to have an agile system that is able to adapt to changing conditions with very little or no change to hardware. Different types of radiometers can be implemented such as a Dicke radiometer, dual polarization radiometer, or a radiometer that can perform Stokes parameters. It is also possible to implement new types of radiometers that have not been developed yet. Finally, since we have both frequency and

power information, we can create a system that is able to adapt to changing conditions, such as dealing with an interfering signal.

6.6 Drawbacks of a SDR-based Radiometer

Although we have outlined a number of advantages of using a COTS SDR-based Radiometer and how a SDR can add additional value to the radiometer system, there are some disadvantages to a SDR-based Radiometer.

6.6.1 Power consumption

One of the largest drawbacks to a SDR-based radiometer can be in the power consumption of the SDR. With the move to perform functions such as power detection and filtering we now require additional computational power to perform these tasks. With those computational cycles additional power is now required. The use of FPGAs and SoC can help to minimize these power concerns as they are more efficient than using a PC.

Power and CPU requirements also increase as we add additional functionality such as filtering an offending signal. While these additions may not require additional hardware, it can require additional processor and memory requirements. An increase and processing requirements will also increase power requirements and can also add additional requirements in cooling.

6.6.2 Bandwidth constraints

While SDR technology has advanced, bandwidth is still a constraint that affects SDRs and in turn a SDR-based Radiometer. Bandwidth plays a critical role in a radiometer's sensitivity as explained in this thesis, therefore the fact that many SDRs are limited in bandwidth does create a disadvantage. In many cases, this bottleneck takes place in

both the transport and processing of large bandwidth systems. This also relates to the power consumption disadvantage since larger bandwidth also means requiring additional computational cycles.

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this thesis we have shown that a SDR-based radiometer, using off the shelf components, is able to perform comparable to a traditional radiometer. Additionally, it was demonstrated that our SDR-based radiometer could be easily extended to mitigate radio frequency interference.

7.2 Future Work

Two possible future work items are: 1) moving the signal processing to a dedicated processor, and 2) building more complex radiometer systems (e.g. Dicke radiometer). These topics will be covered in the following paragraphs.

Removing the PC. For this thesis, we used software that would run on a PC or comparable computer system running a full operating system such as Linux. This allowed for rapid development by using software tools such as GNURadio to develop the SDR-based radiometer. While this is a great platform for prototyping a SDR-based radiometer, it does require hardware that is capable of running a full operating system and the associated software. Some radiometer applications would find this acceptable, however, other remote sensing applications (e.g. space satellite) would require a more efficient configuration. One solution is to move the signal processing from the PC to a Field Programmable Gate Array (FPGA) or an Application-Specific Integrated Circuit (ASIC) to improve efficiency.

Implementing different types of radiometers. This thesis focused on implementing a simple total power radiometer in software. While this radiometer is effective, it relies on the fact that the components of the radiometer are stable. Other types of radiometers have been developed that reduce this need.

A common method is a Dicke radiometer, which was covered in this thesis. A future work for our SDR-based radiometer would be to use a digitally generated noise source, such as a Gaussian noise source, and then switch between the antenna and this known noise source. This noise source could also be adjusted in software, therefore stability of the noise source would not be an issue.

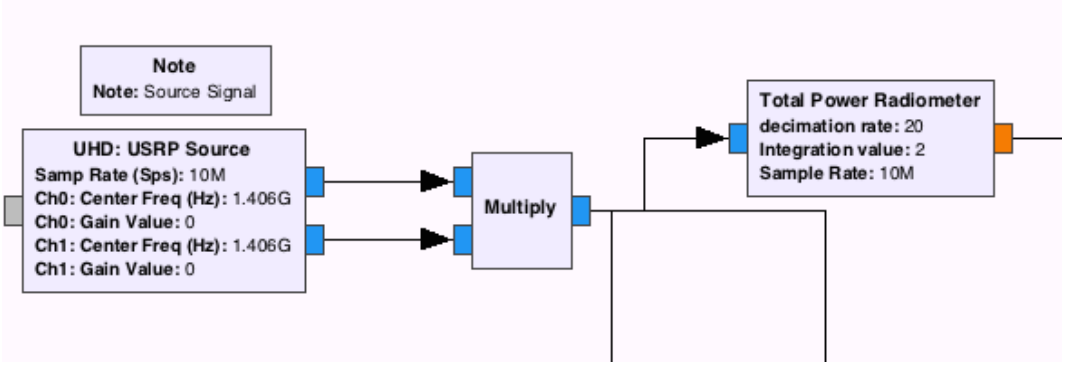

Figure 7.1 A screenshot of an implementation of a correlating radiometer in GNURadio Companion.

Another method to improve stability and sensitivity is to correlate against a second input. The second input could be another antenna looking at the same source, or could be two polarizations from the same antenna[Clapp and Maxwell (1967)]. This results in a two receiver system looking at the same source with two signal outputs, S_1 and S_2. Since we are looking at the same source, both signals will be correlated in time, and when multiplied they will provide an output proportional to the strength of the source signal. The noise introduced by each receiver will then have a lower correlation due to the random nature of the noise. This results in a radiometer with greater sensitivity due to the reduction of the noise, even though two receivers are used [Fujimoto (1964)].

The N200 software defined radio was chosen as it is capable of having two different daughter-cards plugged in. Therefore, it is possible to have both sources enter the software defined radio and once digitized we can sum the magnitudes of the two incoming sources. This is easy to do and is shown in Figure 7.1. Although Figure 7.1 shows a correlating SDR-based radiometer, it has not been tested. In theory, this should correlate the signal and improve the sensitivity of the radiometer. Additional experimentation would be required to confirm that this implementation operates as expected.

BIBLIOGRAPHY

Behnke, P., Soberal, D., Bredeweg, S., Dunne, B., Sterian, A., and Furton, D. (2013). Senior capstone: A software defined radio design for amateur astronomy. In *Interdisciplinary Engineering Design Education Conference (IEDEC), 2013 3rd*, pages 104–111.

Clapp, R. and Maxwell, J. (1967). Complex-correlation radiometer. *Antennas and Propagation, IEEE Transactions on*, 15(2):286–290.

De Roo, R., Ruf, C., and Sabet, K. (2007). An l-band radio frequency interference (rfi) detection and mitigation testbed for microwave radiometry. In *Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International*, pages 2718–2721.

Department, A. (2013). *Electronic Warfare and Radar Systems Engineering Handbook*. Naval Air Warfare Center Weapons Devision, Point Mugu.

Dicke, R. H. (1946). The measurement of thermal radiation at microwave frequencies. *Review of Scientific Instruments*, 17(7):268–275.

Ellingson, S., Hampson, G., and Johnson, J. (2003). Design of an l-band microwave radiometer with active mitigation of interference. In *Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International*, volume 3, pages 1751–1753.

Erbas, C., Hornbuckle, B., and De Roo, R. (2006). Iowa state university/the university of michigan direct sampling digital radiometer. In *Geoscience and Remote Sensing*

Symposium, 2006. IGARSS 2006. IEEE International Conference on, pages 3074–3077.

Evans, G. and McLeish, C. W. (1977). *RF Radiometer Handbook*. Artech House, Dedham, MA.

Fischman, M. A. (2001). *Development of a direct-sampling digital correlation radiometer for earth remote sensing applications*. PhD thesis, University of Michigan.

Forte, G., Tarongi Bauza, J., dePau, V., Vall llossera, M., and Camps, A. (2013). Experimental study on the performance of rfi detection algorithms in microwave radiometry: Toward an optimum combined test. *Geoscience and Remote Sensing, IEEE Transactions on*, 51(10):4936–4944.

Fujimoto, K. (1964). On the correlation radiometer technique. *Microwave Theory and Techniques, IEEE Transactions on*, 12(2):203–212.

Hardy, W. N., Gray, K., and Love, A. (1974). An s-band radiometer design with high absolute precision. *Microwave Theory and Techniques, IEEE Transactions on*, 22(4):382–390.

Jonard, F., Weihermuller, L., Jadoon, K., Schwank, M., Vereecken, H., and Lambot, S. (2011). Mapping field-scale soil moisture with l-band radiometer and ground-penetrating radar over bare soil. *Geoscience and Remote Sensing, IEEE Transactions on*, 49(8):2863–2875.

Jondral, F. K. (2005). Software-defined radio: basics and evolution to cognitive radio. *EURASIP journal on wireless communications and networking*, 2005(3):275–283.

Kerr, Y. (2012). Smos rfi detection: Today's maps. Website. http://www.cesbio.ups-tlse.fr/SMOS_blog/?p=2963.

Leech, M. (2006). Gnuradio and usrp: Solderless breadboarding for the 21st century. In *25th Anniversary Conference of the Society of Amateur Radio Astronomers*.

Leech, M. and Ocame, D. (2007). A year of gnu radio and sdr astrononomy: experience, practice and observations. Website. http://www.sbrac.org/documents/gnuradio_at_one_year_20070401.doc.

Leinweber, G. (2001). Square law diode detectors in 50 ohm systems.

Liu, P.-W., Judge, J., DeRoo, R., England, A., and Luke, A. (2013). Utilizing complementarity of active/passive microwave observations at l-band for soil moisture studies in sandy soils. In *Geoscience and Remote Sensing Symposium (IGARSS), 2013 IEEE International*, pages 743–746.

McMullan, K. D., Brown, M., Martin-Neira, M., Rits, W., Ekholm, S., Marti, J., and Lemanczyk, J. (2008). Smos: The payload. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(3):594–605.

Misra, S., De Roo, R., and Ruf, C. (2012). An improved radio frequency interference model: Reevaluation of the kurtosis detection algorithm performance under central-limit conditions. *Geoscience and Remote Sensing, IEEE Transactions on*, 50(11):4565–4574.

Mitola, J. (1995). The software radio architecture. *Communications Magazine, IEEE*, 33(5):26–38.

Nyquist, H. (1928). Thermal agitation of electric charge in conductors. *Physical review*, 32(1):110–113.

Rashid, R. A., Sarijari, M. A., Fisal, N., Yusof, S. K. S., and Mahalin, N. H. (2011). Spectrum sensing measurement using gnu radio and usrp software radio platform. In

ICWMC 2011: The Seventh International Conference on Wireless and Mobile Communications, pages 237–242.

Richaume, P. (2012). Outburst of anger. Website. http://www.cesbio.ups-tlse.fr/SMOS_blog/?p=3381.

Ruf, C. and Gross, S. (2010). Digital radiometers for earth science. In *Microwave Symposium Digest (MTT), 2010 IEEE MTT-S International*, pages 828–831.

Shi, J., Njoku, E., Chen, K., Jackson, T., and O'niell, P. (2003). Estimation of soil moisture with repeat-pass l-band radiometer measurements. In *Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International*, volume 1, pages 413–415 vol.1.

Skou, N. and Vine, D. L. (2006). *Microwave Radiometer Systems Design and Analysis*. Artech House, Norwood, MA.

Smith, S. W. et al. (1997). *The scientist and engineer's guide to digital signal processing*. California Technical Pub. San Diego.

Ulaby, F. T. and Long, D. G. (2014). *Microwave Radar and Radiometric Remote Sensing*. The University of Michigan Press, Ann Arbor, MI.

Ulaby, F. T., Moore, R. K., and Fung, A. K. (1981). *Microwave Remote Sensing*. Artech House, Dedham, MA.

Weng, Q. (2012). *An Introduction to Contemporary Remote Sensing*. McGraw-Hill's AccessEngineering. Mcgraw-hill.

APPENDIX A. SOURCE CODE

This appendix contains a copy of the source code used to create a total power radiometer in software and software that is able to load and parse information stored from this SDR-based radiometer. The first code displayed is the source code that creates the total power radiometer in software and analyzes the data generated from the SDR-based radiometer. The first code is the heir block created that detects the power and smooths the data. An heir block is a user created block that is used in GNURadio Companion. The second code supplied is Python code that could be used to read the data generated from the SDR-based radiometer and plot it. This provides an example of reading and parsing data generated from our SDR-based radiometer.

The code included in this appendix is provided as a point of reference. It may be out of date or incomplete. Copies of this thesis' source LaTeXcode, most experimental data, and additional code used may be found on the author's GitHub repository, https://github.com/matgyver/Radiometer-SDR-Thesis.

Python code for total power radiometer

This code defines a custom block in GNURadio Companion (GRC) that detects power, smooths the data and then decimates the data to reduce the storage size. This heir block may then be imported in GNURadio Companion and used like any other block in GRC. A screenshot of this block is shown in Figure A.1.

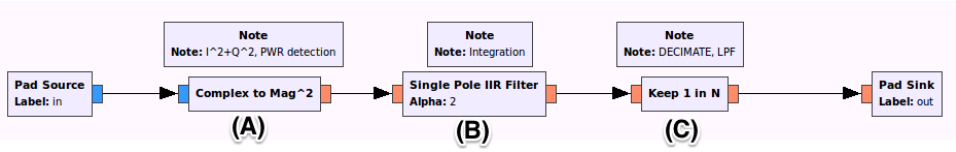

Figure A.1 Blocks used for creating a total power radiometer in software. Source: GNURadio Companion

Total Power Radiometer Block

```
#!/usr/bin/env python
##################################################
# Gnuradio Python Flow Graph
# Title: Total Power Radiometer
# Author: Matthew Nelson
# Description: Blocks for power detection, integration and
   LPF for a total power radiometer
# Generated: Sun Apr 12 23:03:59 2015
##################################################

from gnuradio import blocks
from gnuradio import filter
from gnuradio import gr
from gnuradio.filter import firdes

class TPR(gr.hier_block2):

    def __init__(self, integ=1, samp_rate=1, det_rate=1):
        gr.hier_block2.__init__(
            self, "Total Power Radiometer",
```

```
        gr.io_signature(1, 1, gr.sizeof_gr_complex*1),
        gr.io_signature(1, 1, gr.sizeof_float*1),
)

##################################################
# Parameters
##################################################
self.integ = integ
self.samp_rate = samp_rate
self.det_rate = det_rate

##################################################
# Blocks
##################################################
self.single_pole_iir_filter_xx_0 = filter.
    single_pole_iir_filter_ff(1.0/((samp_rate*integ)
    /2.0), 1)
(self.single_pole_iir_filter_xx_0).
    set_processor_affinity([1])
self.blocks_keep_one_in_n_4 = blocks.keep_one_in_n(
    gr.sizeof_float*1, samp_rate/det_rate)
self.blocks_complex_to_mag_squared_1 = blocks.
    complex_to_mag_squared(1)

##################################################
# Connections
##################################################
```

```
        self.connect((self.blocks_complex_to_mag_squared_1,
            0), (self.single_pole_iir_filter_xx_0, 0))
        self.connect((self.blocks_keep_one_in_n_4, 0), (self
            , 0))
        self.connect((self, 0), (self.
            blocks_complex_to_mag_squared_1, 0))
        self.connect((self.single_pole_iir_filter_xx_0, 0),
            (self.blocks_keep_one_in_n_4, 0))


    def get_integ(self):
        return self.integ

    def set_integ(self, integ):
        self.integ = integ
        self.single_pole_iir_filter_xx_0.set_taps(1.0/((self
            .samp_rate*self.integ)/2.0))

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.single_pole_iir_filter_xx_0.set_taps(1.0/((self
            .samp_rate*self.integ)/2.0))
        self.blocks_keep_one_in_n_4.set_n(self.samp_rate/
            self.det_rate)
```

```
def get_det_rate(self):
    return self.det_rate

def set_det_rate(self, det_rate):
    self.det_rate = det_rate
    self.blocks_keep_one_in_n_4.set_n(self.samp_rate/
       self.det_rate)
```

Python code for analyzing data

IPython notebooks was used to perform an analysis on the data used in this thesis. The code presented here is an example of using iPython to read and parse data from the SDR-based radiometer. The Markdown language as well as some HTML is used to create easy to read pages that include the python code, generated graphs and descriptive text.

Total Power Radiometer

```
#-*- coding: utf-8
#Radiometer Parsing Function

#This code shows an example of reading in and plotting data
   that is outputted from a GNURadio GRC file.
#In this example a Total Power Radiometer is developed in
   GNURadio GRC and uses the File Sink function to store the
   data.
#The plot then shows the total power output from the
   radiometer as a matched load is submerged in Liquid
   Nitrogen,
#then Ice Water and then left to dry.
# - - -
#
### Read the data

# Import Needed functions
```

```
# Import needed libraries
from pylab import *
import pylab
import scipy
import numpy
import scipy.io as sio
import csv


# Use this to set the filename for the data file and CSV
   Calibration file.


tpr = 'tpr_2014.06.12.Lab0.dat'
calib = 'tpr_calib_2014.06.12.Lab0.csv'
x2_data = 'tpr_x2_2014.06.12.Lab0.csv'


# Uses SciPy to open the binary file from GNURadio


f = scipy.fromfile(open(tpr),dtype=scipy.float32)


# Because of the valve function in GNURadio, there are zeros
    that get added to the file.  We want to trim out those
```

```
   zeros.

# In[5]:

f = numpy.trim_zeros(f)


# Create an index array for plotting.  Also, since we know
   the interval the data is taken, we can convert this to an
   actual time.

# In[6]:

y = numpy.linspace(0,(len(f)*.5),numpy.size(f))


### Plot the data

# In[7]:

plot(y,f)
xlabel('Time (sec)')
ylabel('rQ Values')
title('rQ vs Time')
grid(True)

pylab.show()
```

```
# ## Calibration

# The rQ values are the raw values from the total power
   radiometer and are uncalibrated.  While the graph shows
   the change in the total power recorded and shows that the
   radiometer can detect changes in noise temperature, it has
    no other meaning than that.  What we want is to show what
    the total power is in relation to a noise temperature.
   Since we have recorded the values of the rQ at fixed and
   known teperatures, we can create a calibration line and
   calibrate the radiometer.  For this experiment, we found
   that the following values matched our two known
   temperatures.
#
# |rQ Value|X^2 Voltage|Temperature
# |--------|-----------------------
# |.0977    | 1.9617     |77 K
# |.1507    | 2.085      |273.15 K
#
# We can now solve for y = mx + b since we have two
   equations and two unknowns.
#
# To work with this, a calibration file is created.  This is
    a very simple CSV file that contains 3 values: The raw rQ
    value, the raw voltage from the square-law detector (
   discussed later) and the observed temperature.  The table
```

```
   above would then look like the following in the file.
# '''
# .0977,1.9617,77
# .1507,2.085,273.15
# '''
# - - -

# We need to read in the values from our CSV file that
   contains the values

# In[67]:

read_csv = open(calib, 'rb')
csvread = csv.reader(read_csv)
rQ_values = []
temp_values = []
voltage = []

for row in csvread:
    rQ,volt,temp = row
    rQ_values.append(float(rQ))
    voltage.append(float(volt))
    temp_values.append(float(temp))
read_csv.close()

a = numpy.array([[rQ_values[0],1.0],[rQ_values[1],1.0]],
   numpy.float32)
```

```
b = numpy.array([temp_values[0],temp_values[1]])

z = numpy.linalg.solve(a,b)
print z


# Now we apply these values to the array that holds our raw
   rQ values

g = f*z[0]+z[1]


# Now we can re-plot the graph but this time with the
   calibrated noise temperatures

plt.figure()
plot(y,g)
xlabel('Time')
ylabel('Noise Temperature (K)')
title('Temp vs Time')
grid(True)

pylab.show()

# This is looking better, but the time at the bottom doesn't
    have much meaning.  Since we know the sample rate of the
   Software Defined Radio, we can calculate the time interval
```

```
    between each sample.

# - - -
# # Square-law data
#
# We now want to look at the data from the Square-Law
   detector to verify the operation of the SDR.  In the
   experiment that was conducted above, a power splitter was
   used to split the RF signal so that one went to the SDR
   and the other to a square-law detector (with a 3.1 dB loss
    though).  Therefore both data should be the same.  Let's
   read and then plot this data.

read_csv = open(x2_data, 'rb')
csvread = csv.reader(read_csv)
dummy = []
x2_voltage = []

for row in csvread:
    dummy,x2voltage = row
    x2_voltage.append(float(x2voltage))
read_csv.close()


# Like the SDR data, we want to have a time reference at the
    bottom.
```

```
w = numpy.linspace(0,(len(x2_voltage)*.01),numpy.size(
   x2_voltage))

plt.figure()
plot(w,x2_voltage)
xlabel('Time (sec)')
ylabel('Voltage (V)')
title('X^2 Voltage vs Time (Noisy)')
grid(True)

pylab.show()

# The Square-law detector doesn't have a filter on it unlike
    the data we get from the SDR.  The GNURadio program takes
    the data and applies a Low Pass Filter to "clean up" the
   information.  We need to do the same with our Square-law
   data.

from scipy import signal
N=100
Fc=2000
Fs=1600
h=scipy.signal.firwin(numtaps=N, cutoff=40, nyq=Fs/2)
x2_filt=scipy.signal.lfilter(h,1.0,x2_voltage)


plt.figure()
```

```
plot(w,x2_filt)
xlabel('Time (sec)')
ylabel('Voltage (V)')
title('X^2 Voltage vs Time')
axis([0, 610, 1.94, 2.12])
grid(True)

pylab.show()

# Now we wish to calibrate this data as well.  We will use
   the same file and use the calibration points in that file.

a = numpy.array([[voltage[0],1.0],[voltage[1],1.0]],numpy.
   float32)
b = numpy.array([temp_values[0],temp_values[1]])

z = numpy.linalg.solve(a,b)
print z

x2_calib = x2_filt*z[0]+z[1]

plt.figure()
plot(w,x2_calib)
xlabel('Time (sec)')
ylabel('Voltage (V)')
title('X^2 Noise Temp vs Time')
axis([0, 610, 70, 300])
```

```
grid(True)

pylab.show()

# This looks to be the same as our SDR graph, but let's
   overlay them to make sure

plt.figure()
plot(w,x2_calib,'r',label='X^2')
plot(y,g,'b',label='SDR')
xlabel('Time (sec)')
ylabel('Voltage (V)')
title('Noise Temperature vs Time')
axis([0, 610, 70, 300])
grid(True)
legend(loc='lower right')


# We have some timeshift due to two reasons.  One, the
   timing isn't always perfect when starting the collection
   of the two data sets.  And two, we get a timeshift from
   filtering the square-law data

pylab.show()
```