Implementation of a Total Power Radiometer in Software Defined Radios

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 (Embedded Systems)

Program of Study Committee:

Phillip H Jones, Major Professor

John Basart

Mani Mina

Iowa State University

Ames, Iowa

2015

DEDICATION

I would like to dedicate this thesis to my wife Jennifer who has stood by me through this long journey towards my Masters. I would also like to thank my parents without whose support I would not have been able to complete this work.

I would also like to thank my friends and family for their continued support and constant encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

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

A software defined radio is defined as a communication system where components of a communication system that are typically done in hardware are now done in software. The result is a highly flexible communication system that can adapt to changes to the system based on requests by the user or due to conditions in the radio frequency channel. Software Defined Radios (SDRs) have been used for a variety of applications, mostly in the area of communications. However, they have not been widely applied to remote sensing applications such as a radiometer. A SDR based radiometer offers a very flexible and robust system more so than a more traditional radiometer which has fixed hardware and usually little room for flexibility and adaptability based on the needs of the radiometer application or due to changes in the radiometers environment. The price of SDRs have also been steadily dropping making implementing them into radiometers a more attractive option compared to using traditional RF hardware. In this thesis we will look into the feasibility and theory of using an off the shelf SDR hardware platform for a radiometer application. In addition, we will look into how we can use the GNURadio software to create a total power radiometer within software and how this software can be used to make easy changes to the functionality of the radiometer. Finally we will look at the preliminary results obtained from laboratory tests of the SDR radiometer system.

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, to 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 in to 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 of 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

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.

Three areas closely related to the work in this thesis are:

1. Digital radiometers,
2. Software defined radio based radiometers, and
3. Radio frequency interference mitigation (RFI).

2.1 Digital Radiometers

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

1. Hybrid and
2. 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. When the the ISU radiometer was built, an analog to digital converter (ADC) that could sample accurately at 1.4 GHz were expensive and not easily obtainable. Since this radiometer was only interested in power information, it could undersample at 1.4 GHz. The samples are sent to a Field Programmable Gate Array (FPGA) to extract power information. The correlation stage is another place where a radiometer could be digitized. [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 relativly new subclass of digital radiometers. With the advent of software defined radios that are wildly available, their has been increasing interest in applying this technology to radiometers.

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 mitigation, offending signal detection and a software implemented noise generator. Additionally, elements of the graphical user interface (GUI) were enhanced to aid in 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,
2. Python scripts and

for low-level communications with their software defined radio. The students reported on the project that their SDR based radiometer worked well to implement a higher level user interface. One aspect that differentiates the work in the thesis and Grand Valley State Universities work is that they build their own custom hardware for their software defined radio, while in this work, off the shelf components where used with an aim of making radiometers more wildly accessible to the research community.

This section discussed 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.

2.3 Radio Frequency Interference (RFI) Mitigation

When an RF signal generated from a source other than the object or phenomena of interest interferes (i.e. masks or contaminates) with the RF signal of interest to a radiometer this is referred to as radio frequency interference (RFI). Radio Frequency Interference (RFI) 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 experiments. It is for this reason certain frequency bands have been designated protected frequencies for radiometer use by the international community. 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, statiscal methods have been explored that loot at variations in the received power to determine when RFI is occurring.

With respect to RFI mitigation, the use of the kurtosis statistic method[De Roo et al. (2007)] or polarization signature method. To mitigate the offending signal, mechanical filters are used to selectively filter out the offending signals [Misra et al. (2012)].

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

CHAPTER 3. BACKGROUND

This chapter will examine background information on the basic operation of a radiometer, a software defined radio and background on the development platform used to build a software defined radiometer. We begin with an overview of a traditional radiometer and how this type of radiometer works. This is followed by high level examination on how a software defined radio operates. Finally we will discuss both the hardware and software selected and used for our development platform to create this software defined radiometer.

3.1 Radiometer Basics

The primary task of a radiometer is to measure power, or more precisely, to accurately measure the correct power within a certain degree of accuracy. In order to accurately and within a high degree of precision measure power a radiometer must take into account factors such as the system noise, the bandwidth of the signal and the stability of the system as a whole[Evans and McLeish (1977)].

The amount of noise that is generated by the object of interest is due to the thermal agitation of the charge carriers, usually the electrons, which is directly correlated to the physical temperature of the source [Nyquist (1928)]. It is because of this correlation that we often refer to the amount of noise received as the noise temperature and it is measured in Kelvins.

In all radiometers there are six stages that is common in all radiometers. They are:

1. Source (antenna or T_A)

2. Amplification (Gain or G),

3. Bandwidth (β),

4. Power detection (X^2),

5. Integration (τ),

6. Output (Voltage, rQ or Kelvin).

Figure 3.1 shows each stage as they relate to each other. We begin with our source which is often an antenna. Amplification or gain in the system then amplifies this signal so that it is easier to measure changes in our noise. Our bandwidth is often restricted by filtering and can occur before or after the amplification stage and in some cases filtering is done before and after the filtering stage. Power detection extracts the power information from the signal. Because the signal is often noisy, we use integration to smooth our signal and improve our sensitivity. Finally our output is the total power information. This information may be as simple as a voltage output, an uncalibrated value referred to as rQ, or may be a calibrated noise temperature measured in Kelvins.

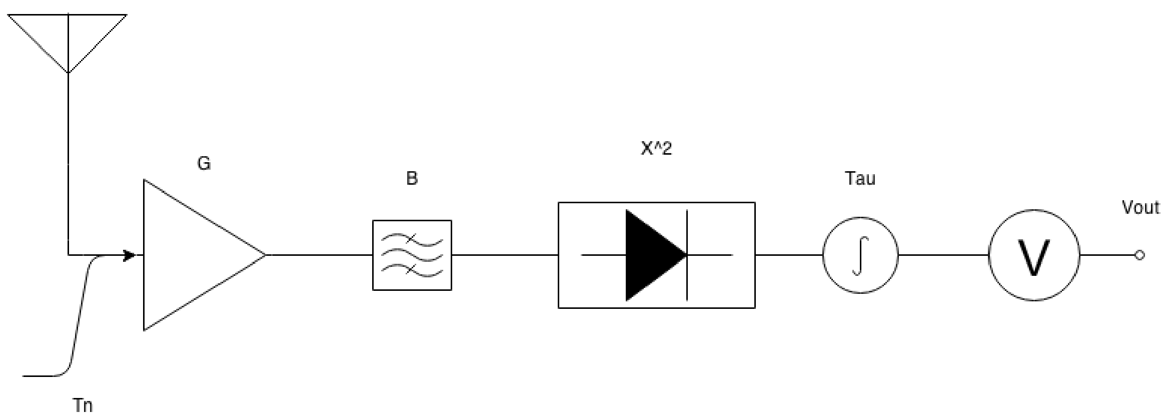

Figure 3.1 A total power radiometer block diagram

In figure 3.1 we have one more component that is common in all radiometers but is not a physical component or stage in the radiometer. That item is additional noise that gets added to the system from within the system itself is called system noise or T_N.

In addition to the system noise, another very common problem with radiometers is stability. Most instabilities in a radiometer are due to fluctuations that occur in the amplification stage or the Low Noise Amplifiers (LNAs) used. There are two factors that cause changes in the gain values, voltage feeding the LNA and the physical temperature of the LNA. These gain fluctuations can be controlled to a point by closely monitoring and controlling both the voltage and the temperature of the LNAs. However, this is not an easy task and in some cases is not practical. Therefore, other methods have been developed to compensate for these fluctuations. There are three common types of radiometers designed to adjust for gain fluctuations. They are:

1. Dicke radiometer,

2. Noise injection radiometer,

3. Polarimetric or correlating radiometer.

One of the first methods used is the Dicke radiometer which switches between the measurement of the antenna and a known noise source[Dicke (1946)]. By referencing this known source very quickly the Dicke radiometer can account for and greatly reduce gain fluctuations.

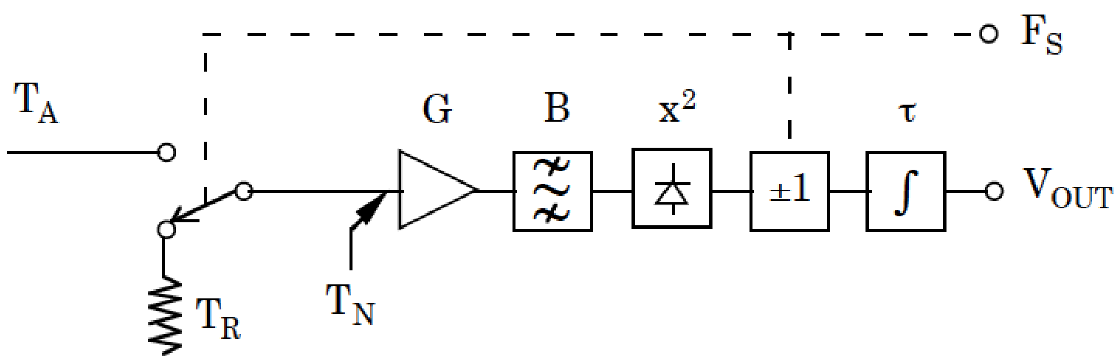

Figure 3.2 A block diagram of a Dicke radiometer

While a Dicke radiometer greatly reduces the fluctuation in the gain of the radiometer and improves stability, it does so at the cost of not seeing the object of interest while it is referencing the known source. This decreases the overall sensitivity of the radiometer.

A noise injection radiometer is a variation of the Dicke radiometer where a variable noise source is used and injected into the RF chain as seen in Figure 3.3. The output of this noise

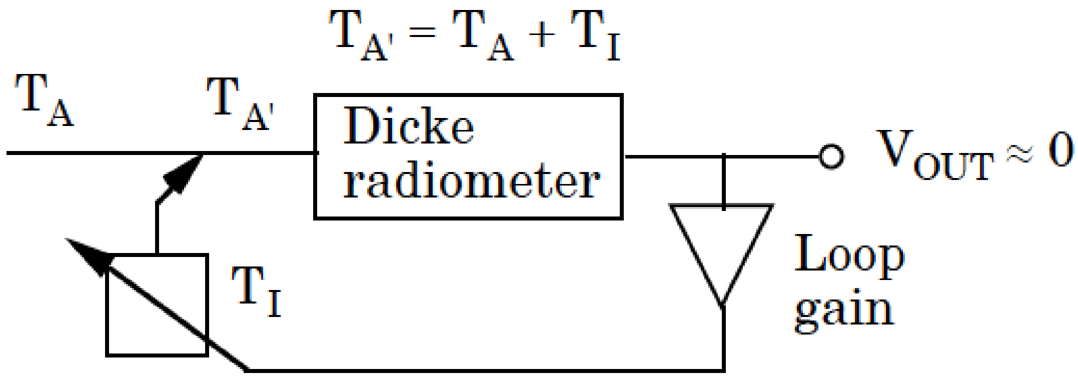

Figure 3.3 A block diagram of a Noise Injection radiometer

source is then adjusted so that the noise input plus the signal from our source is equal to the reference noise. This completely eliminates the gain fluctuations however increases the system noise which also reduces our sensitivity of the radiometer.

Our source signal is often times assumed to be not polarized. However, this is not the case and an incoming signal often has some polarization. A correlating radiometer [Fujimoto (1964)] uses a dual polarization antenna to split the vertical and horizontal polarization of the signal. Each of these signals is then fed into a radiometer and is correlated. Since gain fluctuations or uncorrelated, they are reduced in the system. This reduces gain fluctuations and also helps maintain the sensitivity of the radiometer. However it adds a much greater complexity to the radiometer and requires two identical receivers.

3.1.1 Measuring RF power

To measure power in a radiometer, several factors are taken into consideration. To begin with we have the noise signal coming from the antenna. Our antenna is assumed to be looking at our target of interest and it is assumed that we can relate the antenna noise to the noise from the source. It is often easier to refer to this noise as the brightness temperature. Therefore, the brightness temperature of the source can be related to the brightness temperature at the antenna. We will refer to this brightness temperature as T_A.

To calculate our power from the radiometer, we multiply each item with the Boltzmann constant referred to as k. Equation 3.1 gives us the total power, in watts, of an ideal radiometer.

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

As discussed in section 3.1, the system noise is another component of the radiometer that needs to be addressed. Figure 3.1 shows the additional noise that is injected into the system.

As it can be seen, this additional noise is added to the noise coming from the antenna source. Therefore, T_N is added to T_A and our final equation for the power measured is shown in equation 3.2.

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

Figure 3.1 showed us a more typical radiometer and was discussed in the previous section. However, power detection and the associated voltage output has not been discussed. Power detection is accomplished typically with a square-law detector and this output is often run through an integrator to smooth the output[Leinweber (2001)]. Finally we have the output which is a voltage represented as V_{out}. This results in equation 3.3.

$$V_{OUT} = c * (T_A + T_N) * G \tag{3.3}$$

Here V_{OUT} is shown by the addition of both the noise from the system T_N and the noise from the antenna, T_A and multiplied by the gain in the system, G [Skou and Vine (2006)]. A constant factor c is useful for when we are looking at a radiometer like a Dicke radiometer in which the value of c is $\frac{1}{2}$. In most applications outside of a Dicke radiometer, c is 1 and can be ignored.

The voltage output from this radiometer is then either measured or may also be sampled by an analog to digital converter. This voltage then represents the total power measured by the radiometer, however this measurement is not calibrated.

3.1.2 Integration and filtering

Filtering with a traditional radiometer is usually accomplished by using mechanical filters. Often these are band-pass filters that limit the bandwidth that the radiometer sees. Other filters, such as a low pass filter are also used, but usually to smooth out the output from the square law detector. Another item used to help smooth out the signal from the square law detector is an integrator.

In a traditional radiometer, we can integrate by using a simple RC circuit, which consists of a resistor and capacitor.

To begin, we will examine how a RC filter is analogous to an integrator where the R and C values determine our time constant and our integration time for the filter[Aitken (1968)]. We know a RC filter is analogous to an integrator by looking at equation 3.4.

$$\frac{1}{RC}\int V_i dt \tag{3.4}$$

To begin with, we look at what an analog RC filter looks like.

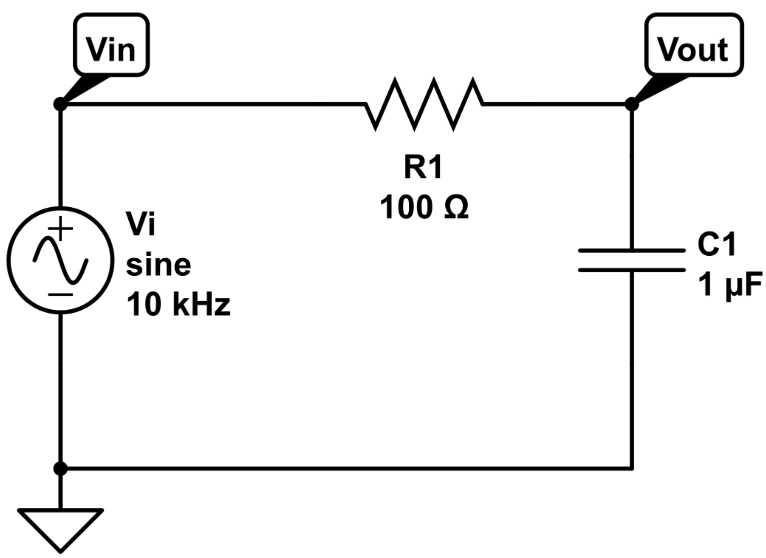

Figure 3.4 A simple RC circuit

This circuit can be represented by equation 3.5.

$$\frac{V_{in} - V_{out}}{R} = C\frac{dV_{out}}{dt} \tag{3.5}$$

This integration smooths out the signal from the square law detector and also improves our sensitivity of the radiometer. This will be further explained the next section.

For the cutoff frequency of a RC circuit, we know that it has the relationship shown in equation 3.6.

$$f_c = \frac{\sqrt{3}}{2\pi RC} \rightarrow RC = \frac{\sqrt{3}}{2\pi f_c} \tag{3.6}$$

The RC term gives us our time constant of the circuit and can be used to calculate out our coefficients.

3.1.3 Radiometer Performance Metrics

There are two criteria that determines how well a radiometer performs. The first criteria is with the sensitivity of the radiometer. This tells us how well the radiometer can differentiate between the information we want and the noise we do not want. The second criteria is stability.

Sensitivity. Sensitivity of the radiometer relates to the amount of power that the radio selects from the antenna. This selection is then dependent on the bandwidth that the radiometer is able to listen to. The radiometer however detects not only the signal of interest but also receives a noise signal as well. This noise is added to the signal and can not be separated from the signal. Because this noise is added to the signal, we must be able to determine a change in the signal while the noise signal is also present.

To understand this, let us look at the example of T_A has a value of 200 K and T_N has a value of 800 K. Since T_N is added to our antenna signal, we have a total noise temperature of 1,000 K. This means that if we want to detect a change as small as 1 K, we must be able to measure the difference between 1,000 K and 1,001 K[Skou and Vine (2006)].

The ability of a radiometer to detect these small changes is the radiometer's sensitivity, or the standard deviation of the output signal from the radiometer. This sensitivity is also referred to as the Noise Equivalent Delta (Δ) Temperature or NEΔT and is shown in equation 3.7.

$$NE\Delta T = \frac{T_A + T_N}{\sqrt{\beta * \tau}} \tag{3.7}$$

The sensitivity of the radiometer is based on both the bandwidth, β, of the incoming signal and the integration time, τ. As it can be seen in equation 3.7, we want to have as much bandwidth as possible. In a traditional radiometer, this bandwidth is often fixed and is dependent on the band-pass filters used in the radiometer. We can however control τ and a longer integration time will help improve the sensitivity of the radiometer to a certain degree.[Ulaby et al. (1981)]

Stability. Stability and accuracy are additional problems that need to be considered when looking at the radiometer system. To begin we can once again look at the power received equation 3.2.

As we look at this equation, we can see that if k, β, G, and T_N are constant, then stability can be assured. The Boltzman constant k is a known constant and we can also assume that our bandwidth, β, will also remain constant. Gain and the noise temperature however will vary.

Fluctuations in gain is the largest factor that affects stability in the system and this can be seen in equation 3.8. As discussed earlier, it is this factor that has been a driving force for changes to the design of a radiometer as demonstrated by the Dicke and Correlating radiometer designs.

Gain fluctuations are caused by two factors: the physical temperature of the amplifier and the voltage that feeds the amplifier. High accuracy voltage regulators can help control fluctuations in voltages that can in turn affect the gain. A factor however that is harder to control is temperature. Various methods have been used to control the temperature when the radiometer is used in fluctuating temperature environments such as the outdoors or in space.

$$\Delta T_G = T_{sys}\left(\frac{\Delta G}{G}\right) \tag{3.8}$$

With stability we need to either attempt to stabilize the radiometer as best as we can or compensate for the gain fluctuations. Compensation has been discussed and three types of radiometers have been identified that attempt to compensate for these fluctuations. To control the stability many radiometers will use highly accurate voltage regulators and will control the temperature of the LNA through methods such as thermal electric coolers. Other methods to control or account for instability has lead to the development of other types of radiometers such as the ones discussed in section 3.1.

3.2 Software Defined Radios Basics

The basic concept behind a SDR is that it will digitize the RF signal as soon as possible. Once digitized, it is now evaluated by a computer, FPGA, or a dedicated System on Chip (SoC). A canonical software defined radio architecture is one that consists of a power supply, antenna, multi-band RF converter, and a single chip that contains the needed processing and memory to carry out the radio functions in software [Mitola (1995)]. This allows us to extract certain hardware functions, such as filtering, into the digital domain which can then be manipulated by software. Since software is now manipulating the signal, we can rapidly change what functions we execute on the signal by changing the software. This gives SDRs a high amount of flexibility as components that are normally done in hardware can now be done in software and can be changed by simply uploading new software or firmware to the system. This also gives us a benefit in cost as certain components are no longer needed and changes done in software do not require additional hardware to be added or to be swapped out.

An ideal software defined receiver simply has an antenna connected to an analog to digital converter which sends information to a processing unit such as a Field Programmable Gate Array (FPGA) or computer. For a transmitter, we reverse it and use a digital to analog converter to produce the correct waveform which is then transmitted by the antenna. In reality, SDRs require some additional hardware to make a viable working transceiver. Amplification is still required to amplify the incoming signal and to amplify the signal going to the antenna. Some SDRs also use a mixing stage to move a high frequency signal to a lower frequency signal. This allows for less expensive analog to digital and digital to analog converters to be used.

3.2.1 Software Defined Radio Normal Operation

Software Defined Radios (SDRs) are used for a variety of applications but their primary application has been in the area of communications. They appeal to applications where being able to change a modulation scheme or filter on the fly is desirable. In these areas, SDRs often outperform a traditional hardware only radio with their ability to rapidly change their operations by simply changing their software. Early SDRs were expensive due to the high costs in the analog to digital converters (ADCs) needed and in the high speed Field Programmable Gate Arrays (FPGA) used. In recent years however, the cost of SDRs have decreased due to the cost of these key components decreasing in cost as well. Even though the cost has gone down the performance of SDRs have increased and this has lead to new developments in applications for using SDRs in new and different ways.

Software defined radios have been used in a number of applications. Some applications they have been used in are:

1. Cellular communications,
2. Wireless local area networks,
3. Personal area networks,
4. Digital broadcast.

There are of course other applications than those listed and more applications added as new communication methods continue to evolve [Jondral (2005)]. But this is the strength of a software defined radio, it is capable of performing all of these operations and can be adapted to new ones by simply updating the software that defines it.

3.3 Software Defined Radio Development Platform

The work of this thesis is to use an off the shelf software defined radio (SDR) to perform the same operation or better of a traditional analog radiometer. Using a SDR radio also means that we are able to be more flexible in how the radiometer performs, is capable of frequency agility and adapting to changing conditions such as interference. Using a software defined radio also allows for implantation of different radiometer types such as a correlation radiometer or a polarimetric radiometer that uses Stokes parameters [Wang et al. (2012)]. Normally, these require changes to hardware, but all of these types of radiometers can be implemented in software increasing the flexibility of the system.

3.3.1 Hardware Platform

The key component for a software defined radio radiometer is the hardware that will do most of the work of sampling and processing the signal. The equipment selected for this work is the Ettus Research Group N200 SDR and can be seen in figure 3.5. The N200 has the following features that made it well suited for our specific application:

- Dual 14-bit ADC,
- 50 MS/s Gigabit Ethernet streaming,
- Modular daughter-board system for RF front end

This SDR utilizes daughter boards as the RF front end to the SDR and up to two daughter boards may be installed into a N200. Another important reason the N200 is selected was due to its ability to handle up to 50 MHz of bandwidth to the computer and up to 25 MHz of RF bandwidth per daughter-board plugged in to the SDR. This means that it is possible to have two receive cards that can stream up to 25 MHz bandwidth each.

The N200 utilizes a flexible architecture through the use of daughter-boards for a variety of RF interface systems based on the frequency range desired and if receive and/or transmission is needed. Figure 3.6 shows the overall architecture of the N200 SDR. These daughter boards directly receive the RF signal and then outputs the analog I and Q signals that are then sampled by the N200 A/D converter for reception or receives the I and Q values from the N200 D/A converter for transmission.

The flexible architecture of the N200 and the ability for it to handle large amounts of bandwidth made this hardware ideal for the software defined radio radiometer work done in this thesis.

Figure 3.5 The USRP N200 from Ettus Research (Image from Ettus Research Website - www.ettus.com)

The DBSRX2 Receiver. The daughter board selected is the DBSRX2 card as this card is receive only and operates between 800 MHz and 2.4 GHz. An image of the daughter-board can be seen in figure 3.7. The DBSRX2 also has built in amplification that is adjustable through software. These daughter boards have the required RF hardware for the signal to be processed. In this application it was required that the signal was detected at 1.4 GHz with a bandwidth of 20 MHz. The DBSRX2 receiver met this requirement and was selected to be used with the N200. In this radiometer application transmission is not needed and is illegal in the 1.4 GHz band, which is reserved for passive radiometer applications.

The DBSRX2 has several key components on it that is used to take the analog RF signal and prepare it for digitization by the analog to digital converter. First the signal is amplified through a Programmable Gain Amplifier (PGA). This PGA is accessible from the software and can be configured by the software. Next the signal goes into a direct-conversion integrated circuit that directly converts the RF signal to analog I and Q values. The integrated circuit,

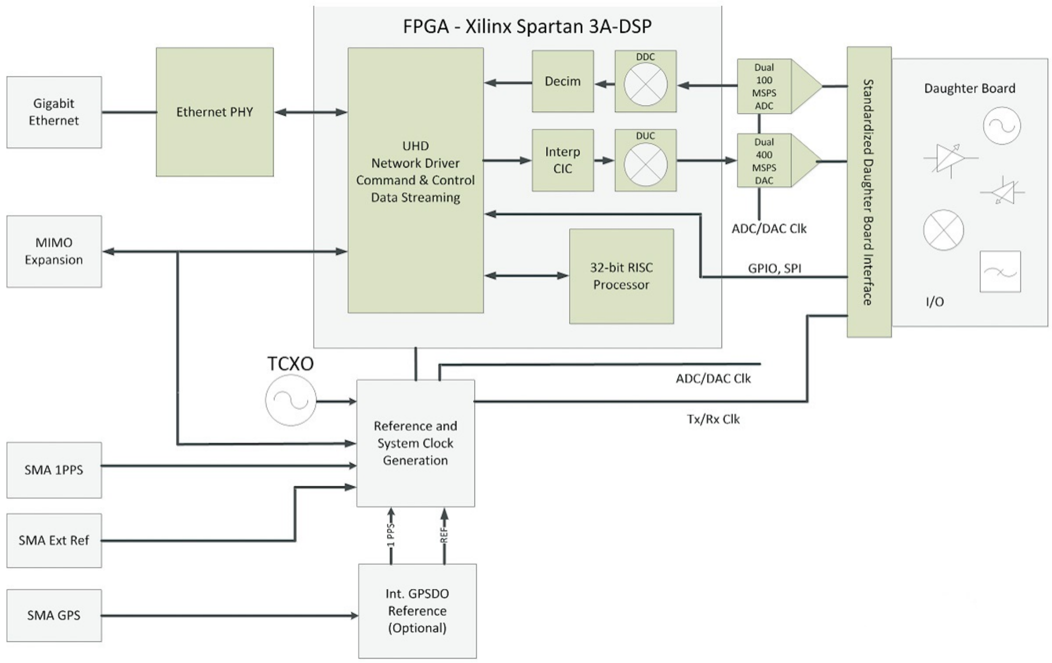

Figure 3.6 A block diagram of the Ettus N200 SDR. (Image from Ettus Research Website - www.ettus.com)

a Maxim MAX2112 device, also includes a Low Noise Amplifier (LNA), mixer and Low Pass Filter (LPF). This essentially amplifies the signal, mixes into baseband and then applies a low pass filter.

These analog I and Q values are then passed to the N200 to be sampled by the analog to digital converter. The IQ values are differential signals to minimize noise possible interference.

Since we are using the DBSRX2 after the LNAs that are already in use, the noise temperature added by the DBSRX2 will be small. The DBSRX2 adds approximately 5 dB to the noise factor of the system. Again though, since this is at the end of the RF chain, the total contribution of the DBSRX2 to the overall system noise temperature is small, and has been calculated to be 1.05 dB to the overall noise factor of the system.

3.3.2 Software Platform

Software of course plays a critical role in a software defined radio and also in our software defined radio radiometer. There are two pieces of software that are in play with the software defined radio we are using. The first is the firmware that is used in the FPGA of the N200. This

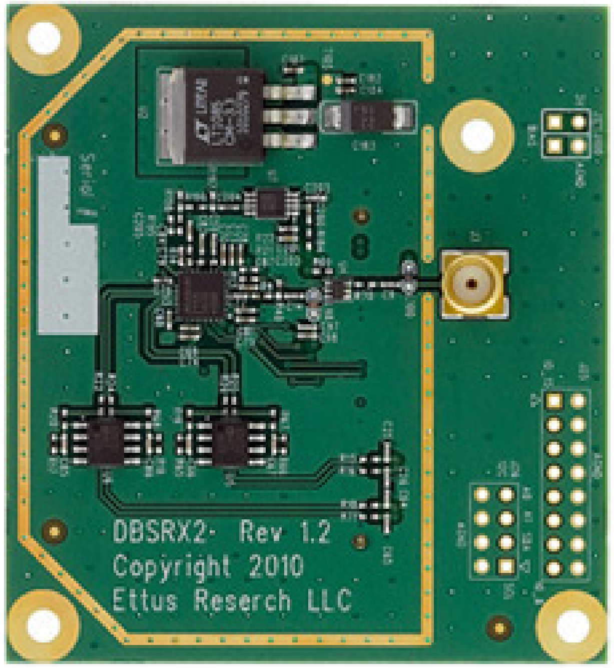

Figure 3.7 The DBSRX2 daughter board from Ettus Research (Image from Ettus Research Website - www.ettus.com)

firmware provides low level processing of the signal so it can be sent to the software located on the computer. It also provides a link for controlling key aspects of the software defined radio such as additional gain, bandwidth and the center frequency. This firmware is already pre-loaded into the FPGA by Ettus Research and can be upgraded through tools provided by Ettus Research.

The second is the software that is running on the host computer. It is this software that provides the calculations on the I/Q data to give us the information we need and also creates a GUI for the user to interface with the radio. For this software, we will be using GNURadio, an open source software program that is used in software defined radios including the N200 SDR that we have.

GNURadio will be used to do all signal processing that is needed. GNURadio is an open source software define radio framework that runs on multiple OSes and offers a rich set of features. In addition, GNURadio is well supported by the Ettus Research group and is the preferred software for interfacing with their hardware.

An easy to use interface was another driving requirement for our implementation of a radiometer in a SDR. GNURadio helps us with this through the use of the GNU Radio Companion or GRC. This was important as it was anticipated that operators of the radiometer have a limited knowledge about programming. GRC uses a simple to use graphical system to design and build radio components in software.

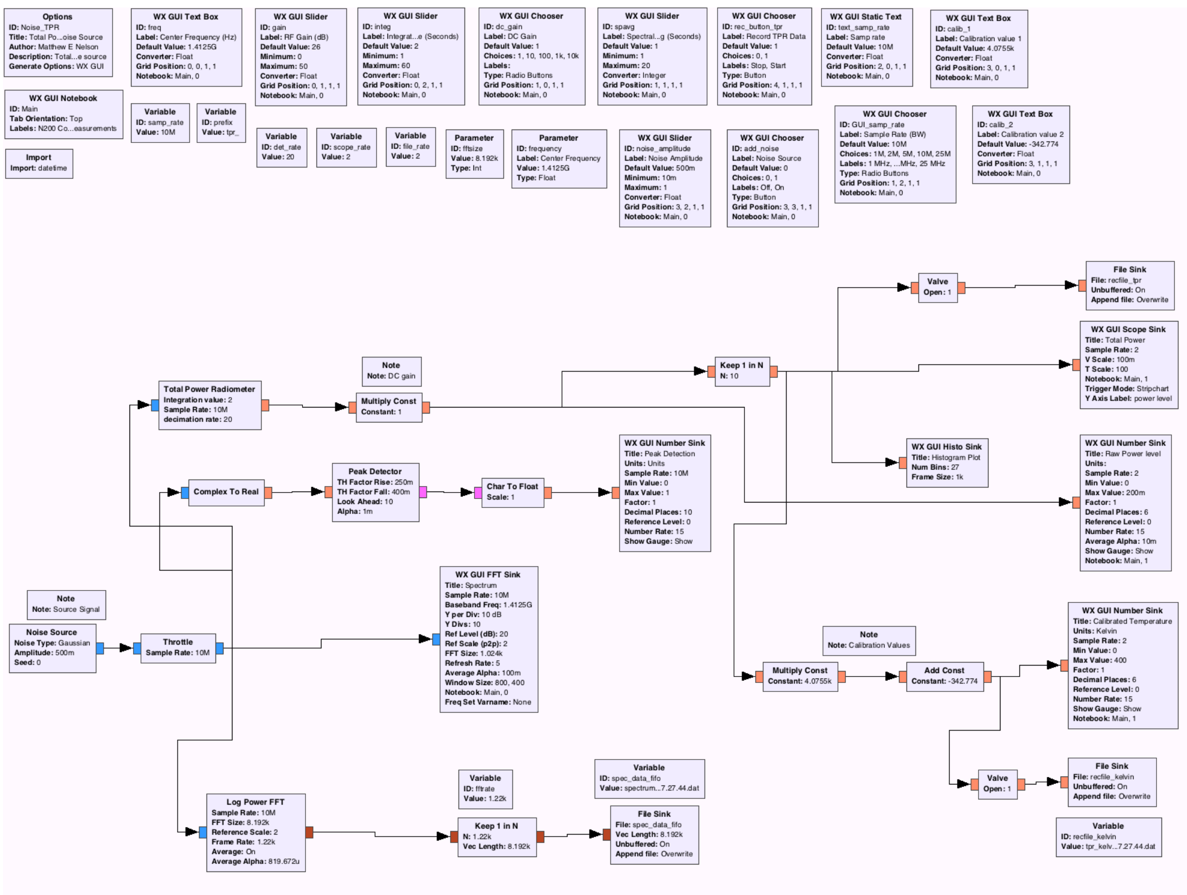

Figure 3.8 A screenshot of the GNURadio Companion editor program. Source: GNURadio

GNUradio Companion works by having the common functions, such as signal sources, signal processing and signal sinks, as blocks that can picked and placed on the screen. Once placed, the blocks can be wired up, much like LabView, and the flow of data can be controlled in this fashion. While GNURadio Companion provides most of the essential blocks used in most applications additional blocks can by added if needed. This is because GNURadio is built using

Python and the blocks in GNUradio Companion are simply blocks of Python code. To compile a GNURadio Companion flow diagram, you simply run the sheet, which then generates the Python code that is then executed.

GNURadio uses a combination of Python and C++, where Python handles most of the interface and the C++ is most of the drivers and low level interface to the hardware. This allows for an easy to use system but still meets the demanding performance needed for handling large amounts of data.

GNURadio Companion also includes blocks that allow it to create a GUI type of interface. The typical method it uses for this is using wxGUI although GNURadio Companion does also include blocks that can use QT for generating widgets as well. However, the wxGUI tends to work better and has better support in GNURadio than QT.

Through these blocks we are able to only manipulate the data we need to perform a total power radiometer in software but to also create a user interface that allows us to control the radiometer as well. We are also able to display the information in real time so the user can see changes in power and even monitor spectral information during the operation of the radiometer.

CHAPTER 4. SOFTWARE DEFINED RADIOMETER IMPLEMENTATION

One of the principal goals with this research was to implement a fully functioning total power radiometer within software. The N200 provides us the link between the RF signal captured by the antenna and converts that to a format that can now be used by the to manipulate the signal. Once the signal has been passed to the computer, GNURadio will implement the correct algorithms to detect the power within the signal, filter and output the information.

One of the advantages of course with a SDR is that filtering can also be done within the software. In addition, thanks to the WxGUI that GNURadio uses, we can also build a user interface that can control several key variables that are useful for us. This includes controlling the gain on the programmable gain amplifier on the DBSRX2 card, the sampling or bandwidth of the signal, the center frequency, and also the integration time. All of these can now be controlled in real time as well. GNURadio will also store the power data so that we are able to do further analysis of the data using a software program like Matlab or Python. Because GNURadio is a very flexible system we are able to do more with the signal such as filtering, polarimetric radiometer, and frequency analysis. We are also able to add additional features and improvements through updates to the software.

4.1 Requirements

To help us quantify the required performance of the radiometer, information was provided to us by Dr. Brian Hornbuckle and also derived from existing radiometers. The requirements given are outlined in table 4.1.

These requirements mainly drive the hardware requirements needed for the software defined

Table 4.1 Required Radiometer performance

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

radio radiometer. Chapter 3 Section 3.3 provides more in depth information on the hardware used and why it was selected.

Software Requirements. The driving force for the software requirement was to have a system that was easy to use yet powerful enough to handle the amount of data that is required. One reason for the development of this platform is to make radiometers more accessible to other researchers and other programs such as education and even amateur radiometer work. Therefore, ease of use was taken into consideration when selecting the hardware and the associated software used with it.

The data flow model for the hardware selected uses the FPGA to perform low level signal processing on the signal and moves high level processing to the host computer. This allows for less processing requirements on the physical hardware but requires a host computer that is able to process the incoming information. It also requires a software package that is able to process this information efficiently.

Because a requirement is an easy to use system GNURadio was selected as it includes GNURadio Companion (GRC). GRC is a supplemental program which uses a graphical interface for creating the radio environment. GNURadio and GRC is discussed in greater detail in Chapter 3, section 3.3.2.

This software meet the criteria of allowing a simple to use interface to be built and used in the control and data recording of the information required. It also uses a simple interface for making changes to the program. These changes can be both in the GUI and also to how the program processes the information.

4.2 Theory of Operation of a Software Defined Radio Radiometer

A Software defined radio consist of both hardware and software that allow it to perform the operations of a radio or communication channel. A software defined radio used for radiometer applications is identical to a software defined radio used for, as an example, a 802.11b radio with one major difference. Since we need to amplify the signal more than what most communication applications require; we do require more powerful or additional LNAs to boost this signal. In addition, since the first LNA plays a major role in the overall system noise and this system noise does affect performance of the radiometer, the selection of this LNA is important. However, all other components are the same components used in other applications.

A software defined radio radiometer behaves analogous to a more traditional radiometer and thus the application is the same as a traditional radiometer. This includes applications such as radio astronomy that includes applications in Earth Science such as soil moisture and ocean salinity[Ruf and Gross (2010)]. A software defined radio radiometer can also allow for new application development that can expand the remote sensing field. Since we have moved the majority of the hardware to software this allows us to further shrink the size and weight of the radiometer. This allows for other radiometer applications such as Unmanned Aerial Vehicles (UAVs) for scanning soil moisture and ocean salinity remotely[Mclntyre and Gasiewski (2007)].

We will now introduce the three major components that make up a software defined radio radiometer.

4.2.1 RF Front End

The RF front end plays a critical role in the radiometer as the LNAs used in the front end has a large impact on the system noise generated by the radiometer itself. A traditional radiometer utilizes both amplification through the LNAs and also includes filtering to the desired bandwidth. A SDR radiometer does not require the filters as we are able to create these in software, however the amplification stages need to remain.

A typical RF front end uses a two to four stage Low Noise Amplifier (LNA) to amplifier the noise while keeping the noise contributed to the system as low as possible. As with any radiometer, the first LNA is the most critical as it contributes the most to the overall system noise temperature. For this reason a LNA that did not have a large gain but had a low noise figure is chosen. The second and third LNA has higher gain values at the cost of a higher noise figure, although not by much. However, since they are further down the chain, they do not contribute as much to the total system noise.

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \frac{F_4 - 1}{G_1 G_2 G_3} + \cdots + \frac{F_n - 1}{G_1 G_2 G_3 \cdots G_{n-1}} \tag{4.1}$$

Equation 4.1 shows us how the noise factor and gain of the amplifiers affect each other. It can be shown that the first amplifier or first noise figure in the system contributes the most to the overall system noise figure. Additional components contribute, but at a much lower contribution.

4.3 Mapping Traditional Radiometer Functions to a Software Defined Radio Radiometer

In order to recreate a radiometer in software we need to identify the key components of a radiometer and then recreate those components in software. As discussed in chapter 3, the three components identified that are key to a radiometer is listed below.

1. Power detection
2. Integration
3. Bandwidth limitation or filtering

The following sections will now examine how these items are mapped from their analog component to the software or digital component.

4.3.1 Power detection

Power detection is a key ability that allows a radiometer to function. At its core a radiometer is a power detector. Therefore, the implementation of power detection is a crucial function of a software defined radio radiometer.

A traditional radiometer uses a square-law detector which takes the input signal and produces a voltage that is proportional to the square of the voltage. This allows us to take an analog RF signal and convert the noise voltage that for all intense and purposes has a mean value of zero, and produce a noise power.

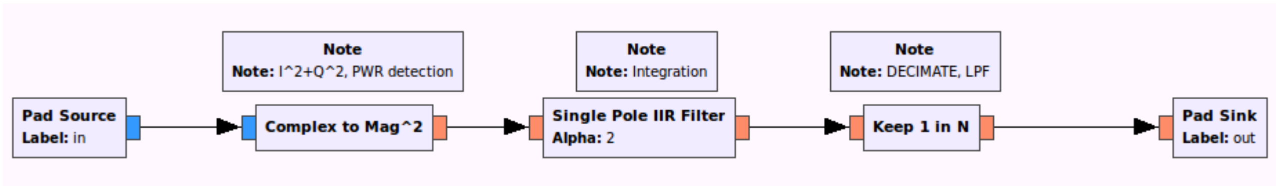

Figure 4.1 A block diagram showing how the radiometer performs the equivalent square law detector in software.

To implement this in software we need to build a square law detector mathematically. A square law detector mathematically is the sum of the squares. Once the signal has been digitized, it is expressed in data bits of I and Q, which represent in-phase and quadrature-phase of the signal. By squaring each term, we get the desired result of the power of the signal [Sarijari et al. (2009)][Rashid et al. (2011)] can be shown in equation 4.2.

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

Therefore, like the analog square-law detector we are taking peak voltage values, which has an equivalent noise voltage and a mean value of zero, and square them to produce a noise power that is proportional to the square of this amplitude. Figure 4.1 shows the GNURadio blocks used to perform this function.

Like the analog square-law detector, this signal will fluctuate rapidly and to improve the sensitivity of the radiometer we wish to integrate this signal. We now want to look at how we can replicate a RC filter or integrator in the software defined radio.

4.4 Integration

Another step that we typically do in a traditional radiometer is to integrate the signal over time. This gives us an average of the signal and smooths out the output. This also helps to improve the sensitivity of the radiometer by equation 3.7. Chapter 3 section 3.1.2 goes in to more detail on an analog integrator. We now want to see how we can replicate this analog component to the digital domain using a Finite Impulse Response filter.

A Finite Impulse Response (FIR) filter is a digital filter that can take an impulse signal and decays to zero after a finite number of iterations. This type of digital filter can be represented by equation 4.3 which mathematically expresses the FIR Filter and simply says that the nth output is a weighted average of the most recent P inputs.

$$y_n = \sum_{i=o}^{P-1} c_i x_{n-i} \tag{4.3}$$

An Infinite Impulse Response (IIR) filter is the same as the FIR filter, except that we add a summation term which feeds back the previous output. Equation 4.4 shows that a FIR filter is a IIR filter, except that $Q = 0$[Cross (1998)].

$$y_n = \sum_{i=o}^{P-1} c_i x_{n-i} + \sum_{j=1}^{Q} d_j y_{n-j} \tag{4.4}$$

To get a better understanding on how the digital IIR filter relates to the RC filter analog, we can look at the Fourier Transform and the relationship of the input to the output in the frequency domain.

$$H(f) = \frac{\sum_{j=o}^{P-1} c_j e^{-2\pi i j f T}}{1 - \sum_{k=1}^{Q} d_k e^{-2\pi i k f T}} \tag{4.5}$$

In equation 4.5, f is our frequency in Hz and T is the time between samples in seconds and is related to our sampling frequency.

We now want to show the link between our analog RC circuit and the IIR filter. Looking at equation 3.5, which represents the differential equation relating the input voltage V_{in} to the output voltage V_{out}, we can substitute for input and output of our IIR filter. Since we are now in the time domain, we need to define what T is and we can do that using equation 4.6.

$$T = timebetweensamples = \frac{1}{samplingrate} \tag{4.6}$$

We can now relate our input voltage to the input to our IIR filter and the output voltage to the output of our IIR filter.

$$x_n = v_{in}(nT) \tag{4.7}$$

$$y_n = v_{out}(nT) \tag{4.8}$$

We can now rewrite our difference equation with x_n and y_n.

$$\frac{x_n - y_n}{R} = C\frac{y_n - y_{n-1}}{T} \tag{4.9}$$

Finally, we can solve for y_n which results in our final equation for showing how a IIR filter is related to an RC filter.

$$y_n = \frac{T}{T + RC}x_n + \frac{RC}{T + RC}y_{n-1} \tag{4.10}$$

It can be seen that an IIR filter can have the same frequency response as we expect from an analog RC filter. As our sampling rate approaches infinity, the approximation gets closer to the original response from the analog RC circuit.

The RC term gives us our time constant of the circuit and can be used to calculate out our coefficients. We are not concerned about the actual values of R and C with our IIR filter, instead we just need the product of R and C.

In GNURadio most of the work is done for us. We can simply enter in our desired cutoff frequency and GNURadio will calculate our IIR filter coefficients. However, this shows that an IIR filter works much like an analog RC low pass filter.

4.5 System overview

Like a traditional radiometer, the SDR will use an antenna to look at the target of interest. SDRs still use a RF stage that takes the power from the source and amplifies it. The difference though begins after that. A SDR will then sample and generate I and Q values that represents the amplitude and phase of the signal. From there, this data is sent to a computer to be processed. We can then use this information to calculate the power that is being seen. In addition, we can manipulate the signal in other ways such as applying a filter to filter out an unwanted source.

As we have shown the two of the major components of a traditional radiometer, the power detection and integration of the signal can be replicated in software and therefore can be implemented in a software defined radio. The information can now be stored, displayed or both for further analysis.

There is one component of the software defined radio that we are not able to implement in software and that is with the signal amplification. This however does play a major role in the performance of the radiometer and is a key element that should not be overlooked. While this is not implemented in software, it still plays a critical role in our software defined radio radiometer.

Next we will look at the software used in defining the radiometer. It will also be shown what the impact is of this software on the performance of the radiometer.

4.5.1 Control of the SDR Hardware through GNURadio

The N200 sends all data across the 1 Gbps connection to be read in by a host computer running GNURadio. This data is the raw I/Q values that is read by the on board A/D and processed by the on board FPGA. An example of a very simple GNURadio software implementation would simply take this data and store the data to a hard drive in a file. This can be very handy if we want to simply record the data and then process it later. However, depending on the sample rate, it can consume a large amount of storage. A short recording can easily consume 1-2 GB with a sample rate of 10 Msps. It also does not give us any immediate feedback on the radiometer and it does not give us controls of the radiometer such as frequency, integration time or other key variables. Fortunately GNURadio has tools that allows us to build up a very rich application that is able to give us the data we need and control the software defined radio as well.

The GNURadio Companion allows us to create python code that is used to not only receive the data from the SDR but also perform signal processing on the incoming information.

Additional controls are added that allow for tuning of the signal processing parameters and control of the radio functions. This allows us to build up an application that can be run on any computer that is capable of running GNURadio.

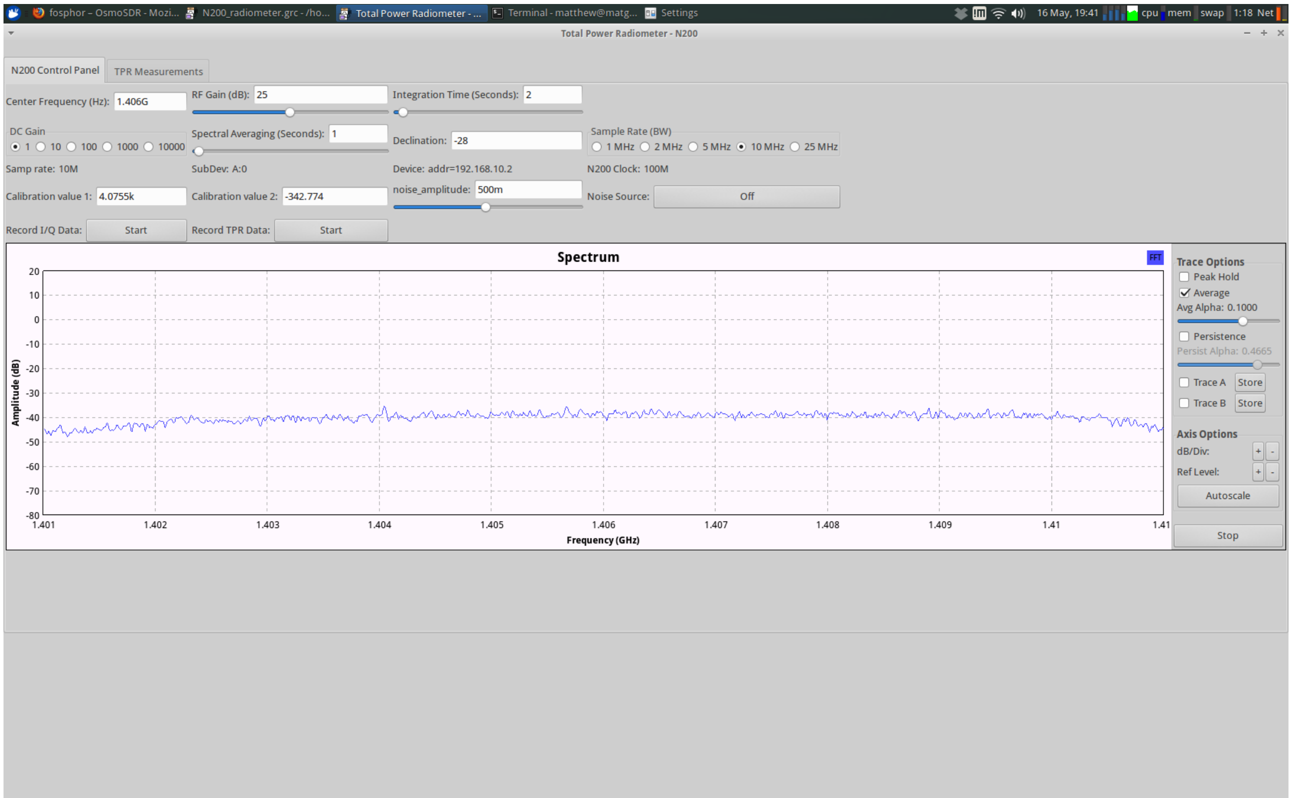

Figure 4.2 A screenshot of the interface made for communication with and controlling the software defined radio

Through this interface we are able to control several key aspects of not only the radio hardware within the SDR but also with the behavior of the radiometer as well. Hardware control of the SDR hardware allows us to change frequency and also adjust gain values within the N200. Bandwidth is another parameter that can be altered from here as well. Bandwidth affects the bandwidth for the RF signal, but also has an impact on the radiometer sensitivity as well. Additional controls allows for alterations to the radiometer and includes being able to adjust the integration time of the radiometer.

4.5.2 Impact of the Controls Related to Radiometry

The controls that have been added for controlling the radiometer can have a large impact on the performance of the radiometer. There is a reason why these controls were added to the GUI for the radiometer and that is they play key roles in how the radiometer performs.

For any radiometer noise temperature is a large consideration and is critical to the design of the radiometer. One method to determine how well a radiometer is to look at the sensitivity of the radiometer. We can do this by looking at the smallest change in temperature the radiometer can see. We will call this the Noise Equivalent ΔT or $NE\Delta T$ of the radiometer and is equation 3.7 covered in chapter 3.

β can be changed by changing the sample rate of the SDR. The sample rate effectively controls the bandwidth in which the SDR is operating at. This also gives us a band-pass filter as well, since the SDR will not respond to frequencies outside of this bandwidth.

τ is the integration time for the radiometer. This parameter is set by the user through the GUI and allows us to change the integration time in seconds.

4.5.3 GNURadio Data Handling

Once we have the data that has been processed by the software defined radio we will want to display this information and be able to store the data so we can analyze it later if needed. Data display can be handled GNURadio where we can plot the total power over time. This allows the user to be able to visualize the total power and be able to determine if the total power has increased or decreased over the time window shown.

Although not usually needed for a total power radiometer we also have the ability to look at the signal in terms of a frequency versus amplitude. This allows looking for any unusual signals that may be interfering with the system or causing erroneous data with our radiometer.

Finally, we will want to store the data so we can do additional analyses on it at a later time. The GNURadio program allows us to store the data in two formats. The first format is storing the raw I/Q data from the radiometer. This format allows us to playback the data through GNURadio at a later time. This can be useful for if we wish to change parameters in GNURadio such as bandwidth or integration time. It is also a good diagnostic tool as we can check that the signal coming in is clean or if we need to apply additional filters to remove an unwanted signal. This file may be quite large, consuming several gigabytes of data for a 20 MHz wide signal in a matter of minutes of record time.

The second format is the total power that has been calculated by the radiometer. This file is much smaller since much of the signal information has now been reduced to simple power versus time information. This allows for easy manipulation through any math program such as Matlab for analysis.

4.5.4 GNURadio Data Display

The information from the software defined radio can be displayed through GNURadio to show a number of things. Since we have both frequency and magnitude information we can display this information. We are able to also display the information that shows the total power that is being seen by the radiometer as well.

We are not limited to just total power from the radiometer. If the radiometer has been calibrated, those calibration points can be entered and GNURadio can calculate the calibrated noise temperature. Additional information may also be added as needed. For example, we are able to view the full spectrum that the radiometer sees. This can be a useful tool for looking at potential RFI issues.

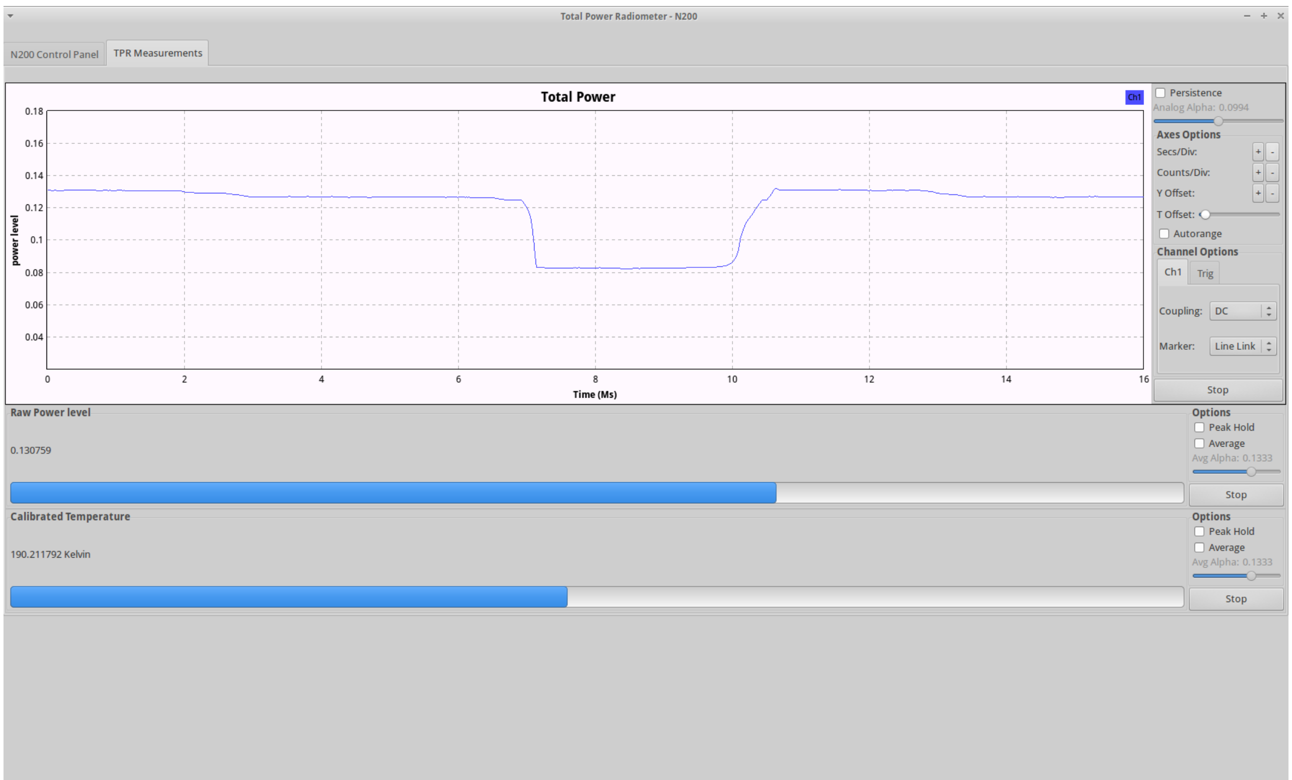

Figure 4.3 A screenshot showing the ticker tape display for the total power readings. In addition, raw and calibrated noise temperature is shown below.

CHAPTER 5. EVALUATION SETUP AND EXPERIMENTAL DESIGN

The experiments outlined in this chapter is to demonstrate and verify that a software defined radiometer is able to perform as well or better than a traditional radiometer. In addition, it will be demonstrated that a software defined radiometer can add additional functionality not typically found in most radiometers.

In this chapter we will look at four experiments. Experiment one verifies the operation of a software defined radiometer by comparing typical operations to a square-law detector, a typical device used in radiometry and calibration of the radiometer. Experiment two verifies the expected sensitivity and stability of a software defined radiometer. Experiment three will demonstrate a software defined radiometer mitigating an interfering signal and compare the results to a traditional radiometer with no interfering signal mitigation. Finally, Experiment four will examine the affect filtering and bandwidth of the signal affects performance.

5.1 Experiment I - Software Defined Radiometer Verification and Calibration

In this experiment we will verify the operation of a software defined radiometer. This experiment will verify that a software defined radiometer is able to perform as expected and will compare our results to a 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 software defined radio. This signal is split using a power divider so that the information will be the same to both devices with the except for the 3 dB plus insertion loss the power divider adds. This allows us to verify the software defined radio with a proven system.

5.1.1 Data Collection

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

Software Defined Radio Data. 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. In our case we will use a file sink to output the data to a file. The flow of data to this sink is controlled by a valve block. This allows for the user to turn on and off recording of the data. A block diagram of the file blocks is in figure 5.1.

Figure 5.1 The File Sink block used in GNURadio. Source: GNURadio

There are two types of files that the SDR generates. The first type of file is the complete I and Q data points, which represents the phase and magnitude of the signal, recorded from the SDR. This file is stored as a little-indian format and as complex values. Due to the sampling rate, it is not uncommon for this file to grow to be quite large, usually several gigabytes of data for a 10-15 minute run. However, this file can then be feed 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 block diagram of this block can be found in Appendix A in figure A.1 and the source code for this can also be found in Appendix A. This file is also a little-endian format however this file only has float values. This file is also much smaller than the file that contains the I and Q data points. A typical file size is 50 - 100 kB for a 10 to 15 minute run.

Square-law detector data. The square-law detector outputs information as an analog voltage that is linearly proportional to the RF power measured. This required a system that can capture the analog signal from the ADL5902 and then send the data to be stored. The National Instruments USB-6009 data acquisition 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 up to 48 ksps with a resolution of 14-bits which is more than adequate for the experiments. 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 the information from the USB-6009 and stored the data in both Labview's binary format and in a more human friendly ASCII format. The USB-6009 then connects to a host computer through the USB interface. This made obtaining the data and using device straightforward to use.

The Labview program created generates a user interface to display the data and allow for turning on and off the recording. This allowed for easy monitoring of the square-law data while the experiment was being run. A screen shot of the GUI created in Labview is shown in figure 5.2.

This program uses the National Instruments DAQ assistant which allows for quick configuration and setup for the computer to talk to a number of NI devices. Labview also includes blocks that allows us to easily record the data to a file and also use a low pass filter. These blocks made up most of the program and resulted in a program that was quickly made. Figure 5.3 shows the blocks used and the wiring of the blocks.

5.1.2 Experimental setup

For this experiment, the following hardware was used:

1. N200 Software Defined Radio with DBSRX2 Daughter-board
2. ADL5902 Square-law detector
3. National Instruments USB-6009 Data Acquisition Unit
4. ZN2PD-20-S+ Power Divider
5. 50-ohm matched load

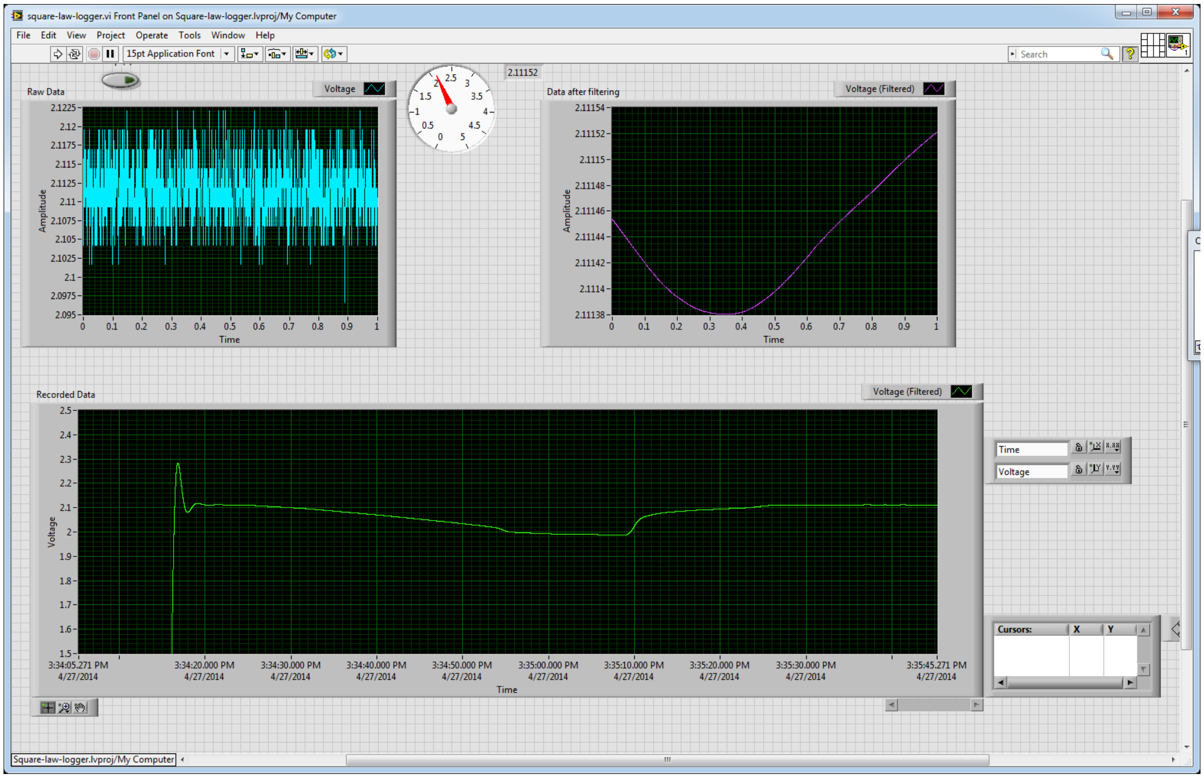

Figure 5.2 A screenshot of the Labview GUI interface. Source: Labview

6. Rigol DP832 Power Supply

7. ISU Radiometer RF Front End

The ISU Radiometer, as shown in figure 5.4 was used as it has the necessary Low Noise Amplifiers (LNAs) to provide the needed amplification. The only item powered for these experiments was the LNAs, all other components are passive or were deactivated. The ISU Radiometer uses the following hardware:

1. 4 x Integrated Microwave Bandpass filters (1400 - 1425 MHz)

2. 2 x Miteq AMF-3F-01400147-30-10P LNA

3. 1 x Miteq AMF-2F-01400147-04-10P LNA

Figure 5.5 shows a block diagram of the experimental setup. A matched load is used to simulate our antenna or source. This matched load is then submerged in temperature baths.

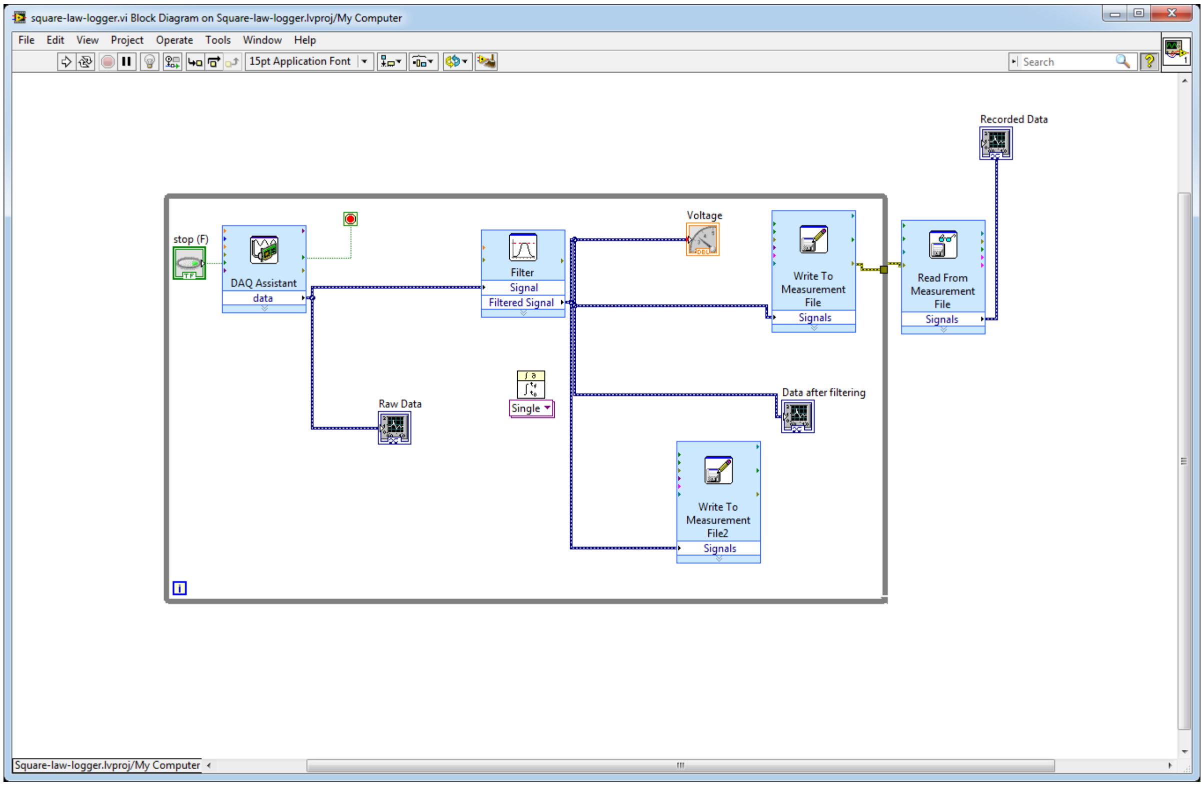

Figure 5.3 A screenshot of the Labview block diagram. Source: Labview

These baths were temperature controlled first by using 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 could easily be monitored and maintained. The load was submersed in each bath for a minimum of 2 minutes to allow for 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 ISU radiometer provides the amplification needed for our experiments and finally we divide that signal between the ADL5902 square-law detector and the N200 Software defined radio. The N200 is then connected to a personal computer running XUbuntu Linux and GNURadio.

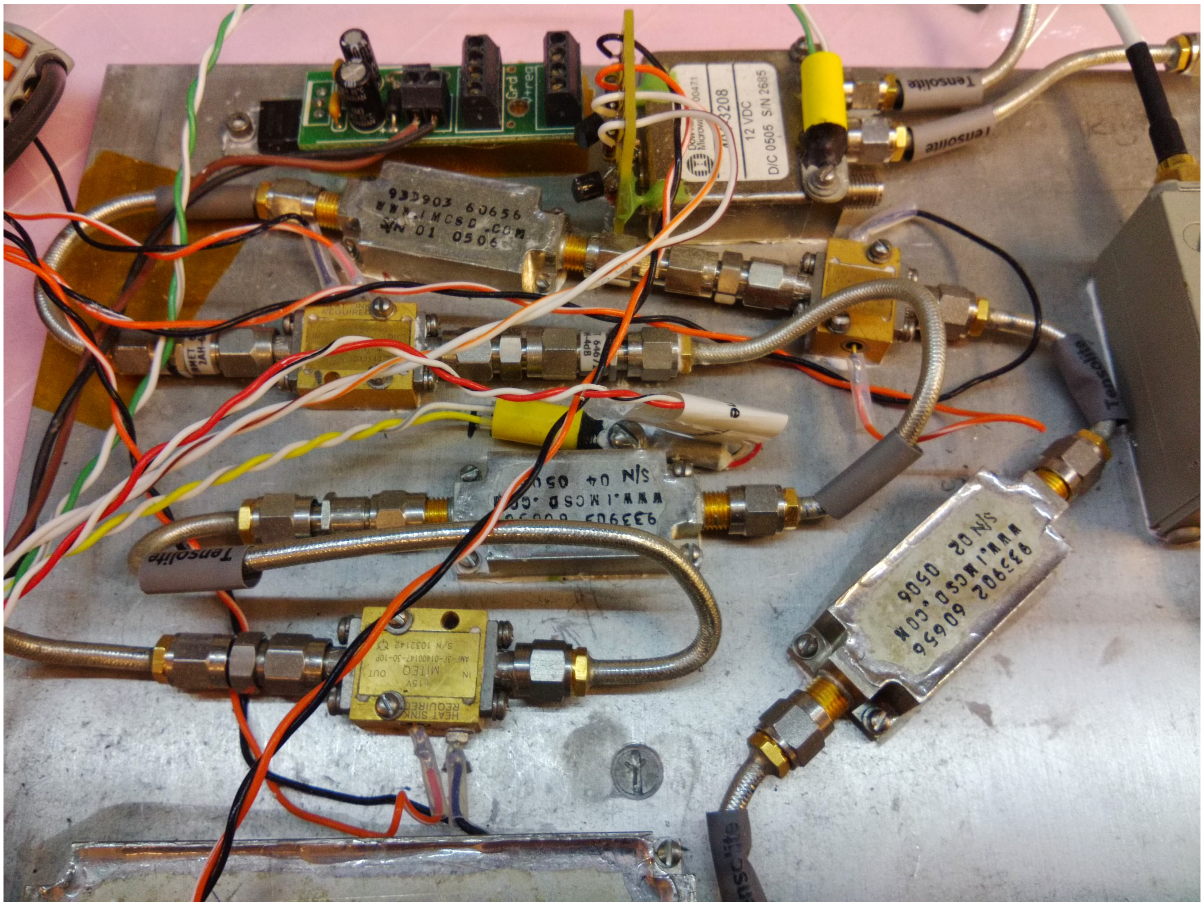

Figure 5.4 The ISU radiometer's LNAs and bandpass filters used in most experiments. Source: Matthew E. Nelson

5.2 Experiment II - Verification of sensitivity and stability

In this experiment we will verify the sensitivity and stability of a software defined radiometer. This experiment will verify that a software defined radiometer is able to meet the expected sensitivity and stability within an acceptable range.

5.2.1 Data Collection

The data collected for this experiment was the total power measurements made from the software defined radiometer. These measurements uses the same method as outlined in section 5.1.1.

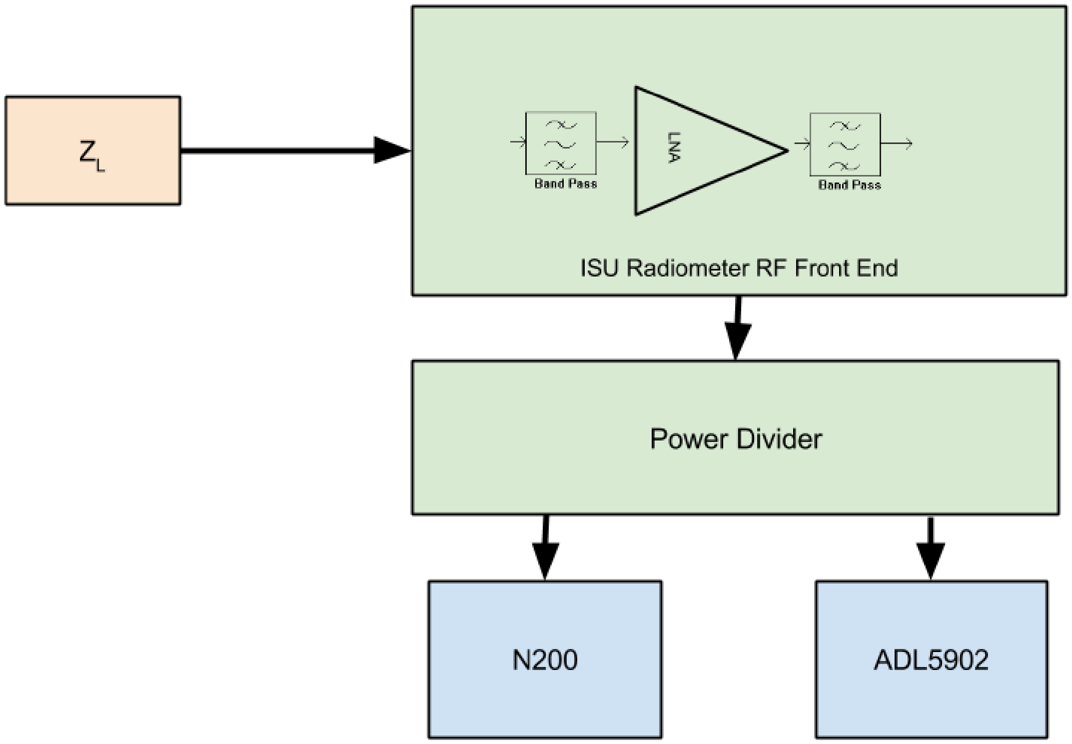

Figure 5.5 Block diagram of Experiment 1 setup. Source: Matthew E. Nelson

5.2.2 Experimental setup

The experimental setup used for experiment two is the same experiment as outlined in section 5.1.2.

5.3 Experiment III - Interfering Signal Mitigation

In this experiment we will generate an interfering signal and then mitigate the signal using a software defined filter. A square-law detector will be hooked up in parallel to measure the same signal but will have no mitigation methods for the signal. We will then compare the two signals to verify that the software defined radiometer can mitigate an interfering signal and still make total power measurements.

The addition of an unwanted signal can be a determinant to the radiometer and has an adverse effect on how the radiometer operates. In today's world though, it is getting more difficult to control intentional radiators as the RF spectrum becomes crowded with more devices [Ellingson et al. (2003)]. This problem becomes even a greater issue with radiometers used in orbiting spacecrafts as they are able to see large areas[Misra et al. (2012)]. Even though the band we are working in of 1.4 GHz is an internationally protected frequency, there can still be both intentional and unintentional radiators that cause interference.

RFI detection and mitigation is not a new topic in radiometry and there have been other methods in both the detection and mitigation of these signals. [Forte et al. (2013)][McIntyre and Gasiewski (2012)][De Roo et al. (2007)].

This test was designed to test whether or not a software defined radiometer would be able to cope with an interfering signal. This test injected 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 in 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 this offending signal, a filter is designed to remove the offending signal. An initial design of the filter was designed using a program that is part of the GNURadio software package, GNU Radio Filter Design tool. Figure 5.6 shows a screen shot of the 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 5.6 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 software designed radiometer.

5.3.1 Data Collection

The data collected for this experiment includes both the total power measurements from the software defined radiometer and the square-law data. Section 5.1.1 explains in detail the setup and configuration of the equipment used to collect this data.

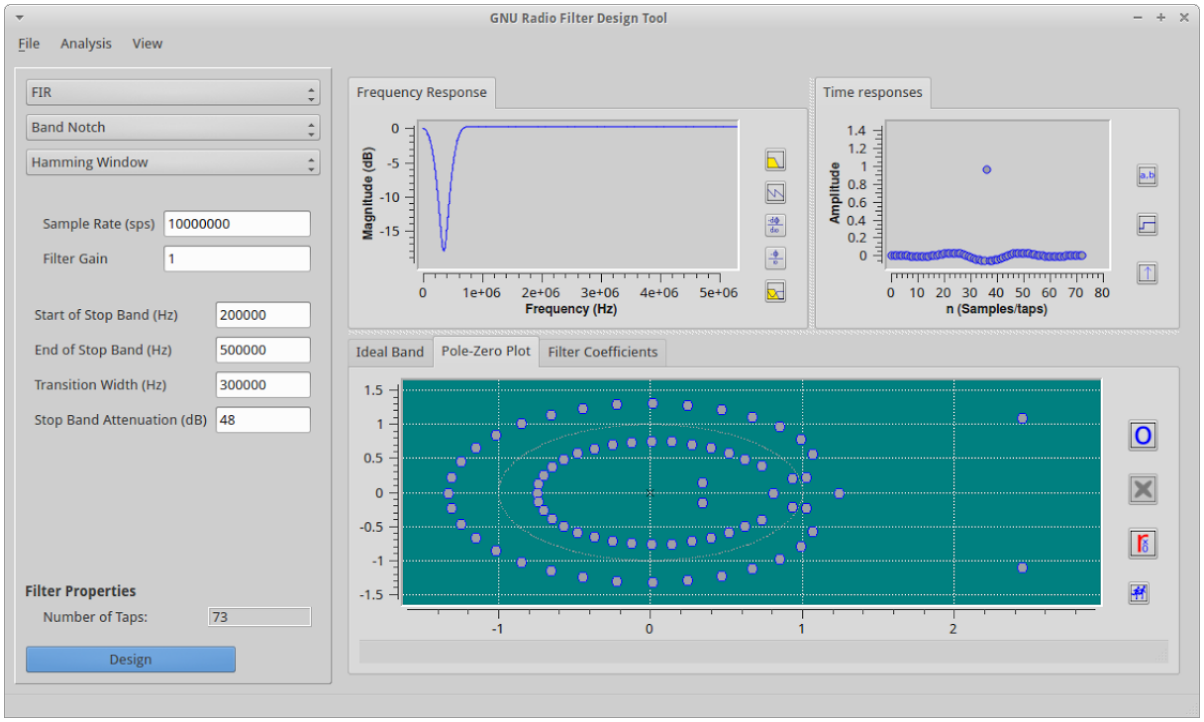

Figure 5.6 Image of the GNU Radio Filter Design tool

5.3.2 Experimental setup

The experimental setup for this experiment is similar to the setup outline in section 5.1. In addition to the setup outline in that section, another software defined radio was added and injected into the RF signal chain. This software defined radio is then configured to operate like a signal generator and generates the offending signal.

Our signal generator is the HackRF One shown in figure 5.7. This SDR is a cheaper SDR and thus has lower specifications than the Ettus Research N200 used as the software defined radiometer. However, for the task of a signal source, it will suit our needs.

For this experiment, the HackRF will generate a sinusoidal wave at a fixed frequency and with the amplitude changed at different times during the experiment. This is controlled from a program called *osmocom_siggen*. Osmocom was originally developed to communicate with OsmocomSDR hardware. However, it has expanded to include the HackRF and Ettus Research hardware. The *osmocom_siggen* program provides us a GUI interface to set the frequency and

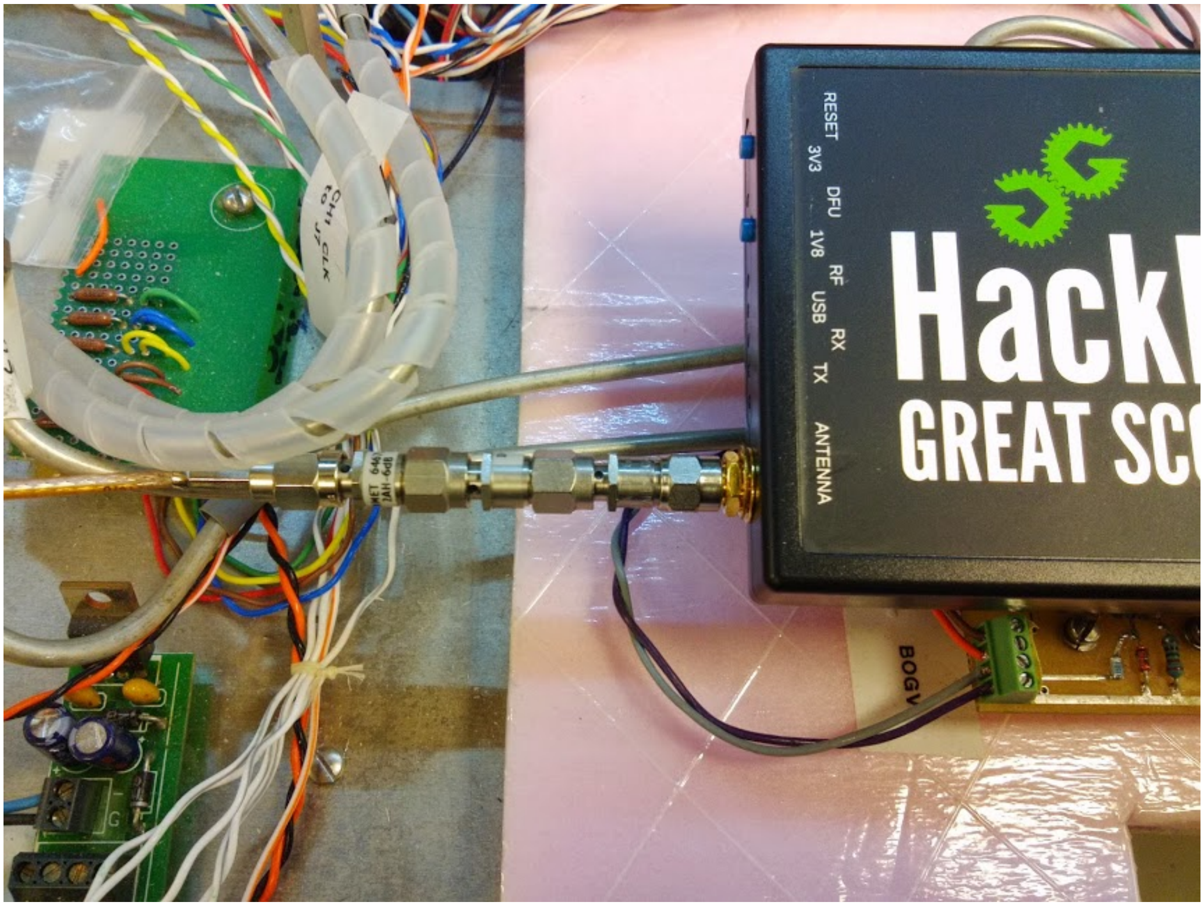

Figure 5.7 Image of the HackRF used to generate the offending signal

amplitude as well as the type of signal generated.

5.4 Experiment IV - Performance impact of interfering signal mitigation

In this experiment we will examine the impact of filtering an interfering signal on the performance of the software defined radiometer. We will specifically look at the $NE\Delta T$ of the signal and also how reducing our overall bandwidth affects our total power received to the radiometer.

5.4.1 Data Collection

The data collected for this experiment is the total power reading measurements obtained from the software defined radio. This data is identical to the method used in section 5.1.1.

5.4.2 Experimental setup

In this experiment we setup our experiment as outlined in section 5.1.2. The rest of the data collected is done by the analysis of the data collected.

CHAPTER 6. RESULTS AND ANALYSIS

This chapter will display the results obtained from the experiments outlined in Chapter 5. We will then give an analysis of the data including data interpretation and the information learned from each experiment. Next we will do an analysis of a traditional radiometer vs our software defined radiometer in terms of price, weight, and quality. Finally we will summarize our results and analysis from this chapter.

6.1 Experiment I - Software Defined Radiometer Verification and Calibration

As outlined in Chapter 5, Section 5.1, this experiment is to verify the operation of a software defined radiometer. This is done by performing experiments that are similar to verification and calibration methods used on a traditional radiometer. By comparing these results to a square-law detector, receiving the same signal, we can verify the operation of the software defined radiometer.

6.1.1 Data Collected

The data collected for experiment one is the total power measurements collected from the software defined radiometer and the voltage measurements, which represent the total power measurements, from the square-law detector. This data was then calibrated by manually recording the rQ values and voltage values when the matched load is submerged into either an ice water bath or liquid nitrogen (LN2) bath. Table 6.1 shows the values collected during the experiment. These data points were then used to generate the data analysis in the proceeding section.

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

In our results, we used iPython Notebook to read 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. Figure 6.1 shows a screen shot of a iPython notebook used in this thesis

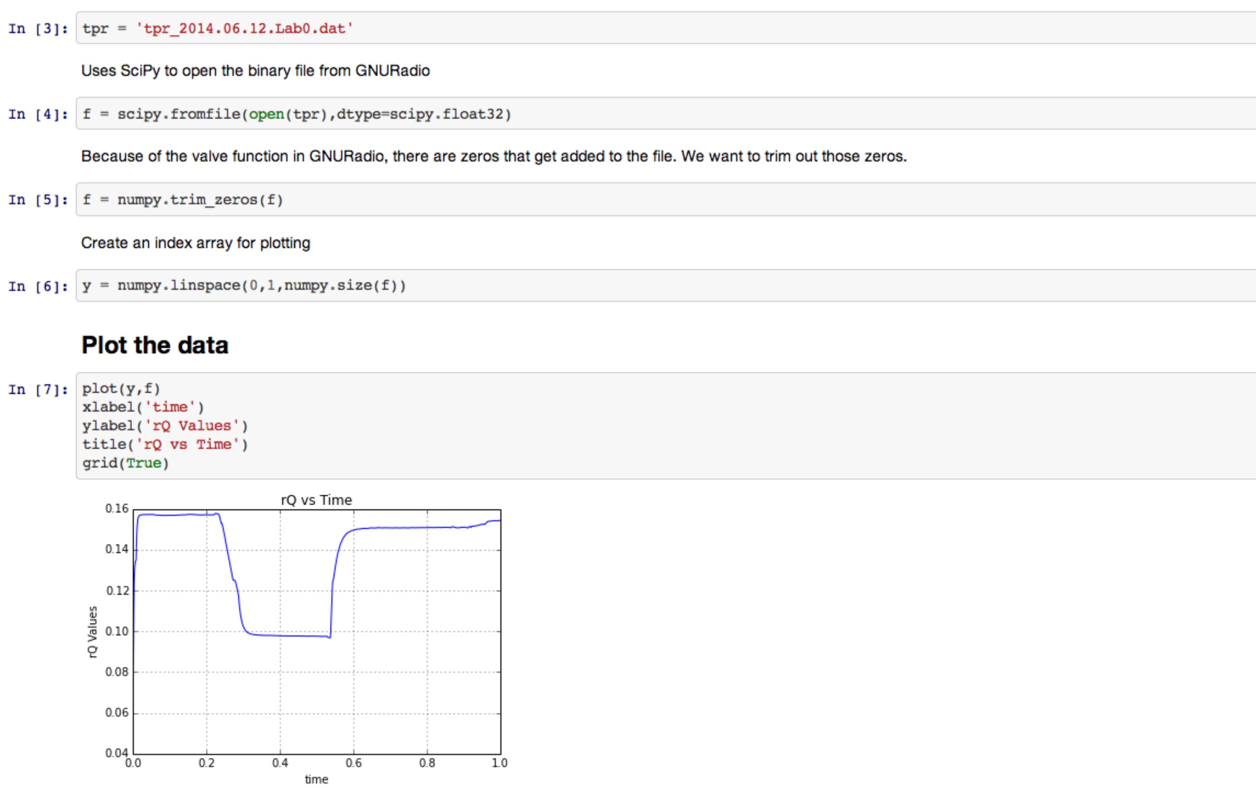

Figure 6.1 A screenshot showing the iPython notebook code and related graphs generated for parsing GNURadio data

The first data we will look at with be with the software defined radio. The SDR records the total power measurements to a binary file that either Matlab or Python can read. This file format is explained in section 5.1.1. Additional information on data analysis is explained in section 6.1.2. We will begin by looking at the raw total power readings which we will call raw Q values or rQ values. These are the uncalibrated total power readings from the software defined radio.

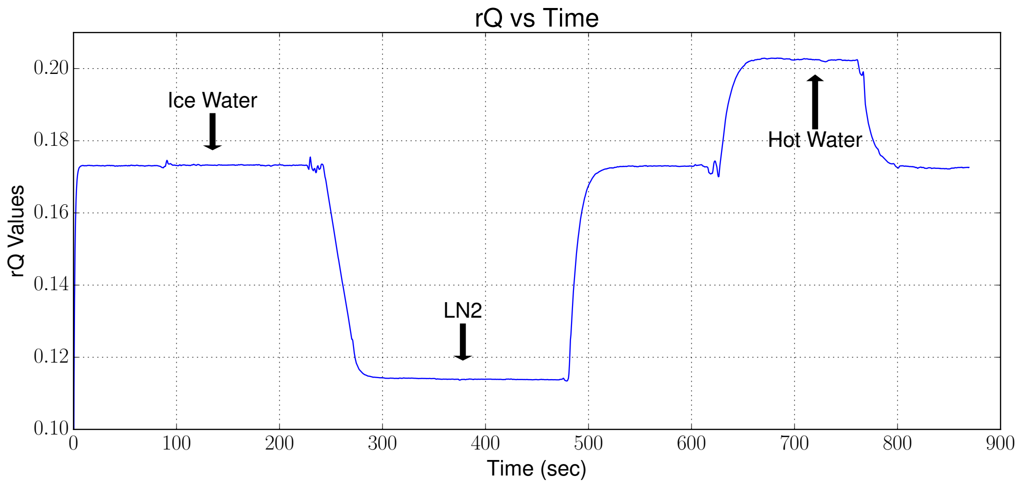

Figure 6.2 Graph of the uncalibrated rQ values of Experiment I

Figure 6.2 shows the total power reading versus time and is also marked when the matched load was submerged in Ice Water, LN2, or a hot water bath. Since we know what the temperatures are for the ice water and LN2, we can now calibrate these readings to a noise temperature reading. This is done by reading in a calibration file we have stored in csv format and performing linear algebra to solve the slope of the line. This was done in our iPython Notebook using the following code.

```
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)
```

Now that we have our calibration points, we can now re-graph this data but now calibrated as a reference noise temperature. Figure 6.3 shows a calibrated graph of the data in relation to noise temperature. In addition, we have colorized this graph to show warmer to cooler temps. This is helpful as we often refer to these as noise temperatures.

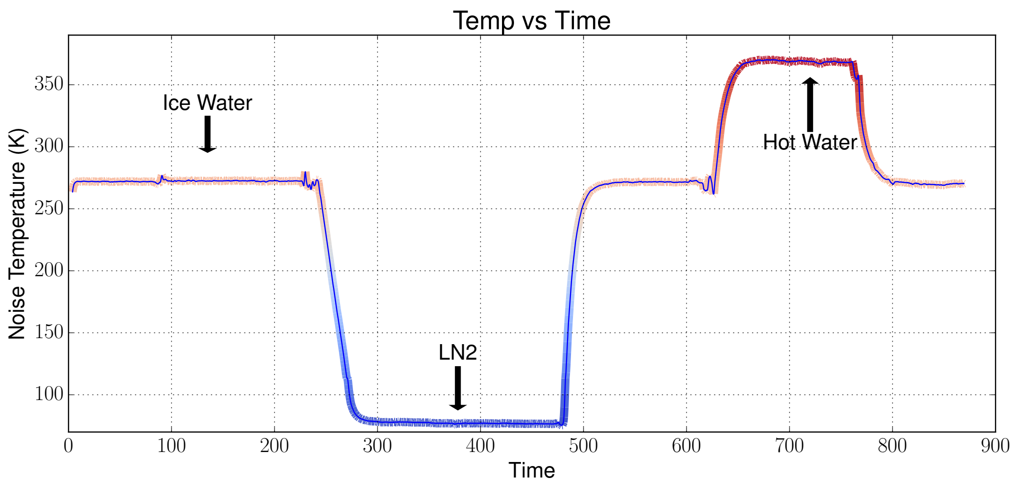

Figure 6.3 Graph of the calibrated noise temperature of Experiment I

Now that we have looked at the software defined radio data, we want to look at the square-law detector with the end result of comparing the two. The square-law detector gives us power information as a voltage, so once again we will need to calibrate this to the known temperature references. However, before that we need to do one other step with the data. Unlike the software defined radio, there is no filter or integrator to help smooth out the data, so the square-law data is very noisy. Our first step will be to filter the data. Figure 6.4 shows our raw data that we get from the square-law detector.

Once again we can use Python to process this information and specifically we can use SciPy which includes several useful signal processing modules. For our use, we will use a low pass filter to clean up the signal. The following code allows us to do just that.

```
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)
```

Figure 6.5 now shows our data that has been filtered by the low pass filter. This is similar

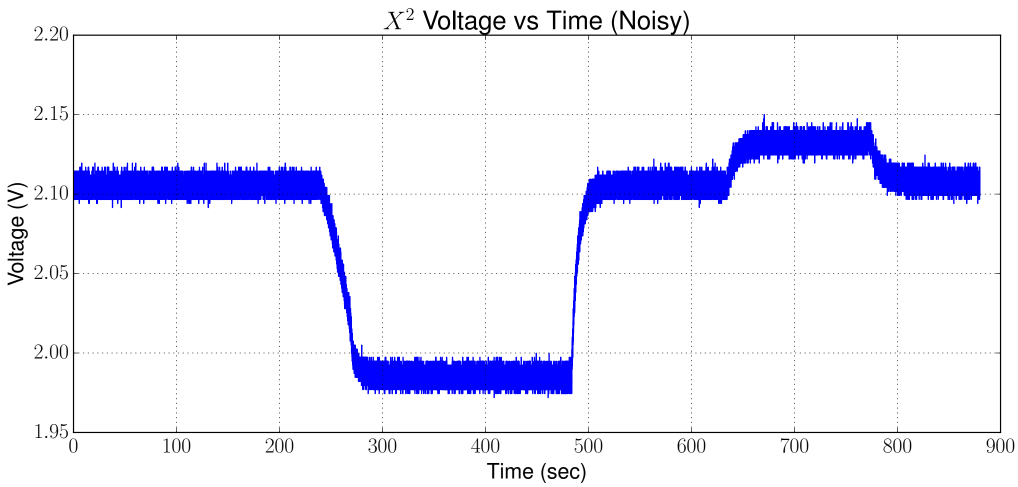

Figure 6.4 Raw and noisy graph from the square-law detector used in Experiment I

to the low pass filter that is also used by the software defined radio.

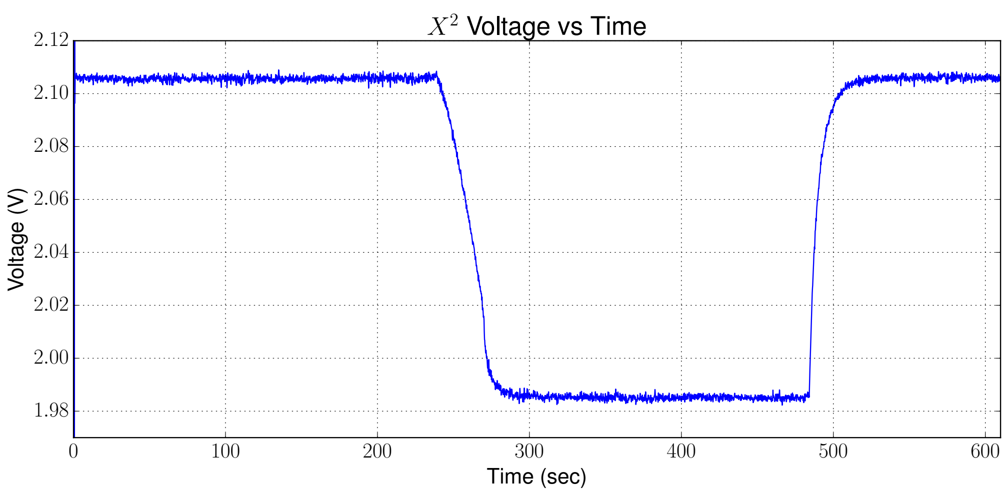

Figure 6.5 Filtered data from the square-law detector used in Experiment I

Using the same technique as earlier, we can now calibrate the raw voltages from the square-law detector to the noise temperature. Like the software defined radio data, we can calibrate the voltages from the square-law detector to the physical temperature that our matched load is placed in. Figure 6.6 shows the data from the square-law detector calibrated to our known temperature points. This now allows us to directly compare the square-law detector to the software defined radio data since we have a common reference point in which to compare the data.

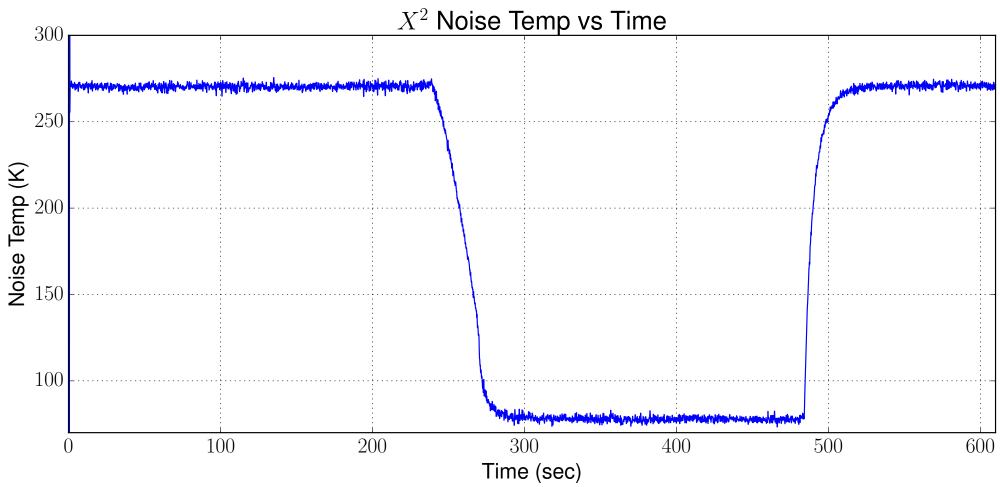

Figure 6.6 Calibrated data from the square-law detector used in Experiment I

We now want to compare both the Software Defined Radio and the square-law to make sure that they match up. Since both the SDR and the square-law are now calibrated to a noise temperature, we can easily graph both of the data and compare to see how well they match up.

We can see in Figure 6.7 that both the software defined radio and the square-law detector match up very nicely. 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.

For the data analysis in this experiment, the data collected from the software defined radiometer was compared with a known method of collecting total power readings by using a square-law detector. The analysis of this data showed that the calibrated information from both the software defined radiometer and the square-law detector matched up. This proved that the software defined radiometer was indeed performing as expected for a radiometer. It

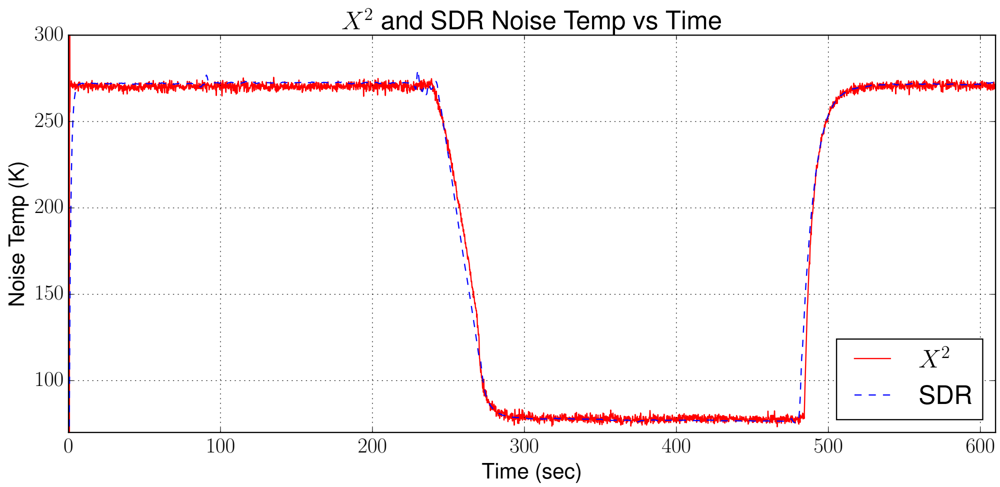

Figure 6.7 Figure showing both the SDR and square-law noise temperature data in Experiment I

also demonstrated that calibration is possible using two reference temperatures.

Further analysis was then conducted to compare how a software defined radiometer might be used in a typical application, such as soil moisture measurements. This is covered in the next section.

6.1.3 Application with Soil Moisture Readings

A common application of radiometers is in the measurement of soil moisture. All items 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, but the amount that reaches the antenna varies by the amount of moisture in the soil. If we can calibrate the radiometer to two known soil conditions, then we can measure the various levels of soil moisture in the soil. At this time, we will simply 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 Software Defined Radio, we can set up two methods to visually look at the noise temperature and thus the soil moisture percentage. Since we do not have an antenna hooked up, we will simulate this by using two reference temperatures. In this case the ice water bath and the LN2 that we just used and was shown earlier.

A unique visual aid we can use with GNURadio is a waterfall display. This display gives us information that includes frequency, amplitude and time. It is referred to as a waterfall display due to the fact the display continually moves from top to bottom and looks like a waterfall. Amplitude information is given by mapping the range of amplitudes to a color bar. Frequency is given on the x-axis and time is on the y-axis. Figure 6.8 shows a screen shot of the waterfall display in the GUI created in GNURadio Companion for this experiment. This is the same program we have used but with the waterfall display now added. The data for the waterfall is pulled directly from our source block or in our case the N200 software defined radio.

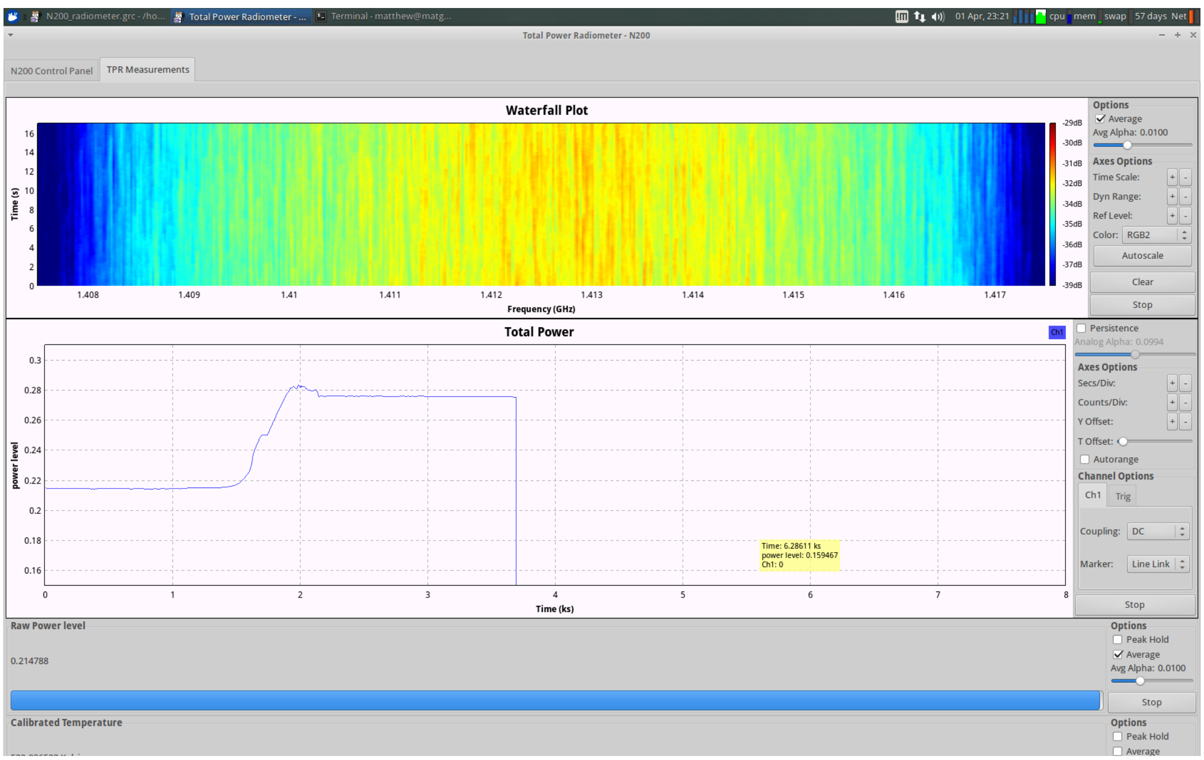

Figure 6.8 Screenshot of the waterfall display used in Experiment I

Let's take a look at two screen shots of the waterfall. We will put them side by side so we can better compare the display when the thermal load is in the ice water bath and when the load is in LN2.

In figure 6.9 we can see that the ice water screen shot appears warmer than the right side

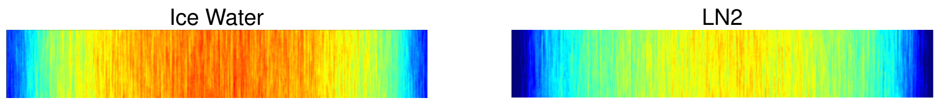

Figure 6.9 Side by side comparison of the waterfall display for Experiment 1

screen shot which appears cooler. There is a limitation in GNURadio and the waterfall display that limits what the range for the power readings are. The overall power that we see only changes by about 3 dB and our current range in the waterfall display is set to 10 dB. If we could reduce the range, the color change would be even greater and more pronounced. However, this does show that a change can be seen visually with color to indicate the noise temperature.

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 6.10 shows the same data we looked at earlier but we have now added a scale to show soil moisture as a percentage.

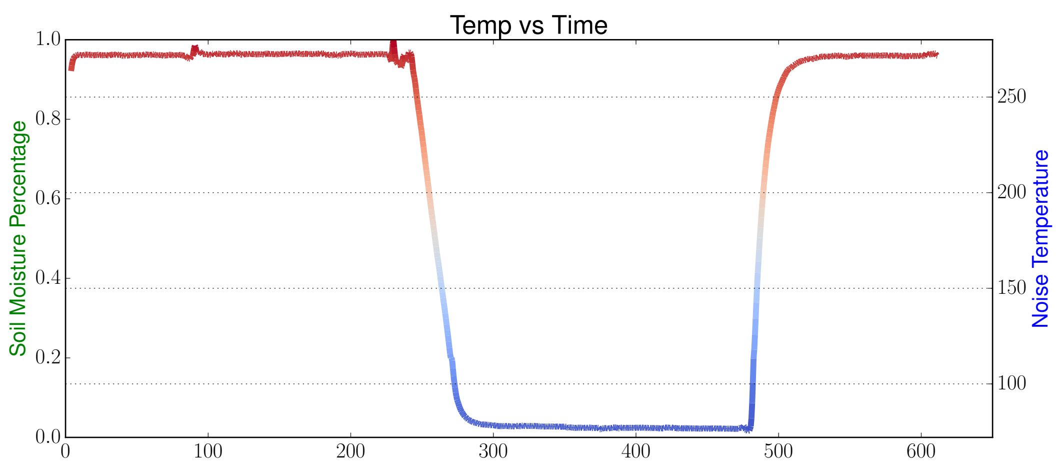

Figure 6.10 Plot of the noise temperature of Experiment 1 with soil moisture percentage

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).

6.2 Experiment II - Verification of sensitivity and stability

6.2.1 Data Collected

For the sensitivity portion of this experiment, the same data collected from section 6.1.1 is used. The data collected for the stability is the total power data collected from the software defined radiometer. For calibration the data points shown in table 6.2 was collected and used to calibrate the total power measurements.

Table 6.2 Total Power calibration data points for the stability experiment

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

6.2.2 Data Analysis

Sensitivity. Sensitivity for a radiometer is the $NE\Delta T$ that was covered in chapter 3 and specifically can be found in equation 3.7. This $NE\Delta T$ is also the standard deviation of the data that we collect from the software defined radiometer.

The standard deviation is calculated to be 0.09 Kelvin by using the standard deviation function found in NumPy. We can compare this to what we expected the $NE\Delta T$ to be which uses equation 3.7. To calculate the $NE\Delta T$, we need to know the integration time, the bandwidth, the total system and antenna noise. This information is found in table 6.3.

Table 6.3 Experimental parameters for experiment one.

Bandwidth (β)	**Integration Time** (τ)	$T_A + T_{sys}$
10 MHz	2 sec	437 K

Python was then used to calculate the $NE\Delta T$ using the data from table 6.3. The python code listed below is used to calculate the expected $NE\Delta T$.

```
tau = 2
BSDR = 10e6
Tsys = 350
TA = 77
NEAT_SDR = (TA+Tsys)/sqrt(BSDR*tau)
```

This gives us the result of 0.1 Kelvin for the expected $NE\Delta T$. We can now look at the data collected from experiment one. 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. To determine our sensitivity, we will zoom in on data that was collected while the matched load was submerged in LN2. Figure 6.11 shows this graph.

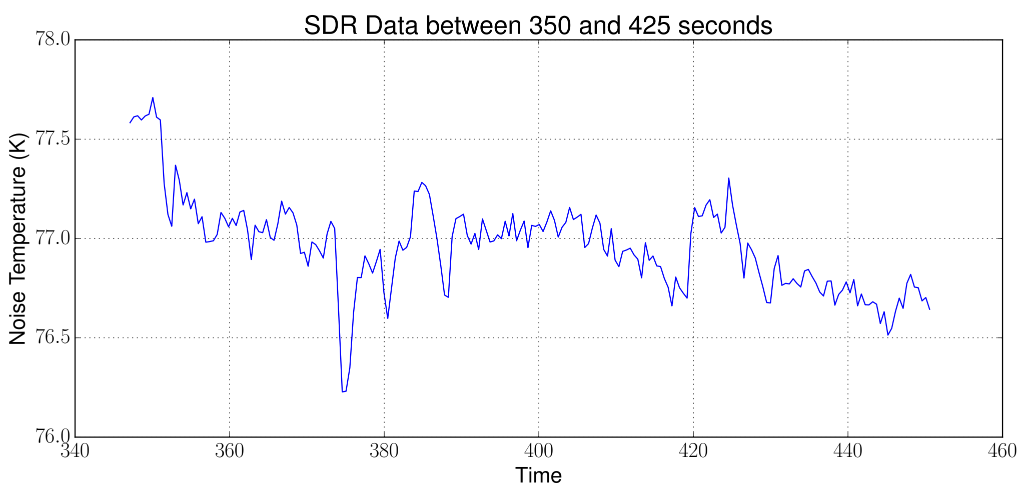

Figure 6.11 Graph of the calibrated total power from experiment one while submerged in LN2.

Using Python we can determine the standard deviation of this range of data. Using Python, we find that the standard deviation for this section of data is .23 Kelvin. We can now plot both the expected sensitivity and the actual sensitivity which is shown in figure 6.12.

While .23 Kelvin is higher 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

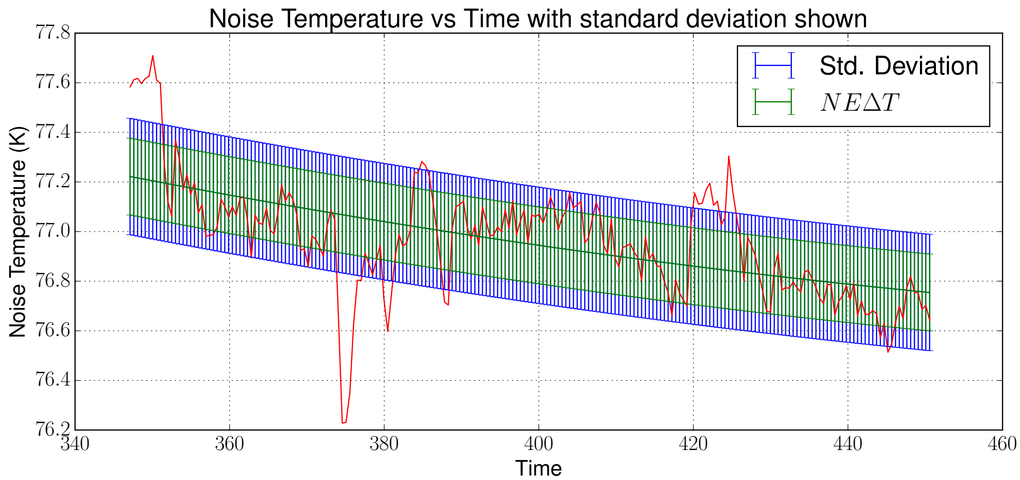

Figure 6.12 Graph of the calibrated total power with expected and actual sensitivity.

our actual performance of .23 Kelvin still meets our requirements for a radiometer.

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 looked at to study the trend of the data. The data is graphed in figure 6.13.

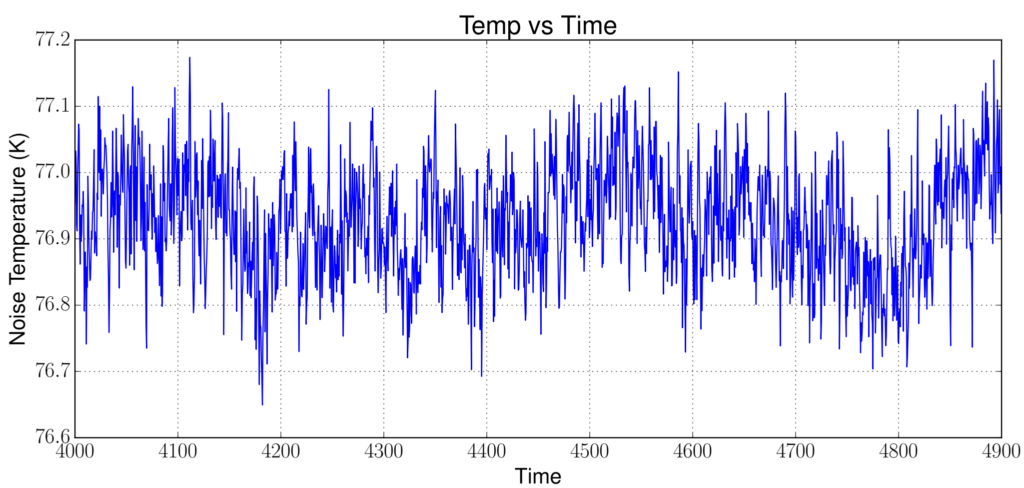

Figure 6.13 Graph of the calibrated total power over a period of fifteen minutes.

Once again we can look at the standard deviation to see how much of a change occurs over this time period.

Figure 6.14 now shows the stability plot with both the actual $NE\Delta T$ which is the standard deviation of the graph, and the calculated $NE\Delta T$.

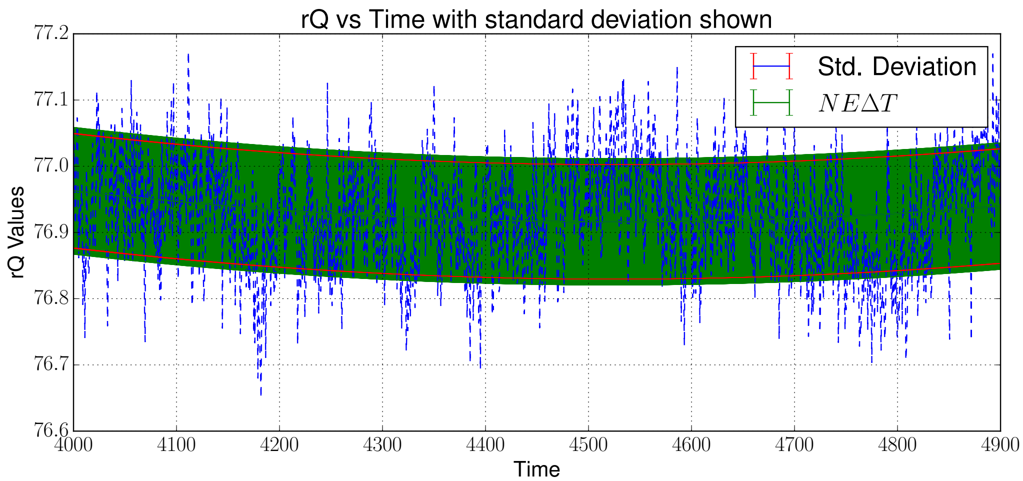

Figure 6.14 Graph of the calibrated total power with the standard deviation plotted.

6.3 Experiment III - Interfering Signal Mitigation

6.3.1 Data Collected

The data collected for experiment three is the total power values from the software defined radiometer and also the voltage data from the square-law detector. These values are calibrated using the data points provided in table 6.4.

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 the what happens to our total power readings with no filter applied. As we stated in Chapter 5, section 5.3.2, the frequency of the offending signal will not change, but the amplitude will. This will mean that we should see clear indications of the total power changing as the amplitude of the offending signal changes.

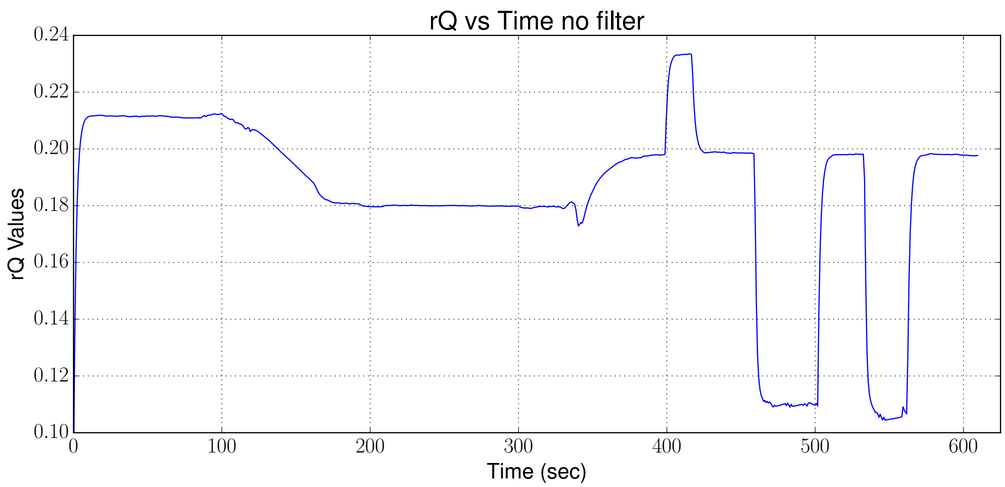

Figure 6.15 Graph showing the unfiltered total power measurements on the software defined radio

We can see that there are pulses that occur in the graph in figure 6.15 and that changes in the amplitude of the offending signal affect our total power readings. These same pulses can also be seen in the square-law detector data as well which can be seen in figure 6.16.

We can clearly 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 on the software defined radio, we can see the signal in questions which is at 1.405 GHz. Figure 6.17 shows us what the software defined radio sees which is a spike at 1.405 GHz. The square-law detector of course has no frequency information, so our only method to detect an interfering signal is by looking at the total power readings. In our case there is elevated readings and we can see the spikes in the square-law data. However, we do not know where in the spectrum the signal is at.

Since we know where the offending signal is, we can now design our filter to filter out this

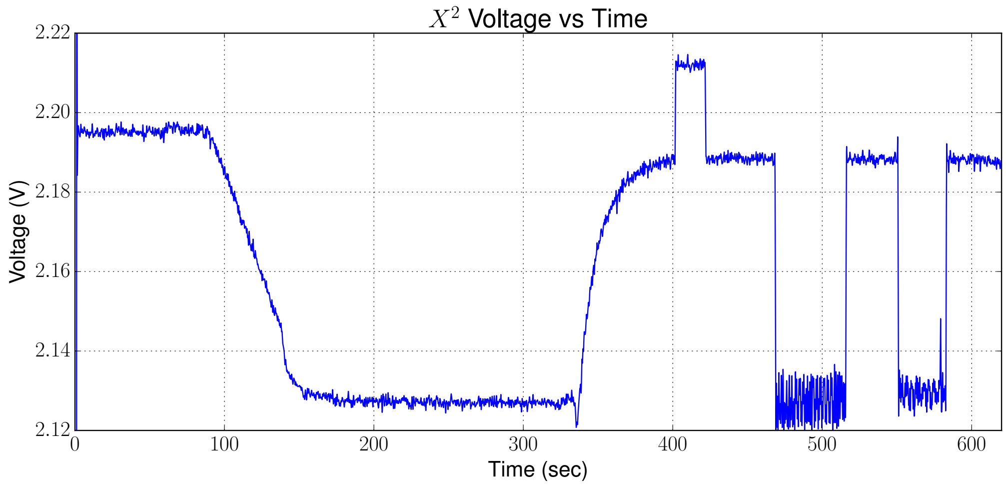

Figure 6.16 Graph showing the raw total power read from the square-law detector with an interfering signal.

signal. In our GNURadio program we can specify both the frequency and the bandwidth our band-reject filter that we desire. Ideally we want to keep the bandwidth of the filter as tight as possible to the offending signal but we also want to make sure our filter is effective in removing the signal. Figure 6.18 shows the spectrum display from the software defined radio with the filter now turned on and filtering the offending signal.

Since we have now removed the offending signal, we will want to re-run our experiment and once again compare the difference between the software defined radio and the square law detector. We can begin by looking at the software defined radio total power readings. In figure 6.19 we can see a calibrated graph of the noise temperature seen by the software defined radio. This graph is very similar to the graphs we expect from our total power radiometer. However, we want to compare this to our square-law detector as well.

Figure 6.20 now 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 the changes in amplitude which would make both calibration and obtaining useful data difficult.

Filter delay. It should be noted that when doing data analysis with this data, that there

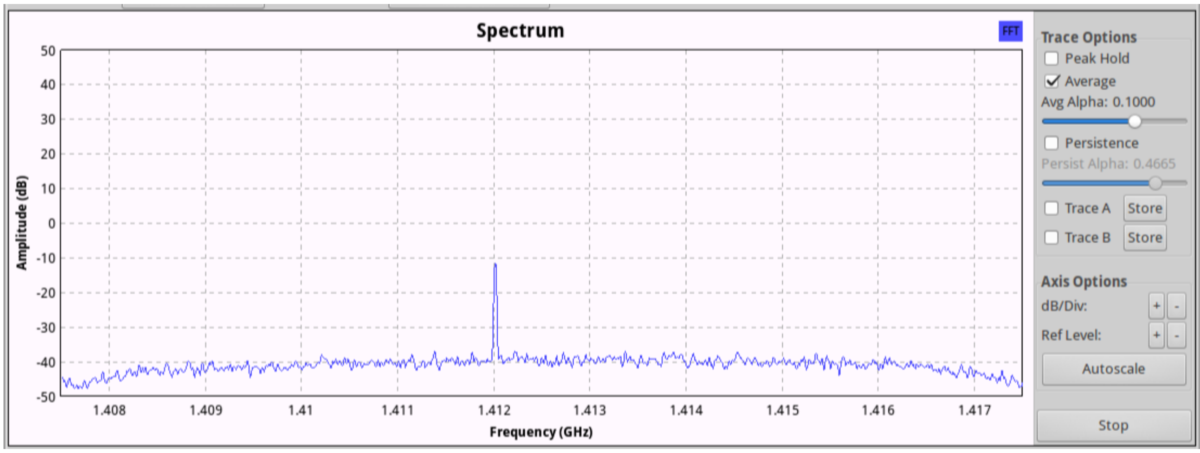

Figure 6.17 Image showing the spectrum view from the software defined radio

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 power 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 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 taps generated was 18,181. Taking that and our sampling rate into account, our FIR filter only delays the signal by 9 milliseconds.

A final note on lining up the square-law detector information 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.

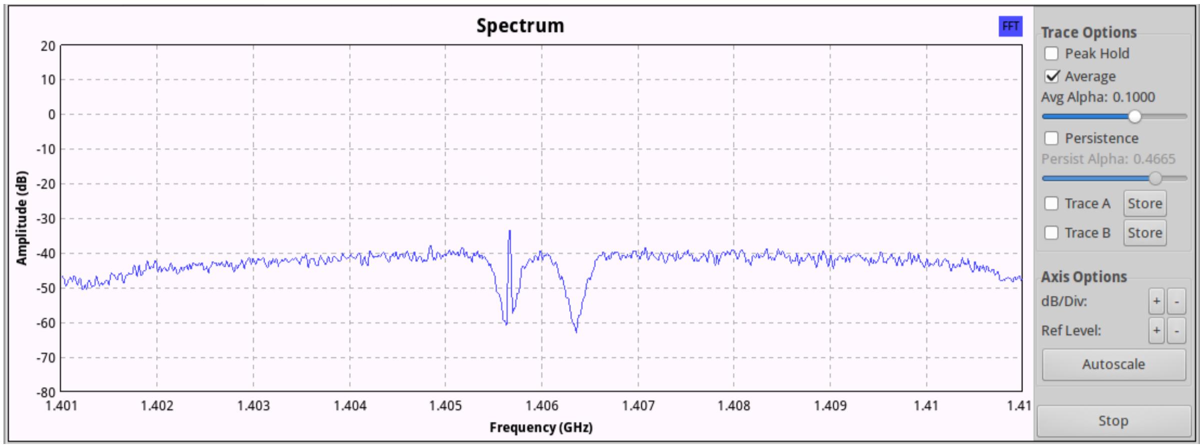

Figure 6.18 Image showing the software defined radio with the filter on and filtering the offending signal

6.4 Experiment IV - Performance impact of interfering signal mitigation

The goal of this experiment is to examine what affect adding a filter does to the performance of the radiometer. This performance would be the same no matter what type of radiometer we use as a f

6.4.1 Data Collected

The data collected for this experiment is the data points collected in table 6.5 and table 6.6. The remaining data is data generated from the equations outlined in this section.

6.4.2 Data Analysis

For experiment four, we will look at the performance of a software defined radiometer when a filter is used and what impact that has. Recall the following equation for $NE\Delta T$ from equation 3.7. Our sensitivity is based on the amount of noise we have from both the antenna or T_A plus the addition of our system noise which is T_{sys}. Finally our bandwidth of the signal, β and our integration time, τ, are the final factors that determine our $NE\Delta T$.

Our integration time is controllable, and can be set using the GUI panel for the software defined radio. In a typical radiometer, we often do not have any control of the bandwidth. It

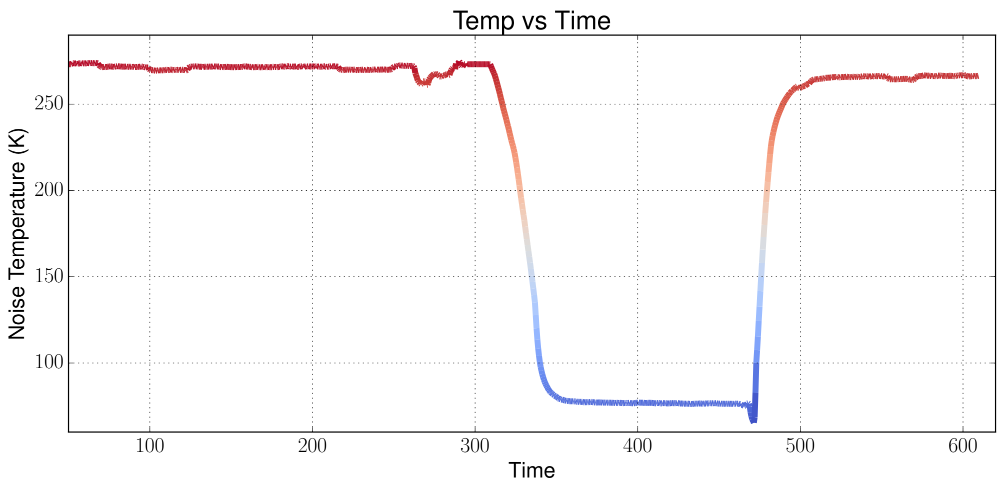

Figure 6.19 Graph showing the calibrated total power readings with the filter removing the offending signal

is often set by the mechanical bandwidth filters that are in place and the detection circuit to detect the noise power. In a software defined radio radio we do have more control on bandwidth as we can change our sampling rate which in turns controls our bandwidth. There is a limit as larger sampling rates require more data and processing bandwidth to process.

Recall from experiment III in section 6.3 where we filtered out an offending signal. While this allows us to filter out the offending signal and resume total power measurements, it comes at a cost of reducing the overall bandwidth available for power detection. In this experiment we look at this and derived the following equation 6.2.

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

Since we are notching out a portion of the bandwidth in order to remove the offending signal, this also takes away that bandwidth for our total power detection. The larger the filter, the more bandwidth that is subtracted from the total bandwidth available. Equation 6.2 accounts for this loss by subtracting the filter bandwidth from the total bandwidth.

This equation now takes into account the subtraction of the filter. The width of the band reject filter used then affects our $NE\Delta T$ and thus our performance of the radiometer.

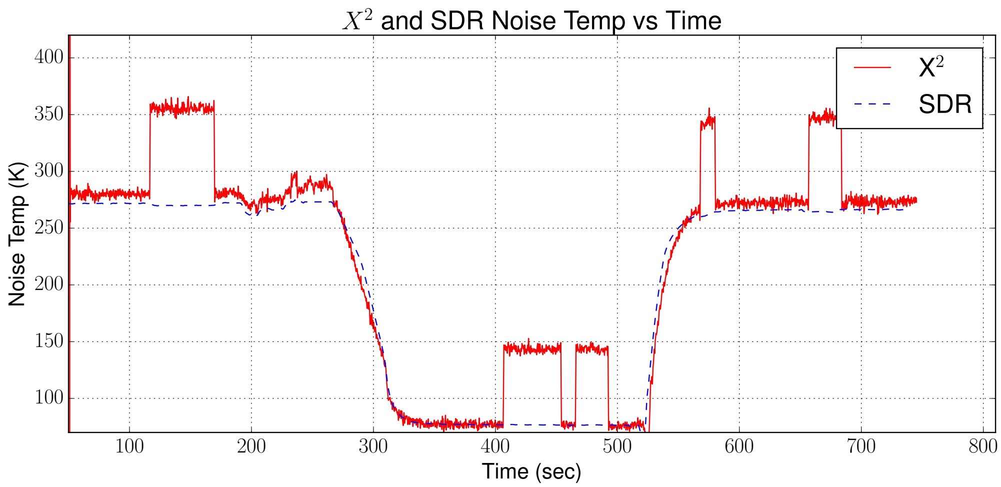

Figure 6.20 Image of the offending signal being filtered out by the SDR. It can be seen that the signal is no longer visible.

We can graph the expected response of the $NE\Delta T$ by adjusting the values of β_{fitler} to range from a narrowband filter, in our example 125 kHz all the way to 9.99 MHz or nearly all of the bandwidth. Figure 6.21 shows the expected exponential response of the $NE\Delta T$ as the size of the filter increases.

Figure 6.21 also shows measured standard deviation points for the software defined radiometer for set filter sizes. These filter sizes and collected data is in table 6.5. It can be seen in figure 6.21 that there is a good correlation between the expected sensitivity of the radiometer and the measured sensitivity of the radiometer.

Finally a line is added to figure 6.21 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 6.21 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 filter out more than 5 MHz or about half of the available bandwidth.

We can look at the relationship of the total power received as the bandwidth decreases. Figure 6.22 shows us both the measured total power received and the expected total power received as the bandwidth of the filter increases.

Table 6.5 Measured sensitivity and Bandwidth of Filter

rQ Value	**Temperature (K)**
.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

The total power is calculated from equation 3.2 and is based on the total system noise, both from the antenna and generated from the system, the bandwidth and finally from the gain of the amplifiers used. Our gain and noise temperatures are fixed, and in this experiment we use a system gain of 30 dB. This represents the gain that we see with the three LNAs used minus any losses or attenuation placed in the RF chain. What changes in this experiment is the amount of bandwidth. Again, we can modify equation 3.2 by subtracting the filter bandwidth from the total bandwidth available and 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 set filter sizes and then measuring the rQ value. These values can be found in table 6.6 and are added to figure 6.22 as the dots indicated on the graph. This also shows a very good correlation between the expected total power received and the measured total power received for experiment four.

Table 6.6 Measured Total Power and Bandwidth of Signal

rQ Value	**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

6.5 Benefits to Software Defined Radio Radiometer

A study was conducted on what benefits a software defined radio radiometer would have over a more traditional radiometer. This was focused on looking at three main areas; cost, weight and size, and the value a SDR 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 cost one hundred and fifty dollars approximately. 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 with the different software defined radios on the market is with both the resolution, or how many bits the ADC is, and the bandwidth they are able to handle.

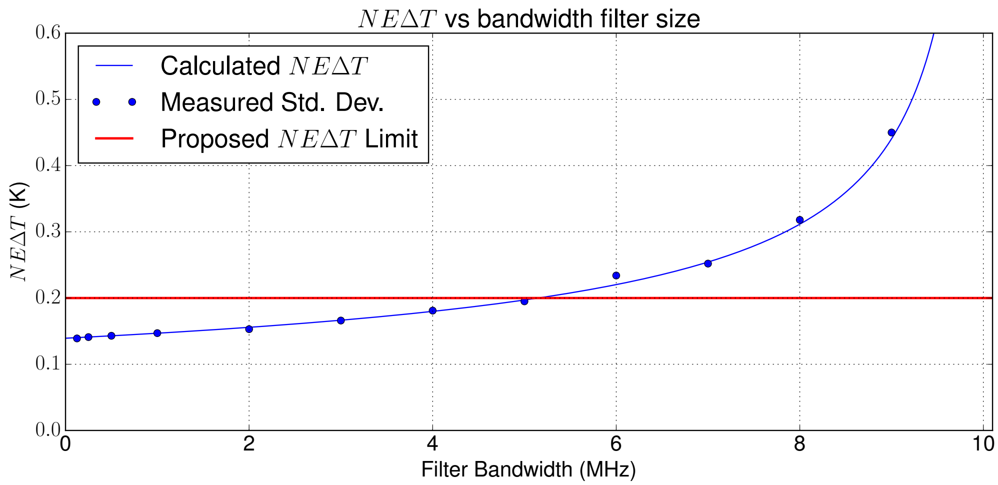

Figure 6.21 Graph of the calculated $NE\Delta T$, the proposed limit for the sensitivity of the radiometer and the measured standard deviation compared to the bandwidth filter size.

As seen in table 6.7, even the higher cost Ettus research equipment is a lower cost option than the custom built ISU radiometer purchased from University of Michigan and even a comparable off the shelf radiometer. It should be noted that the radiometer from the University of Michigan 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

[1] Purchase price in 2005

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[1]
Commercial Off the Shelf Unit		
Spectracyber 1420 MHz Hydrogen Line Spectrometer	1	$2,650

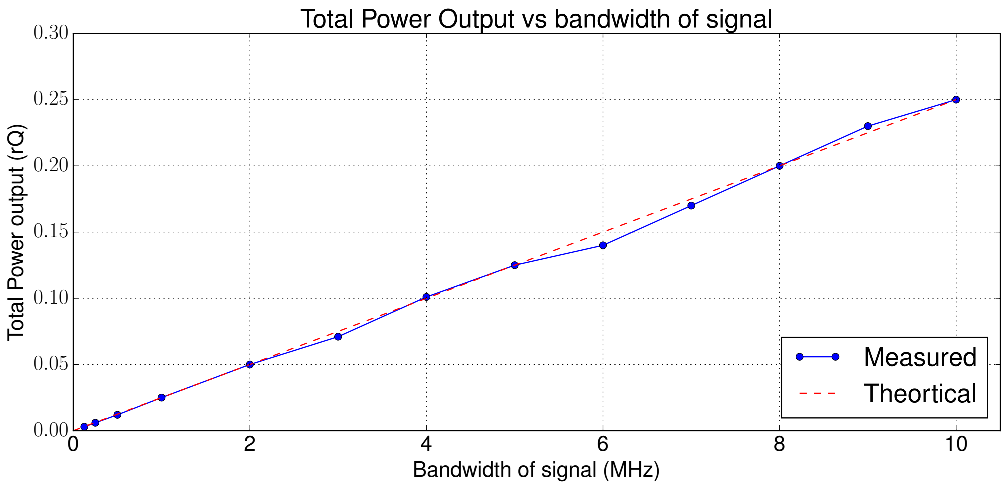

Figure 6.22 Graph of the total power measured and theoretical power versus the bandwidth of the measured signal.

the additional LNAs and daughter-board.

The largest cost benefit is that key components that you find in a radiometer, the filters and square-law detector can now be all done in software instead of needed additional equipment. The system is also much more frequency agile, which means it can work on 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. Some of this does depend on the SDR hardware however. The Ettus N200 for example uses daughter-boards to provide the RF interface. While these boards provide a high quality in the RF signal, it does come at a cost and are usually designed for certain bands of frequencies. Other low cost SDRs however are also very wide range in the frequencies they will work in. The HackRF for example works from 10 MHz to 6 GHz, but does so at the cost of lower resolution, less gain in its front end and a lower bandwidth that it can handle.

6.5.2 Weight and component size benefits

A typical radiometer has many components that are involved in the design of the radiometer. This includes filters, LNAs and the power detection or square-law detector used. These components add both weight, size and costs to the 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 both the ADC and usually a FPGA to do the processing on the signal, advances in semiconductor technology has continued to shrink these components. These components are also lighter than the filters often used in 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 Hydrogen Line Spectrometer	6 kg[2]

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

6.5.3 Value added benefits

A software defined radio radiometer adds additional value for two reasons. One, it is able to work with both frequency and magnitude where most radiometers do not. This allows for additional analysis on the signal and can help identify issues such as an interfering signal that 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. In addition, since we have both frequency and power information we can

[2] Estimated, no data available

create a system that is able to adapt to changing conditions such as dealing with an interfering signal.

6.6 Disadvantages of a SDR Radiometer

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

6.6.1 Power Consumption

One of the largest drawbacks to a SDR 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 however can help to minimize these power concerns as they are more efficient than using a full scale x86 based processor and on board computer system.

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 or computational requirements. This will cause additional strain on the processor and also in the memory requirements for the SDR as well.

6.6.2 Bandwidth constraints

While SDR technology has advanced, bandwidth is still a constraint that affects SDRs and in turn a SDR Radiometer. Bandwidth plays a critical role in the 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 as well.

In contrast, a square-law detector usually has a very large bandwidth, as much as one gigahertz, and is why we usually need to filter to the frequency band of interest.

6.7 Results Summary

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this thesis we have shown that an off the shelf SDR can be used to perform as a radiometer. Using a SDR has several advantages such as a more flexible system and can result in a less expensive system by using commercial off the shelf components. Since a SDR offers high flexibility, changes to the system can be done very quickly and helps in future proofing the system.

7.2 Future work

The work in this thesis demonstrated a basic radiometer system. There are other more complex radiometer systems that can be implemented in software. These include a correlating radiometer or a Dicke radiometer that can improve stability of the radiometer without the need of additional equipment such as thermal electric coolers.

7.2.1 Improvements

Several improvements can be done to further enhance this software defined radiometer. One improvement is to remove the personal computer (PC) and move the software defined radio to a Field Programmable Gate Array (FPGA) or an Application Specific Integrated Circuit (ASIC) solution. A second improvment is to reduce the dependcy on stable Low Noise Amplifiers (LNAs) by being able to compensate for instabilities in the system.

Removing the PC. For this thesis we focused on the software that would run on a PC or comparable computer system running a full operating system like Linux. This aids speeding up development by using software tools such as GNURadio to rapidly develop the software used to create a radiometer in software. While this works just fine for testing the theory on an off the shelf SDR acting as a radiometer, it does require hardware that is capable of running a full operating system and the associated software. For some applications of a radiometer, this is not a huge concern. In the case for the ISU radiometer, the concern is not that large since the radiometer is not designed to be portable and requires additional support equipment such as a generator anyway. However, other remote sensing applications, such as space based applications, would require a more efficient method. It is very possible to move the software generated in GNURadio into the firmware of the N200. This will help to offload the work needed by the computer and would allow for the computer or similar system to act as more of a control method and for data storage.

Improving stability. One of the largest challenges with a radiometer is improving the stability of the radiometer. Drifts in temperature can greatly affect the gains from the LNAs and also change how much noise all components in the radiometer contribute. A software defined radio can help as we are digitizing the signal as soon as possible. This helps in eliminating the analog components for power detection and even for filtering, but does not eliminate all physical hardware, mainly the LNAs. In this thesis, we did not focus on this issue since the tests were done in a controlled lab environment.

However, a more compact, lower cost and easier setup would be to just have the LNAs attach directly to the SDR without any temperature compensation. While this can be done, we have now lost stability in the LNAs and we need to compensate for that. Several methods have been discussed to handle instability in a traditional radiometer. Some of these methods would be suitable for implementation in a software defined radiometer.

A very traditional method is a Dicke radiometer. A Dicke radiometer switches between the antenna and a known noise source. A future work for a software defined 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 can also be adjusted and is performed in software, therefore stability in the noise source is not an issue.

Correlation. Another method to improve stability and sensitivity is to correlate the information with another input which can be another antenna looking at the same source or can be two polarization from the same antenna[Clapp and Maxwell (1967)]. This results in two receiver systems looking at the same source and you have two signals, 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 a 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 quite easy to do and is shown in figure 7.1.

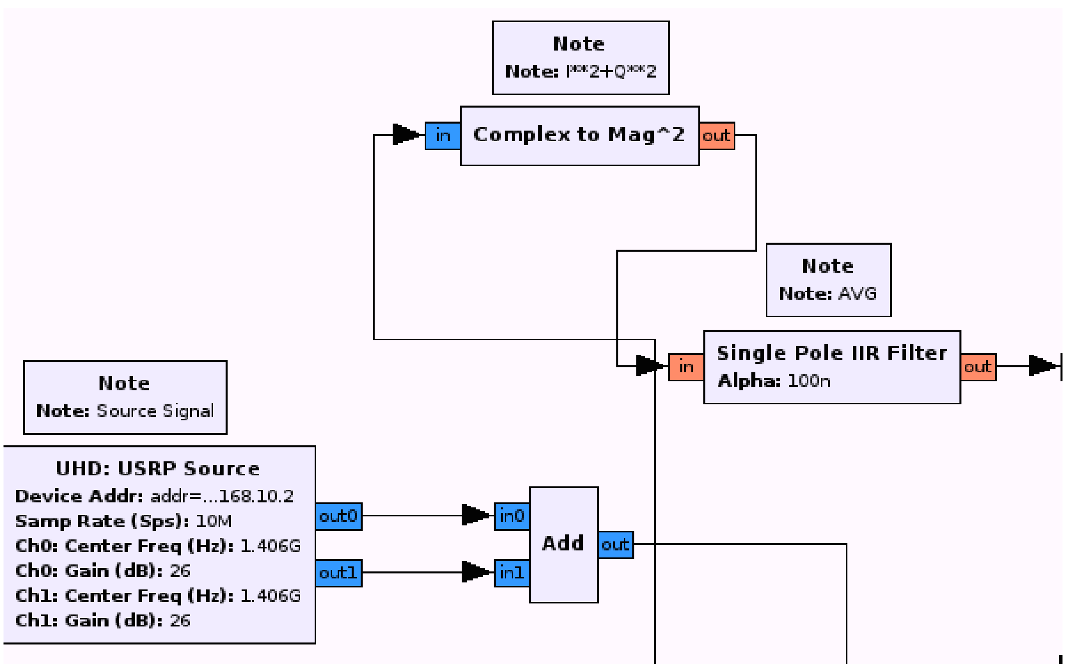

Figure 7.1 The key blocks used for creating a correlating radiometer in software. The key blocks is the USRP source, which allows us to address both daughter-boards and the ADD block which sums the signals.

Although figure 7.1 shows a correlating software defined radio radiometer, it has not been tested. In theory, this should correlate the signal and improve the sensitivity of the radiometer, however further testing is needed.

Correlation allows for unique ways for applications in radiometers including also performing beam steering [Villarino et al. (2002)]

7.2.2 Further testing

The improvements and additional features outlined in Section 7.2 will require additional testing and verification. While it has been shown that a software defined radiometer operates identically to a traditional radiometer, further testing is needed to verify different operating modes of a software defined radiometer.

7.3 Closing statement

This thesis demonstrated that it is possible to use off the shelf components and a software defined radio to implement a working radiometer that can be used in various radiometer applications such as soil moisture, ocean salinity and radio astronomy. Use of a software defined radiometer can potentially allow radiometers to be used by a wider audience of users by creating an easy to user GUI and reducing the cost and hardware complexity that most radiometers require. The result is more locations that are able to use radiometers as remote sensing tools to learn more about our planet and even the cosmos.

APPENDIX A. Source code

The following is the source code to several programs that make the radiometer possible. The first is the Python code used with the Ettus N200 software defined radio. This code is a heir block, which means it can be imported into GNURadio and used as a block. Once imported you may run your source into this block and use any sink needed to either display or log the data. This code was generated using GNURadio Companion.

The third code supplied is Python code that can be used to read the data generated and plot it. In many ways it mimics the functions of the Matlab script buy uses Python, NumPy and SciPy to perform the mathematical functions. This may be a better option for those that wish to look at the data but do not have access to Matlab since Python is free to download.

Finally, this code has been included in this thesis as a point of reference. It may be out of date and some other pieces of code was also used for the experimentation used in this thesis. Copies of this thesis source LaTeXcode, some experimental data, and any other code used can be found on the author's GitHub repository, `https://github.com/matgyver/Radiometer-SDR-Thesis`.

Python code for total power radiometer

The main code that acts as the total power radiometer is the TPR.py code. This module adds a custom block to GRC that can then be called and used like any other block in GRC. A screenshot of the blocks used in this is shown below.

Total Power Radiometer Block

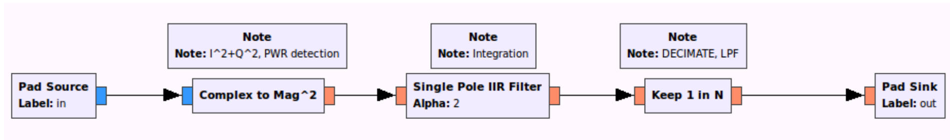

Figure A.1 Blocks used for creating a total power radiometer in software. Source: GNURadio Companion

```
#!/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. The code presented here is the export of this notebook. 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()
```

BIBLIOGRAPHY

Aitken, G. J. M. (1968). A new correlation radiometer. *Antennas and Propagation, IEEE Transactions on*, 16(2):224–228.

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.

Bremer, J. C. (1979). Improvement of scanning radiometer performance by digital reference averaging. *Instrumentation and Measurement, IEEE Transactions on*, 28(1):46–54.

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

Cross, D. (1998). Time domain filtering techniques for digital audio. Website. http://groovit.disjunkt.com/analog/time-domain/timefilt.html.

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.

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.

Goggins, W. B. (1967). A microwave feedback radiometer. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-3(1):83–90.

Hach, J. P. (1968). A very sensitive airborne microwave radiometer using two reference temperatures. *Microwave Theory and Techniques, IEEE Transactions on*, 16(9):629–636.

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.

McIntyre, E. and Gasiewski, A. (2012). A new technique for detecting the presence of weak interfering digital signals in radiometric noise. In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pages 1517–1520.

Mclntyre, E. and Gasiewski, A. (2007). An ultra-lightweight l-band digital lobe-differencing correlation radiometer (ldcr) for airborne uav sss mapping. In *Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International*, pages 1095–1097.

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.

Ohm, E. and Snell, W. (1963). A radiometer for a space communications receiver. In *Antennas and Propagation Society International Symposium, 1963*, volume 1, pages 16–20.

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.

Sarijari, M., Marwanto, A., Fisal, N., Yusof, S. K. S., Rashid, R., and Satria, M. (2009). Energy detection sensing based on gnu radio and usrp: An analysis study. In *Communications (MICC), 2009 IEEE 9th Malaysia International Conference on*, pages 338–342.

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.

Tiuri, M. (1964). Radio astronomy receivers. *Antennas and Propagation, IEEE Transactions on*, 12(7):930–938.

Uengtrakul, B. and Bunnjaweht, D. (2014). A cost efficient software defined radio receiver for demonstrating concepts in communication and signal processing using python and rtl-sdr. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on*, pages 394–399.

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.

Villarino, R., Enrique, L., Camps, A., Corbella, I., and Blanch, S. (2002). Design, implementation and test of the upc l-band automatic radiometer. In *Proc. URSI Commission-F 2002 Open Symp.*

Wang, Z., Liu, J., Lu, H., Zheng, W., Wang, X., and Li, B. (2012). A digital correlation full-polarimetric microwave radiometer design and calibration. In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pages 4688–4690.

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