5-6-2010

Software Defined GPS Receiver

Isaac Renninger

Nathan Clem

Follow this and additional works at: http://opus.ipfw.edu/etcs_seniorproj_engineering

Opus Citation
http://opus.ipfw.edu/etcs_seniorproj_engineering/6
Project Title: Software Defined GPS Receiver

Team Members: Isaac Renninger
              Nathan Clem

Faculty Advisor: Dr. Elizabeth Thompson

Industrial Advisor: Dr. Timothy Loos

Date: May 6, 2010
Table of Contents

Acknowledgment............................................................................................................................2
Abstract/Summary...........................................................................................................................3
Section I: Conceptual Design.........................................................................................................4
Section II: Building Process.........................................................................................................17
Section III: Testing.......................................................................................................................34
Evaluation and Recommendations................................................................................................42
Conclusion.....................................................................................................................................43
References.....................................................................................................................................44
Appendix A: Steps to Create Ubuntu 9.04 with GNU Radio on USB.............................................46
Appendix B: File Used to Record Data from USRP Board...............................................................48
Appendix C: Matlab Code Used to Up-Convert GPS Data..............................................................50
Appendix D: Matlab Code Used to Test Digitized Data from USRP..............................................52
Appendix E: Plot Results for USRP Test.....................................................................................53
Acknowledgment

We would like to thank Dr. Elizabeth Thompson for all her support in our design process for the software defined GPS receiver senior design project. Without her help, a clear path for the development of a software defined GPS receiver could not be established. As our project advisor, Dr. Thompson has continuously gone out of her way to provide guidance towards our attempts to create a final product for our senior design project.

We would also like to thank our industrial advisor, Dr. Timothy Loos. Without his knowledge in communications systems, a clear understanding of GPS receivers could not have been established within the time period needed to complete our project. We want to also give our deepest thanks to the Raytheon Company for donating the money we needed to complete the task of developing a software defined GPS receiver. This project could not be accomplished without this generous donation.

We also want to thank Dr. Todor Cooklev. He was generous enough to let us use one of his USRP Boards throughout this project’s development. This component is essential in our final design. The complexity of our final design would have greatly increased without the USRP Board.
Abstract/Summary

The versatility of today’s travel has increased the need for tracking one’s location with the upmost precision. This cannot be done with GPS receivers that are all hardware because of the variables that are encountered in the environment. A way to overcome this issue is through the use of software. A software defined GPS receiver uses software instead of complex hardware to overcome the variations that occur throughout the environment. By manipulating code, it can be possible to account for all types of geographical propagations.

Several designs were considered during the design process of the software defined GPS receiver. Many of these designs were eliminated because they were either too similar to other designs or the design may not actually be considered a design. A decision matrix was used to help decide the final design. What was left was a piece-by-piece design concept. Because of the high frequency and the very low power of the GPS signal, varieties of the individual components in the design were limited. With time being an issue and the difficulty of high frequency RF circuits, most of the hardware was either bought or loaned. Some of the components were not working properly so the GPS receiver had to be redesigned. The reconfiguration of the final design was possible because of the access to compatible components that were available in the lab. It was this reconfiguration that resulted in a functioning GPS receiver that met our design goals. Most of the construction of the final design was in the interfacing of the individual hardware components, the programming of the analog-to-digital converter, and the modifications to the software that processed the data that was retrieved from the GPS signal.
Section I: Conceptual Design
The conceptual design chosen for the software defined GPS receiver is piece-by-piece design. Figure 1 shows the block diagram of the conceptual design. This design will have components positioned at two locations. One location will be the roof of the Engineering, Technology, and Computer Science Building, and the other location will be in ET 342.

![Block diagram for conceptual design](image)

**Figure 1:** Block diagram for conceptual design.

### Active Antenna

The chosen conceptual design begins with a GPSRSSMA Active Antenna. This active antenna is specially made to receive the 1575.42 MHz GPS signal. The active antenna will be positioned on the roof of the Engineering, Technology, and Computer Science Building on the IPFW campus where it will be securely placed on the “I” beam that is currently on the roof.

### Amplifier/Bandpass Filter

After the active antenna, a 58529A Amplifier with L1 Bandpass Filter will be used to amplify and filter the GPS signal. Since the active antenna has a SMA male connector and the amplifier/filter has a N female connector, a SM4217 Adapter will be used to connect the antenna to the amplifier/filter. The amplifier/filter will also be positioned on the roof.
Coax Cable

The other end of the amplifier/filter will connect to a 50 foot LMR400UF Coax Cable. This coax cable will run from the roof to ET 342. The coax cable has NS5875-29 type N male connectors on each end, so no adapter will be needed to connect it to the amplifier/filter.

Bias Tee

In ET 342, the other end of the LMR400UF Coax Cable will connect to the RF/DC type N female connector of the SB3500 Bias Tee. When connected to a power supply, the bias tee is able to power the amplifier/filter and the active antenna without disrupting the GPS signal. Figure 2 shows the general circuitry of a common bias tee. The power supply used will be a Heath 2718 Tri-Power Supply. The power supply will connect to the BNC female connector of the bias tee. The Heath 2718 will supply 4.5 V to the active antenna and amplifier/filter. The GPS signal will output through the RF N male connector of the bias tee. From here, a SM4215 Adapter will be used to connect the bias tee a S6001 RG-174 Coax Cable.

![Figure 2: General circuit setup for a common bias tee.](image)

Downconverter

The other end of the S6001 RG-174 Coax Cable will connect to the RF input of the MAX2682EVKIT Evaluation Board. This evaluation board is used to house the MAX2682 Downconverter Chip and will aid in the interfacing of the downconverter to the other components in the receiver. A Narda Microline 1.0-12.5 GHz Sweeper Model 9500 Oscillator will be used as the local oscillator for the downconverter. When the local oscillator is set to 1605.42 MHz, the GPS signal will be down converted to 30 MHz. A S60301 RG-174 Coax Cable will be used to connect the local oscillator to the LO input of the MAX2682EVKIT Evaluation Board. The MAX2682EVKIT Evaluation Board will be powered by the same power supply output used earlier for the bias tee since it also requires 4.5 V. The power supply will connect to the VCC and GRD pins that are located on the evaluation board.
Amplifier

The down converted GPS signal will output the RF N type connector of the evaluation board and run through another S6001 Coax Cable to the ZFL-1000LN+ Amplifier. This amplifier will add additional gain to the signal and will be powered by the same power supply used throughout this design. The second voltage output on the power supply will be used to power the amplifier since this amplifier requires 15 V to fully operate.

Bandpass Filter

The signal will output from the amplifier to a BFSA-174-1 RG-174 Coax Cable that will connect to the BNC male connector of the BBP-30+ Bandpass Filter. The 30 MHz signal will output the BNC female end of the bandpass filter and run through a S60301 Coax Cable and connect to the SMA female input of the USRP Board.

USRP Board

A LFRX-LF Rev. 2.2 Daughter Board in the USRP will receive the signal. This daughter board is capable of receiving signals between DC – 50 MHz and has a programmable gain between 0 – 20 dB. The analog signal will be digitized in the FPGA board that is in the USRP at a sampling rate of 64 MHz\[1\]. This sampling rate cannot be changed. Once digitized, the signal will pass through a digital downconverter (DDC), where it will be down converted to baseband\[1\]. The digital down conversion allows for a suitable data rate when sending the data across a USB 2.0 port\[1\]. To narrow down the bandwidth, the decimation rate needs to be changed. Most of the data in the GPS signal is within a 2 MHz bandwidth\[2\]. By programming the decimation rate to 16, the sampling rate of the digitized signal will be 4 MHz (64 MHz / 16). Sampling twice the bandwidth will help prevent aliasing. Each sample is 4 bytes long (16 bits for the I data and 16 bits for the Q data)\[1\]. Since the sampling rate is 4 MHz with each sample being 4 bytes long, the data rate across the USB will be 16 MB/s. Figure 3 is a picture of the USRP Board that is planned to be implemented into the conceptual design.

![Figure 3: Picture of a USRP Board.](image-url)
Software

In order to get the proper amount of data needed to calculate the position of the GPS Receiver, at least 37 seconds of data from the GPS signal needs to be recorded, which is about 592 MB. The Matlab program provided by the book “A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach” requires at least 37 seconds of data in order to calculate the position of the GPS Receiver.

The digitized data that is being sent across the USB 2.0 will be stored on to a USB flash drive. The software that will be used to store the data is called GNU Radio, which is a Linux based program. Ubuntu 9.04 is the Linux operating system that will be used to run GNU Radio. To prevent from having to split a computer’s hard drive or to obtain a computer that has Linux on it, a Persistent Live USB containing Ubuntu 9.04 will be used. The GPS data will be stored to the hard drive of the desktop computer. Several steps and freeware are needed in order to make it possible to have Ubuntu boot up on a flash drive and have GNU Radio running on it. The steps on how to accomplish this can be found in Appendix A.

Once the flash drive contains the GPS data, a computer with Matlab is needed in order to process the data. The Matlab code used to calculate the GPS Receiver’s location is provided by the book “A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach”. Modifications to the code are needed because the program code assumes real data is being used and not I data and Q data. The program will read the recorded data from the USB flash drive and calculate the GPS Receiver’s position.
Table 1: Parts list for conceptual design with specifications*.

<table>
<thead>
<tr>
<th>Component</th>
<th>Decibel Gain/Loss</th>
<th>Noise Figure</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna[3]</td>
<td>28 dB Gain</td>
<td>2.0 dB**</td>
<td>-----</td>
</tr>
<tr>
<td>Adapter[4]</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Amp/BPF[5]</td>
<td>25 dB Gain</td>
<td>3.8 dB</td>
<td>150 MHz</td>
</tr>
<tr>
<td>Large Cable[6]</td>
<td>3.2 dB Loss</td>
<td>3.2 dB</td>
<td>-----</td>
</tr>
<tr>
<td>Bias Tee[7]</td>
<td>0.2 dB Loss</td>
<td>0.2 dB</td>
<td>-----</td>
</tr>
<tr>
<td>Adapter[4]</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Small Cable[8]</td>
<td>0.3 dB Loss (1575.42 MHz)</td>
<td>0.3 dB (1575.42 MHz)</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>0.1 dB Loss (30 MHz)</td>
<td>0.1 dB (30 MHz)</td>
<td>-----</td>
</tr>
<tr>
<td>Downconverter[9]</td>
<td>12.5 dB Gain</td>
<td>8.4 dB</td>
<td>-----</td>
</tr>
<tr>
<td>Evaluation Board[10]</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Small Cable[8]</td>
<td>0.1 dB Loss</td>
<td>0.1 dB</td>
<td>-----</td>
</tr>
<tr>
<td>Local Oscillator</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Amplifier[11]</td>
<td>20 dB Gain</td>
<td>2.9 dB</td>
<td>-----</td>
</tr>
<tr>
<td>Small Cable[8][12]</td>
<td>0.1 dB Loss</td>
<td>0.1 dB</td>
<td>-----</td>
</tr>
<tr>
<td>Bandpass Filter[13]</td>
<td>1.5 dB Loss</td>
<td>1.5 dB</td>
<td>6 MHz</td>
</tr>
<tr>
<td>USRP[14]</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Daughter Board[15]</td>
<td>20 dB Variable Gain</td>
<td>9 dB</td>
<td>-----</td>
</tr>
<tr>
<td>Personal Computer</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Power Supply</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>

*All components have an impedance of 50 Ω.
**Actual noise figure could not be found. Average of similar Active GPS Antennas that were found.

Table 2: Parts list for conceptual design with connectors and required power.

<table>
<thead>
<tr>
<th>Component</th>
<th>Model #</th>
<th>Required Power</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna[3]</td>
<td>GPSRSSSMA</td>
<td>3.3 V – 5.0 V</td>
<td>SMA/M</td>
</tr>
<tr>
<td>Adapter[4]</td>
<td>SM4217</td>
<td>-----</td>
<td>SMA/F – N/M</td>
</tr>
<tr>
<td>Amp/BPF[5]</td>
<td>58529A</td>
<td>4.5 V/13 mA</td>
<td>N/F – N/F</td>
</tr>
<tr>
<td>Large Cable[6]</td>
<td>LMK400UF</td>
<td>-----</td>
<td>N/M – N/M</td>
</tr>
<tr>
<td>Bias Tee[7]</td>
<td>SB3500</td>
<td>Max: 50 V/4 A</td>
<td>N/F – BNC/F – N/M</td>
</tr>
<tr>
<td>Adapter[4]</td>
<td>SM4215</td>
<td>-----</td>
<td>N/F – SMA/F</td>
</tr>
<tr>
<td>Small Cable[8]</td>
<td>S6001</td>
<td>-----</td>
<td>SMA/M – SMA/M</td>
</tr>
<tr>
<td>Downconverter[9]</td>
<td>MAX2682</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Evaluation Board[10]</td>
<td>MAX2682EVIKT</td>
<td>2.7 V – 5.5 V/50 mA</td>
<td>3 SMA/F</td>
</tr>
<tr>
<td>Small Cable[8]</td>
<td>S60301</td>
<td>-----</td>
<td>SMA/M – BNC/M</td>
</tr>
<tr>
<td>Local Oscillator</td>
<td>Sweeper Model 9500</td>
<td>120 VAC Outlet</td>
<td>BNC/F</td>
</tr>
<tr>
<td>Small Cable[8][12]</td>
<td>BFSA-174-1</td>
<td>-----</td>
<td>SMA/M – BNC/F</td>
</tr>
<tr>
<td>Bandpass Filter[5]</td>
<td>BBP-30+</td>
<td>-----</td>
<td>BNC/M – BNC/F</td>
</tr>
<tr>
<td>USRP[14]</td>
<td>USRP1</td>
<td>120 VAC Outlet</td>
<td>SMA/F – USB 2.0</td>
</tr>
<tr>
<td>Daughter Board[15]</td>
<td>LFRX-LF Rev. 2.2</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Personal Computer</td>
<td>Dell</td>
<td>120 VAC Outlet</td>
<td>USB 2.0</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Heath 2718</td>
<td>120 VAC Outlet</td>
<td>Voltage Pins</td>
</tr>
</tbody>
</table>
System Gain Analysis

For the small coax cables, the data sheet from the supplier’s website provides attenuation in dB per 100 feet for several frequencies. These example frequencies are shown in Table 3. The frequencies used in this design were not one of the example frequencies from the data sheet. To find the attenuation for the frequencies used in this design, the given values from Table 3 were plotted and are shown in Figure 4. The line representing nominal attenuation in Figure 4 can be used to estimate the attenuation in dB/100 ft for 30 MHz. This estimation is about 5 dB/100 ft. By multiplying this value by 0.01, it will yield the decibel loss per foot at 30 MHz, which is about 0.05 dB/ft. Rounding this value gives an attenuation of 0.1 dB/ft.

Table 3: Attenuation for selected frequencies[^8].

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Attenuation (dB/100 ft)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.9</td>
</tr>
<tr>
<td>10</td>
<td>3.3</td>
</tr>
<tr>
<td>50</td>
<td>5.8</td>
</tr>
<tr>
<td>100</td>
<td>8.4</td>
</tr>
<tr>
<td>200</td>
<td>12.5</td>
</tr>
<tr>
<td>400</td>
<td>19.0</td>
</tr>
<tr>
<td>700</td>
<td>27.0</td>
</tr>
<tr>
<td>900</td>
<td>31.0</td>
</tr>
<tr>
<td>1000</td>
<td>34.0</td>
</tr>
</tbody>
</table>

Figure 4: Plotted results from Table 3.

Since the example frequencies for the attenuation only go up to 1000 MHz, a different method had to be used to find the attenuation for the 1575.42 MHz frequency. Times Microwave provides an attenuation calculator online. By choosing the correct cable type along with its length and the frequency running through the cable, the online attenuation calculator can give an estimation of the attenuation. For this design, a 1575.42 MHz signal is being sent through a
S6001 RG-174 Coax Cable for a length of one foot. Plugging these values into the online calculator yields an estimated attenuation of 0.3 dB\(^{17}\).

As for the LMR400UF Coax Cable, the data sheet that was provided by the supplier’s website was used to determine the attenuation for the cable. In the data sheet, equation (1) was provided so the attenuation/100 ft can be calculated for frequencies that are not listed in the data sheet\(^{6}\). In equation (1), FREQ is the frequency of the signal in MHz traveling though the large coax cable\(^{6}\). For this design, FREQ = 1575.42 MHz. Since this design is only using 50 ft of cable instead of 100 ft, the outcome of this equation needs to be divided by two.

\[
\text{Attenuation for LMR400UF} = (0.146748)\sqrt{\text{FREQ}} + (0.000312)\times\text{FREQ} \quad (1)
\]

\[
\text{Attenuation for LMR400UF} = (0.146748)\sqrt{1575.42} + (0.000312)(1575.42) \quad (2)
\]

\[
\text{Attenuation for LMR400UF} = 6.3 \text{ dB/100 ft} \quad (3)
\]

\[
\text{Attenuation for LMR400UF} = 3.2 \text{ dB/50 ft} \quad (4)
\]

**Table 4:** Attenuation for coax cables used in conceptual design.

<table>
<thead>
<tr>
<th>Coax Cable</th>
<th>Frequency (MHz)</th>
<th>Attenuation (dB/100 ft)</th>
<th>Attenuation (dB/50 ft)</th>
<th>Attenuation (dB/ft)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMR400UF</td>
<td>1575.42</td>
<td>6.3</td>
<td>3.2</td>
<td>---</td>
</tr>
<tr>
<td>S6001</td>
<td>1575.42</td>
<td>34.8</td>
<td>---</td>
<td>0.3</td>
</tr>
<tr>
<td>S6001</td>
<td>30</td>
<td>5.0</td>
<td>---</td>
<td>0.1</td>
</tr>
<tr>
<td>BFSA-174-1</td>
<td>30</td>
<td>5.0</td>
<td>---</td>
<td>0.1</td>
</tr>
<tr>
<td>S60301</td>
<td>30</td>
<td>5.0</td>
<td>---</td>
<td>0.1</td>
</tr>
</tbody>
</table>
The rest of the components had their gains and losses listed in their corresponding data sheets. By adding the gains for each component to the initial signal power level of the GPS signal, which is -130 dBm, the final power level for the GPS signal can be calculated\(^2\). Table 5 shows the gain for each corresponding component which adds up to a total gain of about 100.0 dB. This yields a final signal power level of -30.0 dBm. The adapters were ignored in the analysis since their effects on the signal are negligible.

**Table 5:** Total gain for the conceptual design.

<table>
<thead>
<tr>
<th>Component</th>
<th>Gain (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna</td>
<td>28</td>
</tr>
<tr>
<td>Amp/BPF</td>
<td>25</td>
</tr>
<tr>
<td>Large Coax Cable</td>
<td>-3.2</td>
</tr>
<tr>
<td>Bias Tee</td>
<td>-0.2</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.3</td>
</tr>
<tr>
<td>Downconverter</td>
<td>12.5</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.1</td>
</tr>
<tr>
<td>Amplifier</td>
<td>20</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.1</td>
</tr>
<tr>
<td>BPF</td>
<td>-1.5</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.1</td>
</tr>
<tr>
<td>USRP</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total Gain</strong></td>
<td><strong>100.0</strong></td>
</tr>
</tbody>
</table>
Noise Figure Analysis

For each component, the noise figure is listed in their corresponding data sheets. By using equation (5), the total noise added to the system can be calculated, where $F_1, F_2, F_3, \ldots, F_k$ are the noise figures for each component and $G_1, G_2, G_3, \ldots, G_k$ are the gains for each component. The linear power values for each component are used in equation (5). Table 6 lists the gain and noise figure for each component. The total noise figure of the system is about 2.0 dB.

$$\text{Total Noise Figure} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \cdots + \frac{F_k - 1}{G_1 G_2 \ldots G_k}$$

(5)

Table 6: Gain and noise figure for each component in conceptual design.

<table>
<thead>
<tr>
<th>Component</th>
<th>Gain (dB)</th>
<th>Noise Figure (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna</td>
<td>28.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Amp/BPF</td>
<td>25.0</td>
<td>3.8</td>
</tr>
<tr>
<td>Large Coax Cable</td>
<td>-3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>Bias Tee</td>
<td>-0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>Downconverter</td>
<td>12.5</td>
<td>8.4</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Amplifier</td>
<td>20.0</td>
<td>2.9</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>BPF</td>
<td>-1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>USRP</td>
<td>20.0</td>
<td>9.0</td>
</tr>
</tbody>
</table>

| **System Noise Figure** | **2.0** |


**Signal-to-Noise Ratio**

**Initial Signal-to-Noise Ratio**

The initial GPS signal power level is at -130 dBm\(^2\). The initial thermal noise level is at -110.97 dBm\(^2\). The thermal noise level was calculated by using equation (6), where \(k\) is Boltzmann’s Constant (1.38 \times 10^{-23} \text{ J/K}), \(T_0\) is the temperature at the receiver input in Kelvin (290 K), and \(B\) is the bandwidth in Hertz (2,000,000 Hz).

\[
Initial \text{ Thermal Noise Power Level} = kT_oB
\]  \hspace{1cm} (6)

\[
Initial \text{ Thermal Noise Power Level} = (1.38 \times 10^{-23})(290)(2000000)
\]  \hspace{1cm} (7)

\[
Initial \text{ Thermal Noise Power Level} = 8.004 \times 10^{-15} \text{ W} = -110.97 \text{ dBm}
\]  \hspace{1cm} (8)

By subtracting the decibel value of the initial thermal noise level from the decibel value of the initial GPS signal power level, the initial Signal-to-Noise Ratio (SNR) can be found. For this situation, the initial SNR is -19.03 dB, meaning the signal is below the thermal noise level.

**Final Signal-to-Noise Ratio**

The GPS signal is a spread spectrum signal which means it is a signal with a small bandwidth that is being spread out over a larger bandwidth\(^2\). The spreading of the GPS signal lowers its power level. This allows the signal to have a power level below the thermal noise level. The method used to spread the GPS signal is called Direct Sequence Spread Spectrum (DSSS)\(^2\). The GPS signal is spread by taking the navigation data, which has a bit rate of 50 bps, and doing a logical XOR with the C/A code, which has a chip rate of 1,023,000 cps\(^2\). When the data is being processed, the data will go from the relatively large bandwidth of the C/A code back to the narrow bandwidth of the navigation data. This reduction of bandwidth gives the signal needed gain. This gain will only be added to the signal and not the noise. Equation (9) calculates the processing gain for the GPS signal\(^2\). This equation yields a processing gain of 43 dB.

\[
Processing Gain = 10\log_{10} \left( \frac{\text{Chip Rate}}{\text{Data Rate}} \right)
\]  \hspace{1cm} (9)

\[
Processing Gain = 10\log_{10} \left( \frac{1,023,000}{50} \right)
\]  \hspace{1cm} (10)

\[
Processing Gain = 43 \text{ dB}
\]  \hspace{1cm} (11)
The final SNR can be found using equation (12), where $S_i$ is the initial signal power level, $G$ is the system gain, and $P$ is the processing gain that was calculated above.

$$Final\ Signal\ Power\ Level = S_i + G + P$$  \hspace{1cm} (12)

$$Final\ Signal\ Power\ Level = -130 + 100.0 + 43$$  \hspace{1cm} (13)

$$Final\ Signal\ Power\ Level = 13\ dBm$$  \hspace{1cm} (14)

The final thermal noise level can be calculated by using equation (15), where $N_i$ is the initial thermal noise power level, $F$ is the total noise figure of the system, and $G$ is the total gain of the system.

$$Final\ Thermal\ Noise\ Power\ Level = N_i + F + G$$  \hspace{1cm} (15)

$$Final\ Thermal\ Noise\ Power\ Level = -110.97 + 2.0 + 100.0$$  \hspace{1cm} (16)

$$Final\ Thermal\ Noise\ Power\ Level = -8.97\ dBm$$  \hspace{1cm} (17)

The final SNR can now be calculated by subtracting the decibel value of the final thermal noise power level from the decibel value of the final signal power level. This simple calculation yields a final signal-to-noise ratio of 21.97 dB.
Cost Analysis

Table 7: Cost analysis for the chosen conceptual design.

<table>
<thead>
<tr>
<th>Component</th>
<th>Company</th>
<th>Quantity</th>
<th>Cost per Component</th>
<th>Subtotal</th>
<th>Shipping &amp; Handling</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna</td>
<td>GPRS SSSMA</td>
<td>1</td>
<td>$25.00</td>
<td>$25.00</td>
<td>$9.48</td>
<td>$34.48</td>
</tr>
<tr>
<td>Adapter</td>
<td>SM4217</td>
<td>1</td>
<td>$17.00</td>
<td>$17.00</td>
<td>-----</td>
<td>$17.00</td>
</tr>
<tr>
<td>Amp/BPF</td>
<td>58529A</td>
<td>1</td>
<td>$435.00</td>
<td>$435.00</td>
<td>$19.53</td>
<td>$454.53</td>
</tr>
<tr>
<td>Large Cable</td>
<td>LMR400UF</td>
<td>50 ft</td>
<td>$1.19/ft</td>
<td>$59.50</td>
<td>$11.25</td>
<td>$70.75</td>
</tr>
<tr>
<td>Cable Connectors</td>
<td>NS5875-29</td>
<td>1</td>
<td>$26.95</td>
<td>$26.95</td>
<td>-----</td>
<td>$26.95</td>
</tr>
<tr>
<td>Bias Tee</td>
<td>SB3500</td>
<td>1</td>
<td>$147.00</td>
<td>$147.00</td>
<td>$5.00</td>
<td>$152.00</td>
</tr>
<tr>
<td>Adapter</td>
<td>SM4215</td>
<td>1</td>
<td>$16.00</td>
<td>$16.00</td>
<td>-----</td>
<td>$16.00</td>
</tr>
<tr>
<td>Small Cable</td>
<td>S6002</td>
<td>2</td>
<td>$10.50</td>
<td>$21.00</td>
<td>$18.00</td>
<td>$39.00</td>
</tr>
<tr>
<td>Downconverter</td>
<td>MAX2682</td>
<td>1</td>
<td>$0.00</td>
<td>$0.00</td>
<td>-----</td>
<td>$0.00</td>
</tr>
<tr>
<td>Evaluation Board</td>
<td>MAX2682EVKIT</td>
<td>1</td>
<td>$90.00</td>
<td>$90.00</td>
<td>$15.00</td>
<td>$105.00</td>
</tr>
<tr>
<td>Small Cable</td>
<td>S60301</td>
<td>2</td>
<td>$12.50</td>
<td>$25.00</td>
<td>-----</td>
<td>$25.00</td>
</tr>
<tr>
<td>Local Oscillator*</td>
<td>Sweeper Model 9500</td>
<td>1</td>
<td>$495.00**</td>
<td>$495.00</td>
<td>-----</td>
<td>$495.00</td>
</tr>
<tr>
<td>Amplifier</td>
<td>ZFL-1000LN+</td>
<td>1</td>
<td>$89.95</td>
<td>$89.95</td>
<td>-----</td>
<td>$89.95</td>
</tr>
<tr>
<td>Small Cable</td>
<td>BFSA-174-1</td>
<td>1</td>
<td>$17.36</td>
<td>$17.36</td>
<td>$10.19</td>
<td>$27.55</td>
</tr>
<tr>
<td>Bandpass Filter</td>
<td>BBP-30+</td>
<td>1</td>
<td>$40.95</td>
<td>$40.95</td>
<td>$8.42</td>
<td>$49.37</td>
</tr>
<tr>
<td>USRP*</td>
<td>USRP1</td>
<td>1</td>
<td>$700.00</td>
<td>$700.00</td>
<td>-----</td>
<td>$700.00</td>
</tr>
<tr>
<td>Daughter Board*</td>
<td>LFRX-LF</td>
<td>1</td>
<td>$75.00</td>
<td>$75.00</td>
<td>-----</td>
<td>$75.00</td>
</tr>
<tr>
<td>Personal Computer*</td>
<td>Average Dell</td>
<td>1</td>
<td>$500.00</td>
<td>$500.00</td>
<td>-----</td>
<td>$500.00</td>
</tr>
<tr>
<td>Power Supply*</td>
<td>Heath 2718</td>
<td>1</td>
<td>$49.95</td>
<td>$49.95</td>
<td>-----</td>
<td>$49.95</td>
</tr>
<tr>
<td><strong>Overall Total</strong></td>
<td></td>
<td></td>
<td>$2,749.35</td>
<td>$2,830.66</td>
<td>$96.87</td>
<td>$2,927.53</td>
</tr>
</tbody>
</table>

*Components provided by IPFW.

** Actual price could not be found so an average of similar local oscillators was taken.
Section II: Building Process
GPS Receiver Construction Issues

Rooftop Setup

The first step in the building process was to place the active antenna on the roof of the engineering building. This step was delayed three weeks due to the snow storm that hit Northeast Indiana at the start of the semester. Once the roof was cleared, the GPSRSSMA Active Antenna was able to be attached to the “I” beam using a dogleg bracket. The SMA connector of the GPS antenna was connected to the 58529A Amplifier using the SM4217 Adapter. The 50 foot LMR400UF was connected to the other end of the 58529A Amplifier. The rooftop setup of GPS receiver is shown in Figure 5. Black electrical tape was used to secure the amplifier/bandpass filter and to protect it from the outdoor elements.

![Figure 5: Picture of rooftop setup of GPS receiver.](image)

The other end of the large coax cable was sent through the roof access panel that is located a couple feet from the location of the active antenna and amplifier/bandpass filter. The roof access panel is located above the lab where the rest of the GPS receiver was constructed. Figure 6 is a picture of the access panel used to send the LMR400UF Coax Cable to ET 342.
ET 342 Hardware Construction

With the other end of the LMR400UF Coax Cable in the lab, the rest of the hardware was constructed simply by following Figure 1 and making sure the connectors matched up for each component. This part of the building process was the quickest and the simplest in the building process and only took a few minutes to accomplish. To connect the downconverter to the power supply, small wires were soldered to the VCC and GND pins of the MAX2682EVKIT Evaluation Board. Since the pins are small and close together, using voltage clips would not be practical. For the ZFL-1000LN+ Amplifier, voltage clips were used to connect the power supply to the VCC and GND pins since those pins are larger and separated more.

Installing Ubuntu

Once the hardware was pieced together, the software portion of the design was assembled. This was where the first major problem was at. Most of the software problems were from the interfacing of Ubuntu when it was being run from the USB flash drive. When GNU Radio Companion was running and data was being captured, Ubuntu would freeze up and would force a hard shutdown of the computer. These frequent freeze ups would happen because the data transfer rate was too much for the USB flash drive to handle. In this design, the data rate from the USRP Board is 16 MB/s. For a common flash drive, the rate at which data is being read from it is 20 – 25 MB/s and the rate at which data is being written to it is 5 – 10 MB/s. When the data is being transferred from the USRP to the hard drive of the computer, the operating system needs to be able to keep up with that data rate. The data rate from the USRP is 16 MB/s, but the information needed from the operating system to perform that transfer was being wrote at a data rate between 5 – 10 MB/s. This caused the flash drive to be pushed past its limit and would make the operating system crash.
To prevent Ubuntu from crashing when acquiring data from the USRP Board, Ubuntu was installed on the desktop computer using software that was provided when creating a Persistent Live USB with Ubuntu 9.04. When installing Ubuntu on the desktop computer, a partition had to be setup to create a dual boot computer. While the partition was being setup, crucial files in Windows XP were over-written causing Windows XP to no longer work on the computer. With windows now gone on the computer, it was decided to have Ubuntu take up the whole hard drive and make the desktop become a 100% pure Linux computer. This decision will not affect the final design since the data acquired from the USRP Board will still need to be transferred via flash drive to a computer that has Matlab on it. Rebooting the computer would take just as much time as using a flash drive to transfer the data.

Component Testing

When testing the individual components of the GPS receiver, it was found that most of the components were working properly except for the Narda Microline Local Oscillator. It was found that the signal that was being generated by the local oscillator would tend to drift. When it was connected to a frequency counter, the center frequency of the generated signal would be off by about 50 MHz and it would drift back and forth by several hundred kilohertz. This kind of drifting is unacceptable since the GPS signal only has a bandwidth of 2 MHz. It was decided that another local oscillator had to be used. This decision was not easy to make because the Narda Microline is the only signal generator on IPFW campus that is able to generate a signal at 1605.42 MHz. That frequency is important since it is the frequency needed to down convert the GPS signal to 30 MHz. The new signal generator that was used was a Marconi Instruments 2022A. This signal generator is capable of generating a signal between 10 kHz – 1000 MHz. This means the LFRX-LF Rev 2.2 Daughter Board in the USRP will no longer be used to receive the GPS signal because the Marconi is unable to generate a signal that can downconvert the GPS signal to a level between DC – 50 MHz. It was decided that the other daughter board that was on the USRP had to be used. The other daughter board is a FLEX900 (now called RFX900). This new daughter board is capable of receiving signals in the range of 750 MHz – 1050 MHz and has a programmable gain between 0 – 90 dB\[15\]. The Marconi 2022A was set to 660.42 MHz so the GPS signal will now be down converted to a 915 MHz signal. Since the GPS signal is down converted to 915 MHz, the BBP-30+ Bandpass filter can no longer be implemented into the design because it is made for a signal with a center frequency of 30 MHz.

With the change in daughter boards, a new range in programmable gains is available. The LFRX-LF Rev 2.2 Daughter Board only has a programmable gain range of 0 – 20 dB. The RFX900 Daughter Board has a programmable gain range of 0 – 90 dB. This increased range allows for the removal of the ZFL-1000LN+ Amplifier from the design. The coax cable BFSA-174-1 and one of the S60301 coax cables were also removed since they are no longer needed to connect the ZFL-1000LN+ Amplifier and the BBP-30+ Bandpass Filter. The range of gain for the daughter board was found by using the program file “usrp_rx_cfile.py” which can be found in Appendix B. The highlighted line in the code was used to display the range of programmable gain on the terminal window in Ubuntu. This file came with the GNU Radio download and is executed by entering the file name in the terminal window of Ubuntu.
**Completed Design Hardware Setup**

With the Narda Microline local oscillator not functioning properly, it caused a series of events to happen that changed the layout of the conceptual design. The new setup for the GPS receiver can be seen on Figure 7. It can be seen that the ZFL-1000LN+ Amplifier and the BBP-30+ Bandpass Filter are no longer implemented in the design. The removal of those components also caused the removal of one of the S60301 Coax Cables and the BFSA-174-1 RG-174 Coax Cable. Also, the local oscillator being used is the Marconi 2022A instead of the Narda Microline and the RFX900 Daughter Board is being used instead of the LFRX-LF Rev. 2.2 Daughter Board.

![Figure 7: Hardware block diagram for completed design.](image-url)
Figure 8 is a picture of the constructed GPS receiver in ET 342. The changes made to the design did not affect the rooftop setup of the GPS receiver so figures 5 and 6 can represent the rooftop setup of the completed design. Tables 8 and 9 show the completed design’s components along with their specifications.

![Figure 8: Picture of GPS receiver setup in ET 342.](image)

**Table 8:** Parts list for completed design with specifications*.

<table>
<thead>
<tr>
<th>Component</th>
<th>Decibel Gain/Loss</th>
<th>Noise Figure</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna[3]</td>
<td>28 dB Gain</td>
<td>2.0 dB**</td>
<td></td>
</tr>
<tr>
<td>Adapter[4]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amp/BPF[5]</td>
<td>25 dB Gain</td>
<td>3.8 dB</td>
<td>150 MHz</td>
</tr>
<tr>
<td>Large Cable[6]</td>
<td>3.2 dB Loss</td>
<td>3.2 dB</td>
<td></td>
</tr>
<tr>
<td>Bias Tee[7]</td>
<td>0.2 dB Loss</td>
<td>0.2 dB</td>
<td></td>
</tr>
<tr>
<td>Adapter[4]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Cable[8]</td>
<td>0.3 dB Loss</td>
<td>0.3 dB</td>
<td></td>
</tr>
<tr>
<td>Downconverter[9]</td>
<td>0.0 dB Gain***</td>
<td>8.4 dB</td>
<td></td>
</tr>
<tr>
<td>Evaluation Board[10]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Cable[8]</td>
<td>0.3 dB Loss</td>
<td>0.3 dB</td>
<td></td>
</tr>
<tr>
<td>Local Oscillator</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USRP[11][14]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daughter Board[15]</td>
<td>90 dB Variable Gain</td>
<td>9 dB</td>
<td></td>
</tr>
<tr>
<td>Personal Computer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Supply</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*All components have an impedance of 50 Ω.

**Actual noise figure could not be found. Average of similar Active GPS Antennas that were found.

***Test result showed 0.0 dB gain when down converting to 915 MHz.
Table 9: Parts list for completed design with connectors and required power.

<table>
<thead>
<tr>
<th>Component</th>
<th>Model #</th>
<th>Required Power</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna</td>
<td>GPRRSSMA</td>
<td>3.3 V – 5.0 V</td>
<td>SMA/M</td>
</tr>
<tr>
<td>Adapter</td>
<td>SM4217</td>
<td></td>
<td>SMA/F – N/M</td>
</tr>
<tr>
<td>Amp/BPF</td>
<td>58529A</td>
<td>4.5 V/13 mA</td>
<td>N/F – N/F</td>
</tr>
<tr>
<td>Large Cable</td>
<td>LMR400UF</td>
<td></td>
<td>N/M – N/M</td>
</tr>
<tr>
<td>Bias Tee</td>
<td>SB3500</td>
<td>Max: 50 V/4 A</td>
<td>N/F – BNC/F – N/M</td>
</tr>
<tr>
<td>Adapter</td>
<td>SM4215</td>
<td></td>
<td>N/F – SMA/F</td>
</tr>
<tr>
<td>Small Cable</td>
<td>S6001</td>
<td></td>
<td>SMA/M – SMA/M</td>
</tr>
<tr>
<td>Downconverter</td>
<td>MAX2682</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation Board</td>
<td>MAX2682EVKIT</td>
<td>2.7 V – 5.5 V/50 mA</td>
<td>3 SMA/F</td>
</tr>
<tr>
<td>Small Cable</td>
<td>S60301</td>
<td></td>
<td>SMA/M – BNC/M</td>
</tr>
<tr>
<td>Local Oscillator</td>
<td>Marconi 2022A</td>
<td>120 VAC Outlet</td>
<td>BNC/F</td>
</tr>
<tr>
<td>USRP</td>
<td>USRP1</td>
<td>120 VAC Outlet</td>
<td>SMA/F – USB 2.0</td>
</tr>
<tr>
<td>Daughter Board</td>
<td>RFX900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personal Computer</td>
<td>Dell</td>
<td>120 VAC Outlet</td>
<td>USB 2.0</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Heath 2718</td>
<td>120 VAC Outlet</td>
<td>Voltage Pins</td>
</tr>
</tbody>
</table>
Software Implementation

Recording Data

Once the hardware setup was figured out, it was time to start recording the GPS data from the USRP. It was planned that GNU Radio Companion (GRC) was to be used to record the data, but the data that was being recorded was the wrong data type. The data from the USRP needed to be recorded as a “short” but GRC was recording the data as a “long” causing the data file to be twice as large. The data files were about 1.2 GB when they were supposed to be about 600 MB for 37 seconds of data. This was solved by using the file “usrp_rx_cfile.py”. To capture the data, the terminal window in Ubuntu needs to be in the directory where “usrp_rx_cfile.py” is stored and the following needs to be entered.

```
usrp_rx_cfile.py -s -g 40 -f 915M -d 16 -RB -N 160M usrpdata.dat
```

For this command, -s stands for “short” data type, -g sets the gain, -f sets the frequency, -d sets the decimation, -R tells which daughter board to read from, and -N is the number of samples to take. One hundred sixty samples were taken because that is equivalent to recording 40 seconds of data when sampling at 4 MHz (4 MHz × 40 seconds). Forty seconds of data was recorded to make sure there was enough data for the GPS software to use even though the software needs 37 seconds of data. With 4 bytes per sample for 160 million samples, the size of the data file is 610.4 MB. The gain was doubled to 40 dB to account for the ZFL-1000LN+ Amplifier no longer being in the design. The decimation was set to 16 since the design calls for sampling rate for the digitized data to be 4 MHz.

Digital Lowpass Filter

Once the data was recorded, a flash drive was used to transfer the data from the Linux PC to a Windows PC. With the GPS data now on Windows, the next step in the post processing stage of the design was to use the free downloadable software from www.brothersoft.com to help filter out any excess noise in the data file[16]. The free downloadable software is called SOX-14.3.0[16]. With the help of Dr. Tim Loos and Matlab’s Filter Design, the coefficients needed for the lowpass filter software was found.

Up-converting of the Data

The next thing that was done to the GPS data once it was filtered with the SOX-14.3.0 software filter was to up-convert the data using the program “int8modulatebyfs4.m” in Matlab. This file is located in Appendix C and is not part of the GPS software used to calculate the receiver’s position. The reason the signal needs to be up-converted is because the GPS software provided by the book “A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach” assumes that the incoming data is real data and not IQ data. The GPS software takes the real data and separates it into IQ data in order to calculate the receiver’s position, but the data being read has to be real in order to do that. The data coming out of the USRP is in 16 bit IQ format (16 bit I, 16 bit Q) and not real. The file “int8modulatebyfs4.m” creates a carrier frequency of 1 MHz and modulates it to the real (I) and imaginary (Q) parts of the GPS data, thus creating a new real signal of 1 MHz that the GPS software can understand.
The GPS software also requires that the data be in 8 bit format and not in 16 bit format. The file “int8modulatebyfs4.m” takes care of this by scaling the down the data so that it is no longer 16 bits but rather 8 bits. This was done by taking the highest absolute 16 bit value in the modulated data and dividing it by 127 which creates a scaling constant. Then every 16 bits in the newly formed real data is divided by this scaling constant. The highlighted line of code in Appendix C is where this calculation occurs. The flowchart for this Matlab file is shown in Figure 9.

Figure 9: Flowchart for Matlab file “int8modulatebyfs4.m”.
Calculating GPS Receiver Position

Now that the GPS data is modulated, the GPS software is now ready to process the data. Figure 10 is a simplified flowchart for the GPS software that will be used to calculate the position of the GPS receiver. This program came from the DVD that comes with the book “A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach”.

Figure 10: Flowchart for GPS software.
This first thing the program does is probes the data file to see what kind of information is on it. The result of this probing comes in the form of a plot that is shown in Figure 11. This plot shows the time domain and the frequency domain of the data file along with a histogram of the binary data.

Figure 11: Probing results of GPS data.
Based on these results, the user can then choose to continue (enter “1”) or exit (enter “0”) the program. If the user enters “1”, then the program will extract the C/A code from the data file and correlate it with the C/A codes that were generated by the software. If there is a strong correlation, then the program assumes that it found a satellite. The program displays the found satellites by means of a bar graph. An example of this is shown in Figure 12.

**Figure 12:** Bar graph showing which satellites were acquired.
Now that the GPS software has acquired some satellites, the program will then begin tracking each satellite for 37 seconds, meaning the program will go through the 37 seconds of recorded data (564.6 MB) for each satellite found. This stage of post processing is the longest because this is a lot of data for Matlab to process. When the tracking is finished, the program will display the results for each satellite that was found. The results are not always good. Figure 13 shows an example of good tracking results. The most important thing to take from these results is the small plot located in the top left corner of the figure. This small plot shows the binary values of the signal. A good signal has separated values of 1’s and 0’s. If the tracking results show 1’s and 0’s then the program is able to distinguish the navigation data from the satellite, thus able to calculate the GPS receiver’s location.

![Figure 13: Example of plotted tracking results for one satellite.](image)

Figure 14 shows what a bad signal would look like based on the tracking results. It can be seen that the binary values are all over the place. This would mean the program would not be able to distinguish any of the data and would either result in an incorrect receiver position or the program would be unable to calculate a position at all.

![Figure 14: Example of a bad signal.](image)
When the tracking stage of the GPS software is finished, the program will then calculate the position of the receiver. The calculated results are plotted and are shown in Figure 15. The lower left plot displays the latitude and longitude coordinates.

**Figure 15:** Plot showing the receiver’s position.
System Gain Analysis

The attenuation for the cables carrying the 1575.42 MHz signal will not change in the completed design since the same cables are used. The only cable that will have a different attenuation from the conceptual design is the cable that is carrying the down converted signal. Since the down converted signal is now 915 MHz and not 30 MHz, the attenuation in the S6001 coax cable will be different. The down converted signal is within the example frequencies that were giving in Table 3, so the attenuation per 100 ft can be estimated from the line representing the nominal attenuation in Figure 4. This estimated value can be found in Table 10.

Table 10: Attenuation for coax cables used in completed design.

<table>
<thead>
<tr>
<th>Coax Cable</th>
<th>Frequency (MHz)</th>
<th>Attenuation (dB/100 ft)</th>
<th>Attenuation (dB/50 ft)</th>
<th>Attenuation (dB/ft)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMR400UF</td>
<td>1575.42</td>
<td>6.3</td>
<td>3.2</td>
<td>---</td>
</tr>
<tr>
<td>S6001</td>
<td>1575.42</td>
<td>34.8</td>
<td>---</td>
<td>0.3</td>
</tr>
<tr>
<td>S6001</td>
<td>915</td>
<td>31.0</td>
<td>---</td>
<td>0.3</td>
</tr>
</tbody>
</table>

The rest of the gains and loss can be found in Table 8 for each component, with the exception being the gain for the USRP which will be set to 40 dB gain. A summarized list of the gains and losses are shown in Table 11. By adding the total gain of the system to the initial GPS signal power level of -130 dBm, the final signal power level of -41.0 dBm can be found.

Table 11: Total gain for the completed design.

<table>
<thead>
<tr>
<th>Component</th>
<th>Gain (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna</td>
<td>28.0</td>
</tr>
<tr>
<td>Amp/BPF</td>
<td>25.0</td>
</tr>
<tr>
<td>Large Coax Cable</td>
<td>-3.2</td>
</tr>
<tr>
<td>Bias Tee</td>
<td>-0.2</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.3</td>
</tr>
<tr>
<td>Downconverter</td>
<td>0</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.3</td>
</tr>
<tr>
<td>USRP</td>
<td>40.0</td>
</tr>
<tr>
<td><strong>Total Gain</strong></td>
<td><strong>89.0</strong></td>
</tr>
</tbody>
</table>
Noise Figure Analysis

The noise figure for the completed design will be calculated the same way it was for the conceptual design. When the correct values are plugged into the variables $F_1, F_2, F_3, \ldots, F_k$ and $G_1, G_2, G_3, \ldots, G_k$ of equation (5), the result will show that the noise figure for completed design will be 2.0 dB. Table 12 shows the summary of the noise figures for each component along with each components gain.

Table 12: Gain and noise figure for each component in completed design.

<table>
<thead>
<tr>
<th>Component</th>
<th>Gain (dB)</th>
<th>Noise Figure (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna</td>
<td>28.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Amp/BPF</td>
<td>25.0</td>
<td>3.8</td>
</tr>
<tr>
<td>Large Coax Cable</td>
<td>-3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>Bias Tee</td>
<td>-0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>Downconverter</td>
<td>0.0</td>
<td>8.4</td>
</tr>
<tr>
<td>Small Coax Cable</td>
<td>-0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>USRP</td>
<td>40.0</td>
<td>9.0</td>
</tr>
<tr>
<td><strong>System Noise Figure</strong></td>
<td></td>
<td><strong>2.0</strong></td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio

By using the same steps that were used to calculate the initial and final signal-to-noise ratios for the conceptual design, the initial and final SNRs can be found for the completed designs. The initial SNR will be the same since the same signal is being sought out, which is -19.03 dB. The final SNR should also be the same even though there is a new value for the total system gain and the downconverter was found to have no gain when down converting to 915 MHz. By plugging in -130 dBm for $S_i$, 89 dB for $G$, and 43 dB for $P$ into equation (12), the result will yield a final signal power level of 2 dBm. By plugging in -110.97 dBm for $N_i$, 2.0 dB for $F$, and 89 dB for $G$ into equation (15), the result will yield a final noise power level of -19.97 dBm. The difference between the final signal power level and the final thermal noise level yields a final SNR of 21.97 dB.
Cost Analysis

Table 13: Cost analysis for the chosen conceptual design.

<table>
<thead>
<tr>
<th>Component</th>
<th>Model #</th>
<th>Quantity</th>
<th>Cost per Component</th>
<th>Subtotal</th>
<th>Shipping &amp; Handling</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Antenna</td>
<td>GPSRSSMA</td>
<td>1</td>
<td>$25.00</td>
<td>$25.00</td>
<td>$9.48</td>
<td>$34.48</td>
</tr>
<tr>
<td>Adapter</td>
<td>SM4217</td>
<td>1</td>
<td>$17.00</td>
<td>$17.00</td>
<td>----</td>
<td>$17.00</td>
</tr>
<tr>
<td>Amp/BPF</td>
<td>58529A</td>
<td>1</td>
<td>$435.00</td>
<td>$435.00</td>
<td>$19.53</td>
<td>$454.53</td>
</tr>
<tr>
<td>Large Cable</td>
<td>LMR400UF</td>
<td>50 ft</td>
<td>$1.19/ft</td>
<td>$59.50</td>
<td>$11.25</td>
<td>$70.75</td>
</tr>
<tr>
<td>Cable Connectors</td>
<td>NS5875 w/installation</td>
<td>1</td>
<td>$26.95</td>
<td>$26.95</td>
<td>----</td>
<td>$26.95</td>
</tr>
<tr>
<td>Bias Tee</td>
<td>SB3500</td>
<td>1</td>
<td>$147.00</td>
<td>$147.00</td>
<td>$5.00</td>
<td>$152.00</td>
</tr>
<tr>
<td>Adapter</td>
<td>SM4215</td>
<td>1</td>
<td>$16.00</td>
<td>$16.00</td>
<td>----</td>
<td>$16.00</td>
</tr>
<tr>
<td>Small Cable</td>
<td>S6001</td>
<td>2</td>
<td>$10.50</td>
<td>$21.00</td>
<td>$18.00</td>
<td>$39.00</td>
</tr>
<tr>
<td>Downconverter</td>
<td>MAX2682</td>
<td>1</td>
<td>$0.00</td>
<td>$0.00</td>
<td>----</td>
<td>$0.00</td>
</tr>
<tr>
<td>Evaluation Board</td>
<td>MAX2682KIT</td>
<td>1</td>
<td>$90.00</td>
<td>$90.00</td>
<td>$15.00</td>
<td>$105.00</td>
</tr>
<tr>
<td>Local Oscillator*</td>
<td>Marconi 2022A</td>
<td>1</td>
<td>$549.99**</td>
<td>$549.99</td>
<td>----</td>
<td>$549.99</td>
</tr>
<tr>
<td>Small Cable</td>
<td>S60301</td>
<td>1</td>
<td>$12.50</td>
<td>$12.50</td>
<td>----</td>
<td>$12.50</td>
</tr>
<tr>
<td>USRP*</td>
<td>USRP1</td>
<td>1</td>
<td>$700.00</td>
<td>$700.00</td>
<td>----</td>
<td>$700.00</td>
</tr>
<tr>
<td>Daughter Board*</td>
<td>RFX900</td>
<td>1</td>
<td>$275.00</td>
<td>$275.00</td>
<td>----</td>
<td>$275.00</td>
</tr>
<tr>
<td>Personal Computer*</td>
<td>Average Dell</td>
<td>1</td>
<td>$500.00</td>
<td>$500.00</td>
<td>----</td>
<td>$500.00</td>
</tr>
<tr>
<td>Power Supply*</td>
<td>Heath 2718</td>
<td>1</td>
<td>$49.95</td>
<td>$49.95</td>
<td>----</td>
<td>$49.95</td>
</tr>
<tr>
<td><strong>Overall Total</strong></td>
<td><strong>$2,856.08</strong></td>
<td><strong>$2,924.89</strong></td>
<td><strong>$78.26</strong></td>
<td><strong>$3,003.15</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Components provided by IPFW.

**Price of used component.
Section III: Testing
**Verification of Test Equipment**

Before doing any testing of the design, the test equipment has to first be verified to make sure it is working properly. This is called calibration and is usually done every couple years. The actual time in-between calibrations depends on the applications in which it is used and also the accuracy that the equipment needs to abide by. Because calibration equipment is not accessible, a simple setup for each piece of equipment should show whether or not it is working correctly. The test equipment used is the HP 8590A Spectrum Analyzer.

**Testing the HP 8590A Spectrum Analyzer:**

In this test a RF generator will be set to a certain frequency and power level and then be connected to the HP Spectrum Analyzer. If the Spectrum Analyzer shows the same frequency and approximately the same power lever (attenuation through the coax could lower signal’s amplitude), both pieces of equipment will be assumed to be working correctly.

1. Set up the Narda Microline RF generator to 1.60542 GHz and 0 dBm.  
   *The RF generator had no control over the RF output power and was stuck at +13 dBm*
2. Connect a coax to the RF output and then connect the other end of the coax to the HP Spectrum Analyzer.
3. Measure the frequency and amplitude of the received signal.

   **Expected Result:** 1.60542 GHz; 13 dBm  
   **Measured Result:** 1.60542 GHZ ± up to 1 MHz; 13 dBm

Because the frequencies were not consistent between the two pieces of equipment, neither could be verified to be working correctly. Another RF generator will be incorporated into this test replacing the Narda Microline RF generator. If the frequency and approximate amplitude between these two pieces of equipment are the same, it will again be assumed that both pieces of equipment are working correctly. This would also prove that the Narda Microline was not working correctly.

1. Set up the Marconi Instruments Signal Generator to 1.0 GHz and 0 dBm.
2. Connect a coax to the RF output of the signal generate and then connect the other end of the coax to the HP Spectrum Analyzer.
3. Measure the frequency and amplitude of the received signal.

   **Expected Result:** 1.0 GHz; 0 dBm  
   **Measured Result:** 1.0 GHZ; 0 dBm

Because the frequencies were consistent between the two pieces of equipment, they were assumed to be working correctly. To double check the results, a HP 5340A Frequency Counter was used to test the Marconi 2022A and Narda Microline signal generators. When the Marconi was set to 1.0 GHz, the frequency counter displayed a solid 1.0000006 GHz signal. When the Narda Microline was set to 1.60542 GHz, the frequency counter displayed a frequency that would continuously jump several hundred kilohertz. This reinforced the assumption that the spectrum analyzer and Marconi were working and Narda Microline was not.
**Individual Component Testing**

Due to the fact that the only working RF generator goes up to only 1 GHz, testing of the SB3500 Bias Tee and MAX2682 Downconverter were done at ITT on calibrated test equipment.

**Testing the SB3500 Bias Tee:**

1. Connect 4.5 VDC from a power supply to the DC input on the SB3500 Bias Tee.
2. Measure the voltage at the RF/DC connector on the Bias Tee.

   **Expected Result:** 4.5 VDC  
   **Measured Result:** 4.5 VDC

3. Measure the voltage at the RF connector on the Bias Tee.

   **Expected Result:** 0 VDC  
   **Measured Result:** 0 VDC

4. Turn off the power supply and remove the cable supplying the voltage to the Bias Tee.
5. Set up a RF generator to 1.57542 GHz and connect the output to the RF/DC connector on the Bias Tee. Use a spectrum analyzer to measure the signal at the RF connector on the Bias Tee.

   **Expected Result:** Spike corresponding to 1.57542 GHz  
   **Measured Result:** Spike at 1.57542 GHz

6. Use a spectrum analyzer to measure the signal at the DC input connector on the Bias Tee.

   **Expected Result:** No spikes except for one possibly at baseband  
   **Measured Result:** Spike at baseband which is normal for spectrum analyzers

![Diagram of setup to test the bias tee.](image)

*Figure 16:* Setup to test the bias tee.
Testing the MAX2682 Downconverter:

1. Connect 4.5 VDC to the DC input on the MAX2682 Downconverter.
2. Set up a RF generator to 1.57542 GHz; -25 dBm and connect the output to the RF connector on the MAX2682 Downconverter.
3. Set up another RF generator to 660 MHz; -5 dBm and connect the output to the LO connector on the MAX2682 Downconverter.
4. Use a spectrum analyzer to measure the signal power level and frequency at the IF connector on the MAX2682 Downconverter.

**Expected Result:** Frequency of 915 MHz and gain of approximately 10 dB

**Measured Result:** Frequency of 915 MHz but zero gain

Further investigation showed that the reason for the lack of gain was because the RF input matching components on the MAX2682’s evaluation board were optimized for a 900 MHz signal and IF of 70 MHz. The USRP is capable of providing up to 90 dB of gain so having zero gain through the downconverter is acceptable.

---

**Figure 17:** Setup to test downconverter.
Testing the USRP board:

1. Setup the Marconi Instruments RF generator to 915 MHz and -30 dBm.
2. Connect the RF output to the RF input on the USRP RFX900 board.
3. Connect the USRP Board to the computer containing Ubuntu as its operating system.
4. Enter the following into the terminal window in Ubuntu to record 0.25 seconds of data.

   Usrp_rx_cfile.py –s –g 40 –f 915M –d 16 –RB –N 1M usrpdata.dat

5. Save this data to the flash drive.
6. Load the data on a computer with Windows.
7. Run the Matlab code located in Appendix D and have the input file be the USRP data file that was collected.
8. Repeat the test for power levels from -40 dBm to -120 dBm with 10 dB steps.

**Expected Result:** The I and Q plots should look like normal sine and cosine waves. The I verse Q plot should look like a thin walled donut, corresponding to a strong signal with accurate sampling.

**Measured Result:** Plotted results for this test are shown in Appendix E.

![Function Generator](image1.png) ![USRP](image2.png) ![PC](image3.png)

**Figure 18:** Setup to test USRP.
Testing the Completed Design

1. Connect the GPSRSSMA GPS Antenna coax cable to the RF in connector on the 58529A Amplifier.
2. Connect one end of the LMR400UF coax cable to the RF out connector on the 58529A Amplifier and the other end to the RF/DC connector on the SB3500 Bias Tee.
3. Connect a small coax from the RF connector on the SB3500 Bias Tee to the RF connector on the MAX2682 Downconverter.
4. Setup the Marconi Instruments RF generator to 660 MHz; 0 dBm and using a small coax connect the RF output to the LO connector on the MAX2682 Downconverter.
5. Connect a small coax cable from the IF connector on the MAX2682 Downconverter to the USRP board.
6. Set up the Heath 2718 Power Supply to 4.5 VDC and connect the output to the DC input on the MAX2682 Downconverter and also to the DC input on the SB3500 Bias Tee.
7. Connect the USRP board USB cable to the PC and then plug in the power supply for the USRP board.
8. Enter the following into the terminal window in Ubuntu to record 40 seconds of data.

   Usrp_rx_cfile.py -s -g 40 -f 915M -d 16 -RB -N 160M usrpdata.dat

9. Save this data to the flash drive.
10. Load the data on a computer with Windows.
11. Use SOX-14.3.0 software to filter data.
12. Modulate data using Matlab program “int8modulatebyfs4.m”
13. Process converted data using the GPS software provided by the book “A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach. The calculated coordinates should match the coordinates that are given by a Garmin GPS unit that will be placed next to the GPSRSSMA GPS Antenna on the roof.
**Figure 19:** Setup to test final design.

**Final Design Results**

**Table 14:** Ten test runs and their respective calculated coordinates along with the standard deviation. Also shown is the average position.

<table>
<thead>
<tr>
<th>Test Run</th>
<th>Latitude (Time)</th>
<th>Longitude (Time)</th>
<th>Latitude (Decimal)</th>
<th>Longitude (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41° 6' 56.5312&quot;</td>
<td>-85° 6' 34.3767&quot;</td>
<td>41.115703</td>
<td>-85.109549</td>
</tr>
<tr>
<td>2</td>
<td>41° 6' 56.5822&quot;</td>
<td>-85° 6' 34.3415&quot;</td>
<td>41.115717</td>
<td>-85.109539</td>
</tr>
<tr>
<td>3</td>
<td>41° 6' 56.6707&quot;</td>
<td>-85° 6' 34.8112&quot;</td>
<td>41.115742</td>
<td>-85.109670</td>
</tr>
<tr>
<td>4</td>
<td>41° 6' 56.3278&quot;</td>
<td>-85° 6' 34.6711&quot;</td>
<td>41.115647</td>
<td>-85.109631</td>
</tr>
<tr>
<td>5</td>
<td>41° 6' 56.6504&quot;</td>
<td>-85° 6' 34.2496&quot;</td>
<td>41.115736</td>
<td>-85.109514</td>
</tr>
<tr>
<td>6</td>
<td>41° 6' 56.6370&quot;</td>
<td>-85° 6' 33.2720&quot;</td>
<td>41.115733</td>
<td>-85.109242</td>
</tr>
<tr>
<td>7</td>
<td>41° 6' 56.7945&quot;</td>
<td>-85° 6' 34.5746&quot;</td>
<td>41.115776</td>
<td>-85.109604</td>
</tr>
<tr>
<td>8</td>
<td>41° 6' 56.3488&quot;</td>
<td>-85° 6' 33.2720&quot;</td>
<td>41.115652</td>
<td>-85.109242</td>
</tr>
<tr>
<td>9</td>
<td>41° 6' 56.5213&quot;</td>
<td>-85° 6' 34.0373&quot;</td>
<td>41.115700</td>
<td>-85.109455</td>
</tr>
<tr>
<td>10</td>
<td>41° 6' 56.8378&quot;</td>
<td>-85° 6' 35.0228&quot;</td>
<td>41.115788</td>
<td>-85.109729</td>
</tr>
</tbody>
</table>

Actual Location  
41.115817  -85.109572

Average  
41.1157194  -85.1095441

Standard Deviation  
4.39504E-05  0.000127531

Distance Between Average and Actual Location  
2.5 m
Figure 20: This shows the plotted coordinates for the 10 test runs and also the average position in blue. The run average distance is 2.5 m.
Evaluation

The obvious first thing to take from this design is the fact that it is able to meet the goals set out for this project. Those completed goals include the ability to capture and digitize the GPS signal and to use software to compute the location of the GPS receiver. Even though not all of the calculated results are within the 10 m goal, the average of ten calculated positions is about 2.5 m, which is within the goal.

The time it takes to process the digitized data is relatively long when compared to on the market GPS receivers, but that has more to do with Matlab processing the data than anything else. With the large amount of data that is being processed (610.4 MB), Matlab is not the best suited software to do the heavy calculations.

Another issue to take note of is the cost of the design. When the overall cost of the design includes the components that were loaned by IPFW, it exceeds the $2000.00 limit set by Raytheon. Having a three thousand dollar GPS receiver is neither practical nor cost effective.

The size of the design is also an issue. Because a desktop computer and a signal generator are used in this design, a restriction to the location of the receiver inevitable. Having a portable GPS receiver was not a requirement for this design, but having it able to easily relocate would have made it a more practical receiver.

Recommendations

One way to improve this design is to use a Windows driver to communicate with the USRP Board. If a Windows driver is used, then there would be no use for a Linux desktop. Plus, there would be no need to use a flash drive or to reboot the desktop to transfer the data to Matlab. This would reduce the post processing time by about 3 minute since Matlab would be ready to read the data right away.

By using a DBSRX Daughter Board in the USRP Board, the RFX900 Daughter Board, local oscillator, and downconverter can be eliminated from the design. This is because the DBSRX Daughter Board has a receiving range of 800 MHz to 2.4 GHz\textsuperscript{[20]}. Also, the overall cost of the design would be reduced by $675.00. Having less focus on the hardware will yield more opportunities to improve the software portion of the design. One improvement could be reducing the tracking time of the satellites which takes about 6 minutes to do. Also, the GPS software could be written in C/C++ which is faster than Matlab when processing large quantities of data.
Conclusion

This design is relatively large in size and consists of many different parts. Having such a large design when compared to GPS receivers out in the market is inevitable because of limitations in time, technology, budget, and education. It does however succeed in its overall objective: the extraction of the navigation data from a GPS signal and the application of software to calculate the receiver’s location. Even though the individual hardware components and most of the software is either borrowed or purchased, there was sufficient complexity and challenges in building this working software defined GPS receiver. Such tasks include the interfacing of each component, testing of components, and the programming of the USRP Board. Matlab programming was also required to up-convert the GPS data for the GPS software that extracts the navigation data and calculates the receiver’s position.
References


Appendix A: Steps to Create Ubuntu 9.04 with GNU Radio on USB

Creating Ubuntu 9.04 Persistent Live USB

In Windows

- Download **HP Format Tool** (*HPFormatTool.exe*)
- Download **Helper Program** (*Ubuntu904P.exe*)
- Download the 4 GB version **casper-rw** (*4GB-casper-rw.zip*)
- Download the **Ubuntu 9.04 iso** (*ubuntu-9.04-desktop-i386.iso*)
- Format USB Flash Drive
  - Double click *SP27608.exe*
  - Follow installation instructions
  - Go to **Program Files > DriveKey**
  - Right click *HPUSBFW*
  - Click **Run as Administrator**
  - Make sure **Device** is correct USB Flash Drive
  - Make sure **File System** is **FAT32**
  - Make sure Label is optional
  - Make sure boxes are unchecked
  - Click **Start**
- Double Click *Ubuntu904P.exe* to extract folder
- Right Click *4GB-casper-rw.zip* and extract
- Place the **Ubuntu 9.04 iso** into the *Ubuntu904P* folder
- Replace the **casper-rw** that is in the *Ubuntu904P* folder with the **casper-rw** that was extracted from *4GB-casper-rw.zip*
- Install into your USB Flash Drive
  - Double Click *Ubuntu.bat*
  - Enter **drive letter** of USB Flash Drive (VERY IMPORTANT)
  - Press **Enter**
  - This process may take some time
  - Press **Enter** to confirm bootable option (MAKE SURE DRIVE LETTER FOR USB IS CORRECT)
- Access Flash Drive
  - Right Click *Makeboot.bat*
  - Click **Run as Administrator**
  - Make sure drive letter is correct and press **Enter**
  - Do this every time flash drive is put in a new USB port
- To boot up from the USB press **F9** (Laptop) or **F12** (Desktop) when the computer is starting up
Installing GNU Radio Companion[19]

In Ubuntu

- Access Internet
- Go to System > Administration > Software Sources
  - Pick Ubuntu Software tab
  - Make sure all sources are checked
  - Pick United States as server in Download From pull down
  - Pick Third Party Software tab
  - Click Add
  - In the APT line enter: deb http://gnuradio.org/ubuntu stable main
  - Click Add Source
  - Click Close
  - Click Reload
  - Repeat for deb-src http://gnuradio.org/ubuntu stable main
- Open the Terminal and enter the following in the home directory
  - sudo aptitude update
  - sudo aptitude install gnuradio gnuradio-companion
- To provide regular access to USRP
  - sudo addgroup ubuntu usrp
  - Reboot Ubuntu
- To verify USRP is being recognized enter the following in the home directory
  - ls -lR /dev/bus/usb | grep usrp
    - Output: crw-rw---- 1 root usrp 189, 514 Mar 24 09:46 003
- To run GNU Radio enter grc in the home directory to start GNU Radio Companion
Appendix B: File Used to Record Data from USRP Board

#!/usr/bin/env python

"""
Read samples from the USRP and write to file formatted as binary
outputs single precision complex float values or complex short values (interleaved 16 bit signed
short integers).
"""

from gnuradio import gr, eng_notation
from gnuradio import audio
from gnuradio import usrp
from gnuradio.eng_option import eng_option
import sys

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

        usage="%prog: [options] output_filename"
        parser = OptionParser(option_class=eng_option, usage=usage)
        parser.add_option("-R", "--rx-subdev-spec", type="subdev", default=(0, 0),
            help="select USRP Rx side A or B (default=A)"
        )
        parser.add_option("-d", "--decim", type="int", default=16,
            help="set fpga decimation rate to DECIM [default=+default]"
        )
        parser.add_option("-f", "--freq", type="eng_float", default=None,
            help="set frequency to FREQ", metavar="FREQ"
        )
        parser.add_option("-g", "--gain", type="eng_float", default=None,
            help="set gain in dB (default is midpoint)"
        )
        parser.add_option("-s", "--width-8", action="store_true", default=False,
            help="Enable 8-bit samples across USB"
        )
        parser.add_option( "--no-hb", action="store_true", default=False,
            help="don't use halfband filter in usrp"
        )
        parser.add_option( "-s", "--output-shorts", action="store_true", default=False,
            help="output interleaved shorts in stead of complex floats"
        )
        parser.add_option("-N", "--nsamples", type="eng_float", default=None,
            help="number of samples to collect [default=+inf]"
        )

        (options, args) = parser.parse_args ()
        if len(args) != 1:
            parser.print_help()
            raise SystemExit, 1
        filename = args[0]

        if options.freq is None:
            parser.print_help()
            sys.stderr.write('You must specify the frequency with -f FREQ\n');
            raise SystemExit, 1

        # build the graph
        if options.no_hb or (options.decim<8):
            #Min decimation of this firmware is 4. contains 4 Rx paths without halfbands and 0 tx
            #paths.
            self.fpga_filename="std_4rx_0tx.rbf"
            if options.output_shorts:
                self.u = usrp.source_s(decim_rate=options.decim,fpga_filename=self.fpga_filename)
            else:
                self.u = usrp.source_c(decim_rate=options.decim,fpga_filename=self.fpga_filename)
        else:
            #standard fpga firmware "std_2rxb_2tx.rbf" contains 2 Rx paths with halfband filters
            #and 2 tx paths (the default) min decimation 8
            if options.output_shorts:
                self.u = usrp.source_s(decim_rate=options.decim)
            else:
                self.u = usrp.source_c(decim_rate=options.decim)

        if options.width_8:
            sample_width = 8
            sample_shift = 8
            format = self.u.make_format(sample_width, sample_shift)

        if len(filename) > 0:
            try:
                data = self.u
                file = open(filename, 'w')
                file.write(data)
```python
r = self.u.set_format(format)
if options.output_shorts:
    self.dst = gr.file_sink(gr.sizeof_short, filename)
else:
    self.dst = gr.file_sink(gr.sizeof_gr_complex, filename)
if options.nsamples is None:
    self.connect(self.u, self.dst)
else:
    if options.output_shorts:
        self.head = gr.head(gr.sizeof_short, int(options.nsamples)*2)
    else:
        self.head = gr.head(gr.sizeof_gr_complex, int(options.nsamples))
    self.connect(self.u, self.head, self.dst)

if options.rx_subdev_spec is None:
    options.rx_subdev_spec = usrp.pick_rx_subdevice(self.u)
self.u.set_mux(usrp.determine_rx_mux_value(self.u, options.rx_subdev_spec))

# determine the daughterboard subdevice we're using
self.subdev = usrp.selected_subdev(self.u, options.rx_subdev_spec)
print "Using RX d'board %s" % (self.subdev.side_and_name(),)
input_rate = self.u.adc_freq() / self.u.decim_rate()
print "USB sample rate %s" % (eng_notation.num_to_str(input_rate))

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

print "Gain Range: Min(%s)dB Max(%s)dB Step(%s)" % (g[0], g[1], g[2])

r = self.u.tune(0, self.subdev, options.freq)
if not r:
    sys.stderr.write('Failed to set frequency\n')
    raise SystemExit, 1

if __name__ == '__main__':
    try:
        my_top_block().run()
    except KeyboardInterrupt:
        pass
```
Appendix C: Matlab Code Used to Up-Convert GPS Data

```matlab
% int8modulatebyfs4.m
% Upmodulate by fs/4
% Create a real signal by upconverting by a carrier of freq fs/4 using the %formula:
% mixer output = Real(data)*( cos(2*pi*n*fm/fs) ) - Imag(data)*(sin(2*pi*n*fm/fs)
disp('Trims off first block of 2000 samples')
% Number of complex samples in a block
blksize = 1000;
% In Hertz
samplingFreq = 4.0e6;
carrierOffset = 4e3;
phase = 0:(blksize-1);
angleVal = 2*pi.*phase/blksize;
% modulation = exp(j*angleVal);
% Where the cos values are [ 1 0 -1 0 ] repeating pattern
% Where the sin values are [ 0 1 0 -1 ] repeating pattern

cos4 = [ 1 0 -1 0 ];
sinv4 = [ 0 1 0 -1 ];
modcosine = [];
modsine = [];
maximumV = 0;
minimumV = 0;
% Replicate sine & cosine cycles
for i=1:blksize/4;
    % Repeat cosine wave
    modcosine = [ modcosine cos4 ];
    % Repeat sine wave
    modsine = [ modsine sinv4 ];
end
fileInput = 'C:\ProcessData\gpslpf.dat';
fileOutput = 'C:\ProcessData\gpsdata07.dat';
disp( 'Processing input file ' fileInput )
% Open file handle
fin = fopen(fileInput,'rb');
% Open file handle
fout = fopen(fileOutput,'wb');
% Initial value for while statement
samplesReadx2 = 2*blksize;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read in first block, drop samples, and gets rid of transients
[rwSignalx2, samplesReadx2] = fread(fin, 2*blksize, 'int16');
if (samplesReadx2 ~= 2*blksize)
    disp('Cannot read first block of data - exiting program!')
    fclose(fin);
    fclose(fout);
    return
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reads input file and finds the maximum and minimum values
while ( (samplesReadx2 == 2*blksize) )
    [rwSignalx2, samplesReadx2] = fread(fin, 2*blksize, 'int16');
    if (samplesReadx2 == 2*blksize)
        disp('No more full blocks of data')
        disp('Breaking out of first "while" loop!')
        disp('Minimum')
        disp(minimumV)
        disp('Maximum')
        disp(maximumV)
        break
    end
end
close(fin);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
samplesRead = samplesReadx2/2;
% Extract even samples as I
rawSignalI = rawSignalx2(1:2:(2*blksize));
% Extract odd samples as Q
rawSignalQ = rawSignalx2(2:2:(2*blksize));
% Transpose vector ( Note do not take complex conjugate )
rawSignalI = rawSignalI.';
% Transpose vector ( Note do not take complex conjugate )
```

50
rawSignalI = rawSignalI';
% Complex mix up by fs/4 - Create a real waveform of blksize samples
outputWaveform = rawSignalI .* modcosine - rawSignalQ .* modsine;
localmax = max(outputWaveform);
maximumV = max([localmax maximumV]);
localmin = min(outputWaveform);
minimumV = min([localmin minimumV]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reopens and reloads file from the beginning
% Read in first block, drop samples, and gets rid of transients again
disp('Reloading File')
disp('Trims off first block of 2000 samples again')
disp([ 'Creating output file ' fileOutput ])
fseek(fin,0,-1);
[rawSignalx2, samplesReadx2] = fread(fin, 2*blksize, 'int16');
if (samplesReadx2 ~= 2*blksize)
disp('Cannot read first block of data - exiting program!')
fclose(fin);
fclose(fout);
return
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reads input file and writes the modulated I and Q values into output file
while ( (samplesReadx2 == 2*blksize) )
[rawSignalx2, samplesReadx2] = fread(fin, 2*blksize, 'int16');
if (samplesReadx2 ~= 2*blksize)
disp('No more full blocks of data - exiting program!')
fclose(fin);
fclose(fout);
disp('Minimum')
disp(minimumV)
disp('Maximum')
disp(maximumV)
return
end
samplesRead = samplesReadx2/2;
% Extract even samples as I
rawSignalI = rawSignalx2(1:2:(2*blksize));
% Extract odd samples as Q
rawSignalQ = rawSignalx2(2:2:(2*blksize));
% Transpose vector ( Note do not take complex conjugate )
rawSignalI = rawSignalI';
% Transpose vector ( Note do not take complex conjugate )
rawSignalQ = rawSignalQ';
% Complex mix up by fs/4 - Create a real waveform of blksize samples
outputWaveform = rawSignalI .* modcosine - rawSignalQ .* modsine;
[count ] = fwrite(fout, outputWaveform/((max([maximumV abs(minimumV)]))/127) , 'int8');
% Check if 2*blksize samples written
if (count ~= 2*blksize)
disp('Not able to write the specified number of samples for tracking - exiting!')
fclose(fin);
fclose(fout);
disp('Minimum')
disp(minimumV)
disp('Maximum')
disp(maximumV)
return
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Closes files and displays the maximum and minimum
fclose(fin);
fclose(fout);
disp('Minimum')
disp(minimumV)
disp('Maximum')
disp(maximumV)
% Simple code used to test digitized data from USRP
fin = fopen('usrp110.dat','rb');
y = fread(fin, 10000, 'short');

I = y(1001:2:length(y));
Q = y(1002:2:length(y));

subplot(1,3,1);
plot(I,'.');
title('I Data: -110 dB');
subplot(1,3,2);
plot(Q,'.');
title('Q Data: -110 dB');
subplot(1,3,3);
plot(I,Q,'.');
title('Q vs I: -110 dB');
xlabel('I Data');
ylabel('Q Data');
fclose(fin);
Appendix E: Plot Results for USRP Test

Figure 21: I and Q plots for generated 915 MHz signal set at -30 dBm.

Figure 22: I and Q plots for generated 915 MHz signal set at -40 dBm.
Figure 23: I and Q plots for generated 915 MHz signal set at -50 dBm.

Figure 24: I and Q plots for generated 915 MHz signal set at -60 dBm.
Figure 25: I and Q plots for generated 915 MHz signal set at -70 dBm.

Figure 26: I and Q plots for generated 915 MHz signal set at -80 dBm. This is the estimated power level of the GPS signal when it enters the USRP Board. The actual GPS signal will look more like Figure 29 and Figure 30 since the signal is below thermal noise level.
Figure 27: I and Q plots for generated 915 MHz signal set at -90 dBm.

Figure 28: I and Q plots for generated 915 MHz signal set at -100 dBm.
Figure 29: I and Q plots for generated 915 MHz signal set at -110 dBm.

Figure 30: I and Q plots for generated 915 MHz signal set at -120 dBm