Oblique Ionosonde Receiver Station

Capstone Design - Final Report

New Mexico Institute of Mining and Technology

Los Alamos National Laboratory

Abie Maez, Jaden Keener, Karl Lukes

May 5th, 2023

Contents

1 Abstract **4**

2 Introduction **4**

2.1 How To Use This Document . . . 4
2.2 Goal Statement . . . 4
2.3 Project Background and Motivation . . . 4
2.4 Technical Overview . . . 4
2.5 Project Requirements . . . 5

3 Hardware Configuration **5**

3.1 Primary Antenna . . . 6
3.1.1 Amplification . . . 6
3.2 Power Supply . . . 6
3.3 Software Defined Radio . . . 6
3.4 Computer . . . 7

4 Software Configuration - Third Party **7**

4.1 Operating System . . . 7
4.2 Python . . . 7
4.3 GNU Radio . . . 7
4.4 UHD . . . 8
4.5 Python Packages . . . 8
4.6 gqrx . . . 8

5 Software Configuration - Custom Software **8**

5.1 Data Capture . . . 9
5.1.1 Goal . . . 9
5.1.2 Implementation . . . 9
5.1.3 Output . . . 9
5.2 Data Processing (Heterodyne Mixing) . . . 10
5.2.1 Goal . . . 10
5.2.2 Implementation . . . 10
5.2.3 Output . . . 11
5.3 Visualization . . . 12
5.3.1 Goal . . . 12
5.3.2 Implementation . . . 12
5.3.3 Output . . . 13

6 User Guide **14**
6.1 What is ChirpRX? . . . 14
6.2 USRP Connectivity . . . 14
6.3 Setting up and using gqrx . . . 16
6.4 Identifying an Ionosonde . . . 17
6.5 Using ChirpRX . . . 18
6.5.1 Creating Timelapses . . . 19
6.6 Troubleshooting and Known Issues . . . 19
6.6.1 No devices found for Empty Device Address . . . 20
6.6.2 [WARNING] [RFNOC::GRAPH] One or more blocks timed out during flush! . . . 20
6.6.3 'O' in console output . . . 20
6.6.4 usrp_source :error: In the last x ms, y overflows occurred. . . . 20
6.6.5 buffer_double_mapped :warning: . . . 20
6.6.6 Why is the high end of my spectrum cutoff? . . . 20

7 Results **21**
7.1 Catalog and Searching . . . 21
7.2 Ionograms . . . 21
7.3 Ionosphere Layers . . . 22
7.4 Ionogram Animation . . . 22
7.4.1 Timelapse 'Drift' . . . 24

8 Budget **24**

9 Conclusion **24**
9.1 Summary of Project Goals . . . 24
9.2 Future Work . . . 25

Appendices **27**

A Bill of Materials **27**

B Ionosonde Reception Catalog **27**
B.1 Known Transmitters . . . 27
B.2 Received Transmissions List . . . 28

1 Abstract

The goal of this capstone design project was to build an oblique Ionosonde receiver station that can detect oblique Ionosonde transmissions and create Ionograms from the collected data. Although only one reliable set of transmitters was found, the team was successful in capturing and processing data into Ionograms and Ionogram animations. This document gives a brief overview of the operating principle of an oblique Ionosonde, provides documentation related to the hardware and software used to build the station, provide instructions on how to commission and operate the station, and presents the results of the data collected with the receiver station.

2 Introduction

2.1 How To Use This Document

The goal of this document is twofold. First, this document provides system information and general commissioning as well as operating instructions that may aid in the re-creation and operation of similar stations. Second, this document presents results and data collected from the original receiver station.

2.2 Goal Statement

Design and develop a receiver to detect oblique Ionosonde transmissions, then catalog observed transmitters, and use automated data collection to record a series of Ionosonde transmissions. Use these recordings to create a set of Ionograms showing the heights of the different layers in the ionosphere varying over time.

2.3 Project Background and Motivation

The ionosphere is a layer of the earth's atmosphere which resides from about 80 to 600 kilometers above the surface of the earth [3]. A high concentration of free electrons and ions makes the ionosphere reflective to radio waves. However, the reflectivity of the ionosphere is always changing, and these changes can affect satellite and terrestrial communications, as well as atmospheric and space research. Because of this, measurements of the reflectivity of the ionosphere are useful from a communications and research standpoint.

The goal of the design team is to build a system to evaluate what frequency ranges of radio waves are reflected, at what times the reflections occurred, and how these two variables change over time.

2.4 Technical Overview

An Ionosonde transmits a range of radio frequencies and listens for their return to the surface of the earth. The height at which a given frequency was reflected can be calculated by multiplying the speed of light in the atmosphere by the time between when a transmission was sent and when the reflected signal was heard. This travel time/travel distance can be plotted versus frequency in an Ionogram.

This project focuses on oblique Ionosondes, where the transmission and reception locations are different, and the transmitter frequency sweeps linearly and usually encodes no data.

Oblique Ionosondes rely on an operation methodology that has decreased in popularity in recent years as newer digital systems replace oblique Ionosondes [1]. This, combined with a lack of documentation meant that signal reception was not guaranteed. Even if transmissions could be received, factors including signal integrity, consistency, and transmission schedule ultimately dictate the feasibility of creating an Ionogram.

2.5 Project Requirements

A list of deliverables requested by the project sponsor are shown below:

- A catalog listing station transmissions that were received by the station
- A Bill of Materials, including the MSRP of each item
- System design drawings
  - Hardware schematic
  - Block diagram of the signal processing chain
- A report describing project methodology and results (this document)
- Spectrograms/Ionograms showing ionospheric conditions over time
- Creation of a time-based Ionogram animation

Other project requirements requested by the project sponsor and or the design course instructor are listed below:

- All data delivered to the sponsor shall be packaged in HDF5 format
- Any custom software written to accomplish the goals of the project shall be written with UNIX compatibility
  - Furthermore, all Python code written shall conform to the PEP-8 style guide

3 Hardware Configuration

This receiver station is composed of three main components, namely an antenna, software-defined radio, and a computer. All of the hardware components mentioned below are listed in the BOM (Appendix A).

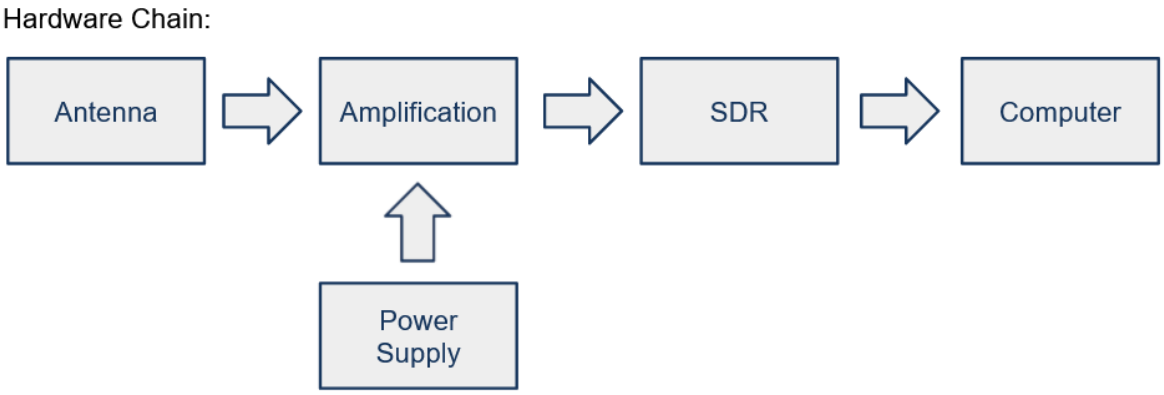

Figure 1: Hardware Chain

3.1 Primary Antenna

This receiver station uses an active Long Wavelength Array (LWA) antenna, designed primarily for stellar radio research. The antenna consists of two pairs of flared and down-turned dipole blades, which allow for a wideband operating range of about 5 to 90 MHz [4]. This dipole design also provides a wide-view skyward radiation pattern, which is useful for capturing Ionosonde signals. This project only requires use of one dipole pair: as the received signals will be circularly polarized there is no need to make use of two perpendicular dipoles.

The antenna comes standard with an onboard balun and amplifier, which requires the use of a bias tee to supply power during normal operation. More information about the operation and design of the LWA antenna can be found here.

3.1.1 Amplification

The project required front end signal amplification to ensure signal integrity on the long coaxial connection between the roof mounted antenna and the lab mounted SDR. As previously mentioned, the LWA antenna comes with a front-end amplifier with approximately 35dB of gain.

3.2 Power Supply

The antenna amplifier and front-end circuitry are powered using a variable linear DC power supply at +15 VDC through a bias tee, connected directly to the main antenna coaxial line. A fixed-voltage power supply would also be adequate for powering the antenna front-end electronics.

3.3 Software Defined Radio

The SDR chosen was the Ettus Research USRP X310. This SDR allows for adjustable sampling rates up to 200MSps, which provides ample bandwidth for the wideband recording necessary for Ionosonde

capture. The BasicRX daughterboard was used, which allows the SDR to operate in the 0-250MHz range where we expect to see Ionosondes.

3.4 Computer

The minimum required specifications for the computer were the read and write speeds of the drive(s). The chosen computer should have atleast one disk capable of 100MB/s write speeds (for 12.5MSps capture). Though exact specifications have not been calculated, the computer should also be modern enough to handle the moderately heavy DSP requirements of this project.

4 Software Configuration - Third Party

This section will discuss all third-party software and dependencies required for this project. This includes the operating system, drivers, packages, etc. The next section will discuss any custom software created for this project. **Please read this section carefully**, as it provides instructions for installing all dependencies for this project.

4.1 Operating System

This project was developed on **Ubuntu 22.04.1 LTS**. Compatibility with other Linux distributions is possible but has not been tested. Please follow the instructions for installing Ubuntu on their website.

4.2 Python

All custom software for this project is written in Python. Our system runs **Python 3.10.6**. The project has not been evaluated on any other Python versions, but it is likely that any version of Python 3 equal to or newer than 3.10.6 will work. Instructions for installing Python can be found on their website.

4.3 GNU Radio

GNU Radio is a UNIX-based development toolkit for working with software defined radios. It provides a fast and efficient DSP engine with many predefined signal processing blocks, and is the framework used to develop all custom software for this project. GNU Radio also features a GUI helper (GNU Radio Companion) that provides a visual tool for creating signal processing flowcharts. This tool is used to open any file with the *.grc* extension in this project. This project was developed on **GNU Radio version 3.10.1.1**.

To install GNU Radio on Ubuntu 22 use the following command

```
$ sudo apt-get install gnuradio
```

This will also install all of the dependencies required for GNU Radio.

4.4 UHD

UHD is the hardware driver for the Ettus USRP brand SDRs. This project was developed with **UHD_4.1.0.5-3**. Without it, the host machine is not capable of communicating with the SDR. To install UHD on Ubuntu 22 use the following command:

```
$ sudo apt-get install uhd-host
```

4.5 Python Packages

Most Python packages necessary for this project are automatically installed as dependencies when installing GNU Radio. The only package which is not automatically installed is *ffmpy*. *ffmpy* is a Python wrapper for FFmpeg, and is how we convert our still images into a timelapse in both '.gif' and '.mov' form. To install *ffmpy*, run the following command:

```
$ pip install ffmpy
```

You must also install FFmpeg for ffmpy to function. Please install FFmpeg with the following command:

```
$ sudo apt-get install ffmpeg
```

4.6 gqrx

Gqrx is a free and open-source spectrum browsing tool built in GNU Radio. It provides an easy way to observe and characterize Ionosondes before automatic capture can begin. The detailed use of gqrx as it relates to this project can be found in the user guide. To install gqrx, run the command:

```
$ sudo apt-get install gqrx
```

Note that gqrx requires GNU Radio to be installed to run.

5 Software Configuration - Custom Software

This section will discuss the custom signal processing software developed for this project. Signal Processing for this project is done in three discrete blocks: recording, processing, and visualization. The high level flowchart for these blocks can be seen in figure 2. This section will detail the operation of each block and its interactions with other blocks.

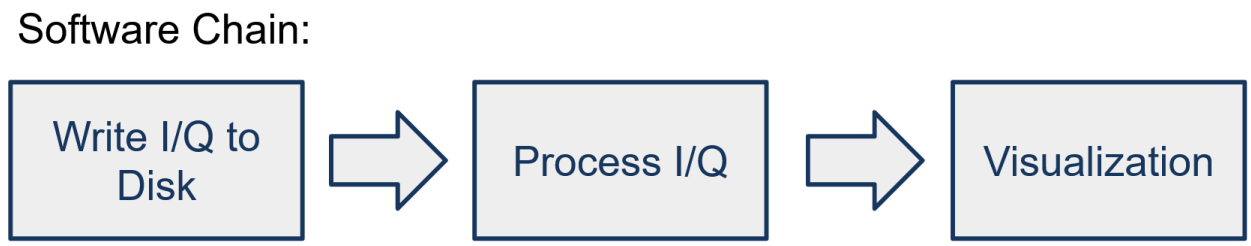

Figure 2: High Level Software Flowchart

5.1 Data Capture

5.1.1 Goal

The first block in the software chain is data capture. The goal of this step is to write the raw SDR data to the disk while the Ionosonde is transmitting. Capturing all data would result in terabytes of unusable data, so it is desirable to have a precise method of only capturing during transmission. This section will *not* cover how to operate these blocks, only their working principles. For a user guide, please see section 6. The internal flowchart for the data capture block can be found in figure 3.

5.1.2 Implementation

In order to capture data only while the Ionosonde is transmitting, we make the assumption that the Ionosonde repeats on a consistent schedule. For example, it starts at 13:00:00 and broadcasts across 10MHz at 100kHz/s, repeating every 12 minutes. The data capture block accepts these parameters (start time, bandwidth, slope, and period) and automatically captures the Ionosonde transmissions. As previously mentioned, this approach *only* works on repeatable and consistent Ionosondes. For such an Ionosonde, the timer is theoretically capable of infinite automatic capture.

The block does not accept a date input, so any time value given is assumed to be on the current date. Time values must be given in 24-hour format. If a time value is given that is in the past, the block will forward calculate the next expected transmission time. After every transmission, the Ionosonde period is added to this time.

5.1.3 Output

The recorded data files are of type ".RAW", where the filename is [DATE]_[TIME].RAW. A file that began recording at 2pm sharp on Jan 3, 2023 would therefore have the filename "20230103_140000.RAW". The data stored in these files is raw I/Q binary data from the SDR where each sample is represented as a 64-bit complex number. There is no delimiter between samples. The NumPy "np.complex64" datatype is excellent for reading this data.

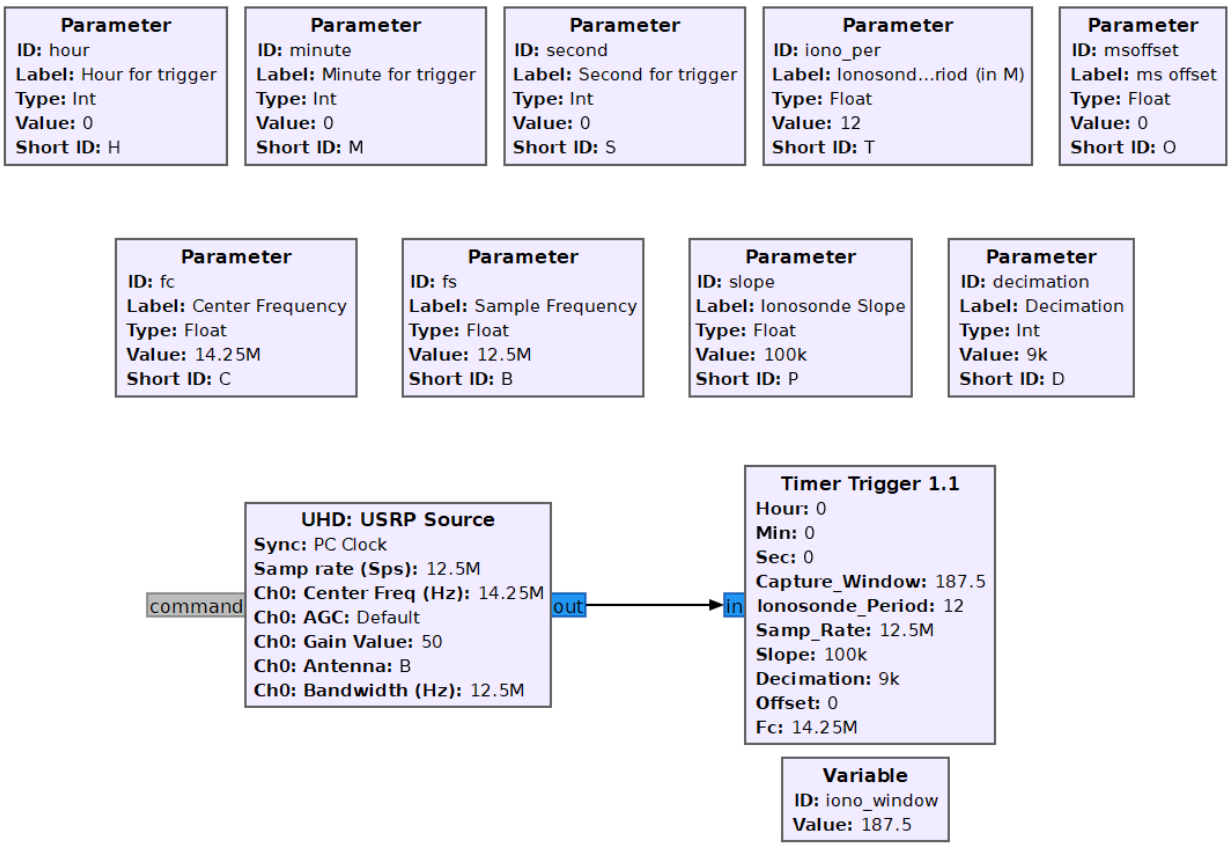

Figure 3: Data Capture Flowchart. See TimerTrigger.grc for more details.

5.2 Data Processing (Heterodyne Mixing)

5.2.1 Goal

After data capture, the data must be processed into a form that is capable of generating an Ionogram. An Ionogram is a spectrogram (x-axis frequency, y-axis time) where the plot is normalized in time to the slope of the Ionosonde, such that a received signal that exactly matches the expected slope would appear as a flat horizontal line. This normalization allows us to see any deviation from the expected slope, revealing changes in reflection height. The internal flowchart for the data processing block can be found in figure 4.

5.2.2 Implementation

We perform this normalization through a process called “Heterodyne Mixing”. This process relies on the principle that the product of two sinusoids is one sinusoid with the sum of the first two’s frequency and another with the difference. By multiplying our captured Ionosonde signal with a local oscillator that increases in frequency at the same slope as the Ionosonde we can extract a frequency difference

term that shows any deviation in the Ionosonde's reflection height.

The I/Q data captured by the SDR can be represented as a complex number where the in-phase component is the real part of the number and the quadrature component is the imaginary part. Using Euler's identity we can simplify this representation into the complex exponential $e^{j\omega_{SDR}}$. Our local oscillator will also be represented as a complex exponential, $e^{j\omega_{LO}}$. We can therefore multiply our SDR signal by the complex conjugate of our Local Oscillator (LO) signal to receive only the difference term: $e^{j\omega_{SDR}} * e^{-j\omega_{LO}} = e^{j(\omega_{SDR}-\omega_{LO})}$.

Our "LocalChirp" block performs this operation. By feeding the block the sample rate and expected Ionosonde slope we are able to generate a vector that represents our LO signal. We then perform the conjugate multiplication on a sample-by-sample basis as we read the recorded data from the timer trigger block. After mixing we will be left with a difference term near 0Hz, allowing us to lowpass filter and decimate the signal for more size-efficient storage. The lowpass filter size is automatically calculated based on the sample rate and decimation. The sample rate, slope, and desired decimation parameters are passed into this block automatically when it is called by "TimerTrigger" after a recording has been captured.

5.2.3 Output

The mixed and decimated signal is stored in a ".chirp" file with the same name as the original ".RAW" file generated by TimerTrigger. This file is also stored as I/Q data (64-bit complex numbers).

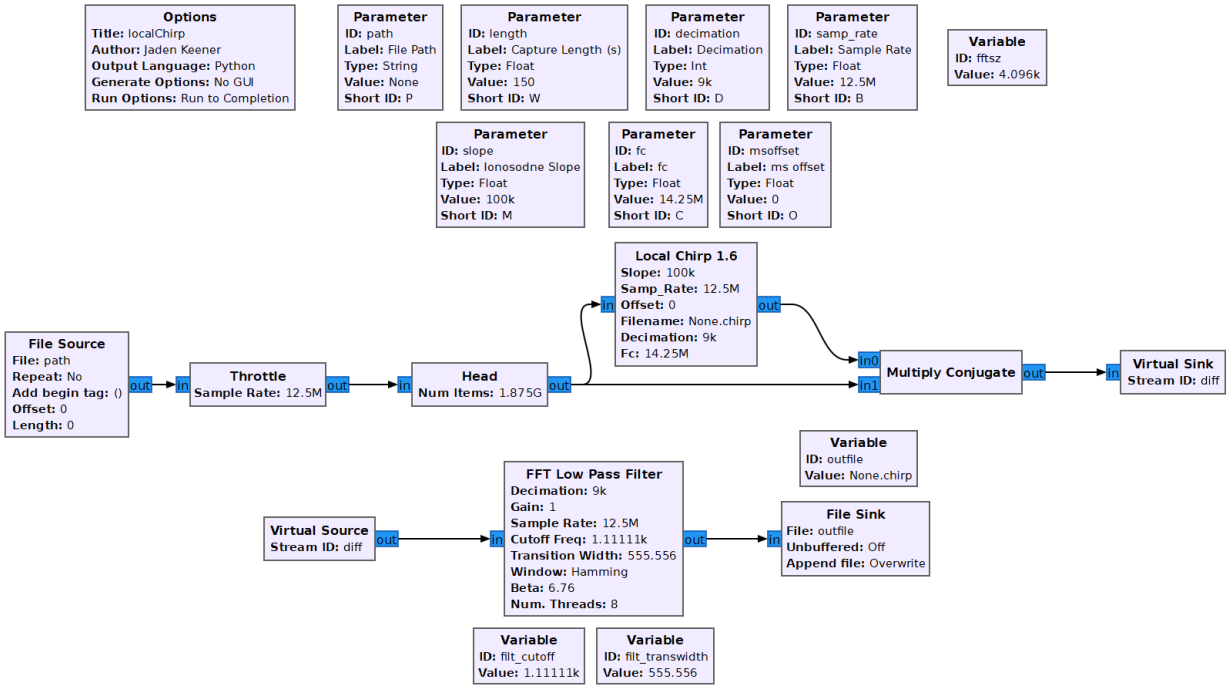

Figure 4: Data Processing Flowchart. See LocalChirp.grc for more details.

5.3 Visualization

5.3.1 Goal

The last step in our signal processing chain is to visualize the data into an Ionogram. As previously mentioned, an Ionogram is a spectrogram that shows any deviation from the expected Ionosonde slope. Such deviations are caused by changes in signal travel time (reflection height). The resulting plot should have frequency on the x-axis and relative travel time on the y-axis. Note that the y-axis is *not* absolute travel time: to calculate this we would need to know exactly when the transmission started.

After capturing and generating a series of Ionograms, it is also desirable to create an animation to observe atmospheric changes over time. A separate 'Timelapser' tool was developed to generate these animations. The internal flowchart for the visualization block can be seen in figure 5.

5.3.2 Implementation

Since the previous LocalChirp block did all of the data processing, the remaining visualization is relatively simple. This block is known as "Ionogrammer". We load the ".chirp" file from the previous step into memory and use the *spectrogram()* tool from the Python matplotlib package. Important parameters (sample rate, center frequency, Ionosonde slope) are passed in from the previous block.

This brings us most of the way to a complete Ionogram, only requiring us to properly scale and label the axes.

Since the normalization performed in LocalChirp creates a *frequency* difference term, the true value of our y-axis is actually Hz. However, since we know the slope of the signal used to normalize this term, we can extract the more meaningful unit of time from this axis. For example, if we normalized a signal with a 100kHz/s tone, then we know that every 100kHz of deviation in our frequency is equal to 1 second of time deviation. Mathematically we scale the y-axis as follows:

$$\frac{F[Hz]}{P[Hz/s]} = t[s]$$

For the same reason, the x-axis is originally given in terms of time. Since our LocalChirp oscillator sweeps through the frequency spectrum at a known slope, we can convert this time axis into a more meaningful unit of frequency. This time we multiply the axis values by our slope, as follows:

$$t[s] * P[Hz/s] = F[Hz]$$

We use the passed-in information on center frequency and sample rate to properly scale these values.

The final step is to create animated timelapses of this data. To do so, we create a simple Python script, 'Timelapser'. Timelapser takes one argument (the path to the folder containing the PNGs) and automatically generates an animation of all images in the folder. This behavior is achieved through a combination of PIL (Python Image Library) and FFmpeg. Frames in the animation are sorted by file name. Since the project formats the filenames as date and time, the images are automatically in order. **Note:** Timelapser is the only script in the ChirpRX suite that is *not* automatically called. Timelapser must be manually called when the desired amount of timelapse data has been collected.

5.3.3 Output

The properly labeled Ionograms generated in Ionogrammer are saved as ".png" files with the same filename as the original ".chirp" input. The title of the Ionogram also mirrors this. **The original .RAW file for this image is deleted after processing to conserve storage space**.

The Timelapser tool outputs a .gif and .mov file into the directory it was run from.

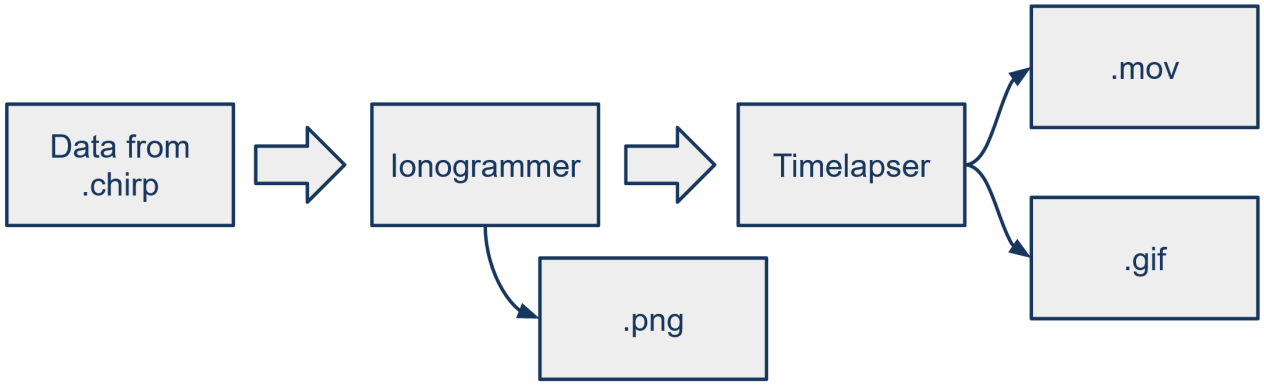

Figure 5: Visualization Flowchart

6 User Guide

This section will provide all of the knowledge necessary to operate the ChirpRX software suite. We will be using Raytheon's ROTHR Ionsondes [1] in this guide, but the same principles will apply to any linear Ionosonde. This section will also provide guidance on troubleshooting potential issues with the ChirpRX suite. **Before beginning this guide** please ensure that you have carefully read and followed the instructions to install dependencies as described in section 4.

6.1 What is ChirpRX?

ChirpRX is the collection of GNU-Radio/Python scripts we have developed to automatically capture and process Ionosonde data into Ionograms. Once tuned into a repeatable and consistent Ionosonde, ChirpRX can theoretically run indefinitely without further intervention[2].

ChirpRX in its current form does *not* automatically identify Ionosondes to record. The user must provide relevant Ionosonde characteristics to 'tune' ChirpRX to capture and process the desired Ionosonde. This process will be detailed in section 6.4.

6.2 USRP Connectivity

The following section details the setup process of the Ettus Research USRP X300/X310 SDR. Please be sure that all dependencies listed in the Software Configuration section have been installed before beginning this process. This guide is written explicitly for Ubuntu 22.04.

The USRP X300/X310 can communicate with the host PC in two ways: PCI-Express or 1Gig/10Gig networking. We will cover the steps to use 10 Gigabit networking. NOTE: This requires that the USRP device has the "HG" FPGA image loaded. If your device is not running the HG (default) image, please follow the guide here.

[1] see B.1 for more information on ROTHR
[2] Assuming infinite disk space

Begin by identifying the network interface on your host computer that is desirable to use to talk to the SDR. To view available network interfaces, use the following command

```
$ ip addr show
```

Figure 6 shows the results on the development system. There are 4 network interfaces: 'lo', 'enp0s25', 'enp3s0f0', and 'enp3s0f1'. 'lo' is an internal loopback, 'enp0s25' is the current internet connection, and the 'enp3s0f*' interfaces are the two ports of the Intel X520-DA2 NIC present on our machine. We will connect our USRP to enp3s0f1.

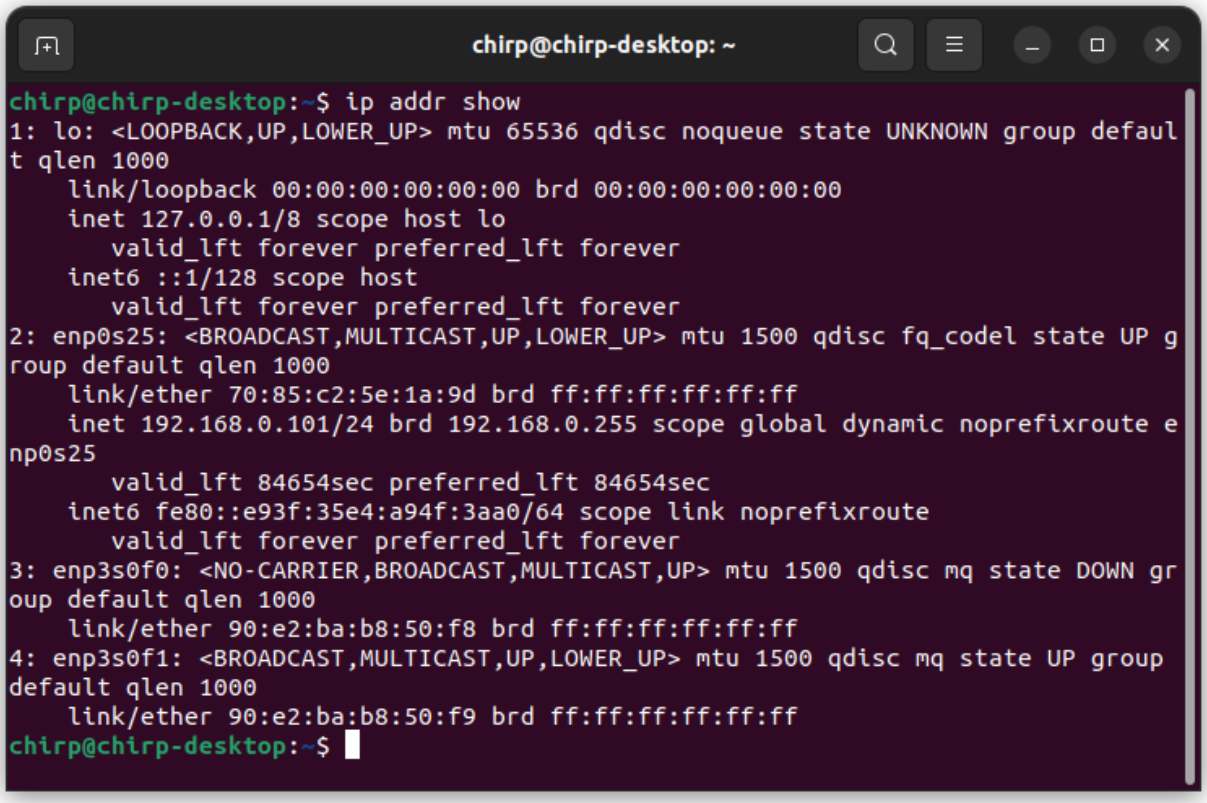

Figure 6: Results of 'ip addr show'

When using the "HG" FPGA image, the host computer should have the address 192.168.40.1 with a subnet mask of 255.255.255.0. Run the following command to set up the host with these parameters.

```
$ sudo ip addr add 192.168.40.1/24 dev <interface name>
```

You can now check to see if the USRP is connected by pinging the USRP, whose default address is 192.168.40.2. Be sure that the host is connected to port 1 of the USRP. Port 0 corresponds to 1 gigabit connection and has a different IP setup. Run the following command to ping the USRP.

```
$ ping 192.168.40.2
```

If the ping is returned, the setup was successful. To confirm that your USRP is being detected correctly or to see more details, run the following command.

```
$ uhd_usrp_probe
```

6.3 Setting up and using gqrx

The first step to using ChirpRX is to identify and characterize the Ionosonde you wish to capture. It is recommended to use gqrx (not part of the ChirpRX suite) as a tool to browse the RF spectrum. When you open gqrx you will be prompted to select your device and certain sampling parameters. Find your X300/X310 SDR in the "device" drop down menu and select it. If you do not see your device in this menu, please reference section 6.2 to set up your SDR.

Leave the 'device string' box as its default value. Next, choose a sample rate in the 'Input Rate' box. It is recommended to choose a sample rate high enough that you can view most/all of the Ionosondes spectrum, with the note that the LWA HF antenna has very poor reception below 8MHz. We have found that a sample rate of 12.5MSps works well for capturing ROTHR Ionosondes. Leave all other boxes as their default value and press "OK".

An important note on sampling rate: your sample rate *must* be an integer decimation of 200MHz (the base clock of the X3*0 SDRs). Gqrx will accept any value, but the SDR will round to the nearest decimation. It is highly recommended to precalculate and use an exact decimation, as this value will be critical later on.

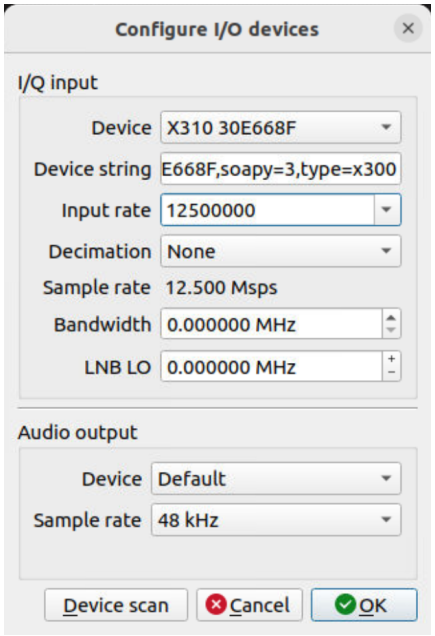

Figure 7: Example of I/O configuration for gqrx.

Gqrx operation is relatively straightforward. Tune your center frequency using the large number style dials in the center of the screen (red box in figure 8). It is recommended to change the waterfall

time span to 2+ minutes, which can be found in the "FFT Settings" tab on the right (green box in figure 8. Most other settings can be left as default, but additional gqrx operating information can be found on their website. Pressing the play button in the top left corner will begin DSP and start forming the waterfall view.

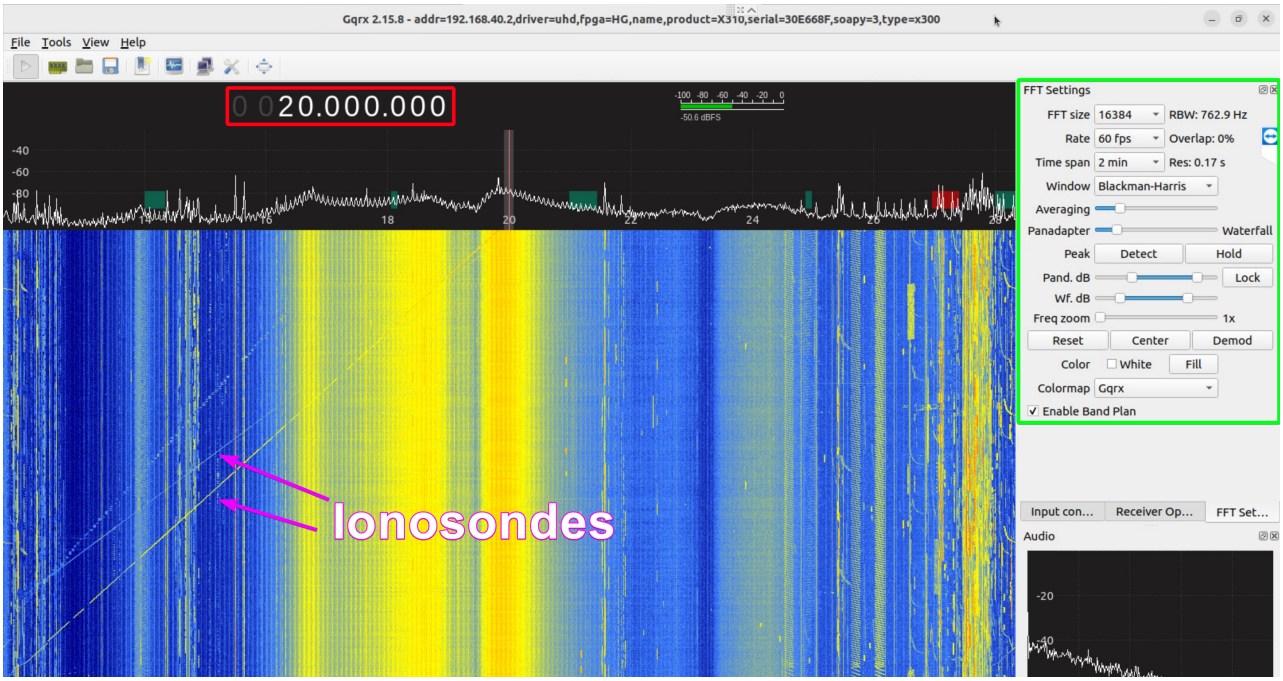

Figure 8: Gqrx GUI showing tuner, FFT settings, and ROTHR Ionosondes.

6.4 Identifying an Ionosonde

An Ionosonde can be identified in the gqrx waterfall by its distinctive diagonal shape, an example of which is seen in figure 8. It is convenient to pause the gqrx waterfall after capturing the Ionosonde so that you can gather its characteristics. You can see the frequency and time of any point in the gqrx waterfall by hovering your mouse over the point. The important characteristics of an Ionosonde (for ChirpRX) and how to find them are as follows:

- Slope - This can be found by choosing two points on the Ionosonde and applying the equation $\frac{F_2-F_1}{T_2-T_1}$. This returns slope in Hz/s, which is the unit ChirpRX accepts. For ROTHR Ionosonde A (see table 3) this is 100kHz/s.

- Period - This is the time between transmissions of the same Ionosonde. This requires multiple observations to calculate. Choose any frequency that the Ionosonde is known to sweep and simply find how long it takes for the Ionosonde to reach that frequency again. For ROTHR this value is 12 minutes.

- Frequency Range - This is the range of frequencies that the Ionosonde sweeps. It can be difficult to determine the true start and stop frequencies of an unknown Ionosonde due to non-

ideal reflection and reception. ChirpRX only requires you to specify a center frequency and bandwidth, **which does not have to perfectly match the Ionosonde**. For ROTHR we recommend using a center frequency of 14.25MHz with 12.5MHz of bandwidth, equivalent to recording from 8-20.5MHz.

- Start Time - This start time is **the time the Ionosonde enters the recorded spectrum**. If you are recording from 8MHz-20.5MHz, your start time should be the time you expect the Ionosonde to reach 8MHz, even if this is not the 'true' transmission start time of the Ionosonde.
  - ChirpRX accepts any past or future start time. If given a start time in the past, it will automatically calculate the next transmission time based on the given period. ChirpRX only needs an initial start time to begin automatic capture.

6.5 Using ChirpRX

Now that an Ionsonde has been identified, we can 'tune' ChirpRX to capture it. ChirpRX can be run from the command line with one command where the Ionosonde characteristics are passed in as arguments. A full list of arguments and their default values can be found in table 1. To capture the ROTHR Ionosonde we identified in the last section, we would run the following command:

```
$ python3 ChirpRX.py --H 14 --M 32 --S 3
```

Note that, since ROTHR is the default system, we were able to leave out many arguments. Please be sure to use the below table if using any non-ROTHR Ionosonde.

Variable	Arg Shorthand	Default Value	Unit	Description
Hour	-H	0	N/A	Start Hour
Minute	-M	0	N/A	Start Minute
Second	-S	0	N/A	Start Second
Period	-T	12	Minutes	Ionosonde Period
Slope	-M	100k	Hz/s	Ionosonde Slope
Offset	-O	0	ms	Millisecond recording offset. Accepts negative values.
F_c	-C	14.25M	Hz	Spectrum Center Frequency
F_s	-B	12.5M	Hz	Sample Rate (Bandwidth)
Decimation	-D	5000	N/A	Post-filtering decimation. Large values give a more zoomed in Ionogram.

Table 1: ChirpRX Arguments

It can be helpful to begin with smaller decimation values (say 100) to ensure that your signal is present on the Ionogram before 'zooming in' with higher decimation. It may also be necessary to adjust your millisecond offset to more carefully align the Ionogram as you increase your zoom.

6.5.1 Creating Timelapses

After ChirpRX has run for some time it may be desirable to create an animation from the generated images. A tool has been developed to create these animations, "timelapser.py". Timelapser will generate an animation in both .gif and .mov formats. Timelapser requires one argument to be passed, the path to the folder containing the pngs. An example of how to run timelapser is the following command:

```
$ python3 timelapser.py -P './path/to/pngfolder'
```

The resulting .gif and .mov file will be created in the directory where timelapser.py was run from.

6.6 Troubleshooting and Known Issues

This section will explain some of the common errors encountered in the ChirpRX software suite and provide guidance when possible.

6.6.1 No devices found for Empty Device Address

This error occurs when the computer can not locate the SDR on initialization. This occurs most commonly when the SDR is being utilized by another program. Ensure that all other programs using the SDR (gqrx, other instances of ChirpRX) are closed and try again. If this error persists, ensure that your SDR is properly set up and connected to the host machine.

6.6.2 [WARNING] [RFNOC::GRAPH] One or more blocks timed out during flush!

This error message comes from the SDR when initialization fails. This most commonly occurs when the previous SDR initialization was closed abruptly. This error is usually fatal, but can easily be remedied by simply re-launching the program and therefore re-initializing the SDR.

6.6.3 'O' in console output

The USRP SDR sends an 'O' to /dev/stdout when the host machine failed to consume some data generated by the SDR. Occasional occurrences of 'O' are expected and are generally harmless. If many 'O's occur in quick succession ensure that the host machine is not being slowed down by another program.

6.6.4 usrp_source :error: In the last x ms, y overflows occurred.

This warning provides a summary of overflows in the last x ms. As mentioned in the previous subsection, occasional overflows are expected and harmless. If a large number of overflows are occurring then ensure that the host machine is not being slowed down by another program.

6.6.5 buffer_double_mapped :warning:

This warning occurs at high decimation levels (5000+). This warning occurs when the automatic low-pass filter generation in the LocalChirp step attempts to make a very narrow filter. This warning can typically be safely ignored, as the exact shape of the filter is not critical to the DSP.

6.6.6 Why is the high end of my spectrum cutoff?

At high decimation levels, LocalChirp sometimes does not process the full spectrum before calling Ionogrammer. The root cause of this is currently unknown. This behavior fluctuates as decimation increases.

If this behavior is unacceptable for your application, consider recording a wider band than is necessary to help mitigate the issue.

7 Results

7.1 Catalog and Searching

The first objective to complete was to find and catalog any active Ionosondes that the receiver station can detect. The documented frequencies that an Ionosonde signal sweeps through are roughly 0 - 30 MHz. However, this station had poor performance below 8MHz, limiting our observable range.

Only one reliable set of transmitters was observed, which were identified to be from Raytheon's ROTHR program (see appendix B.1 for more information on this program). This set consisted of two unique Ionosondes that always transmit together, though only one was usable for our application. The usable Ionosonde sweeps linearly from 2-20MHz at 100kHz/s. The other Ionosonde in this system sweeps a wider range (5-28MHz) and is generally higher power, but is unusable due to its non-linear sweeping pattern. Instead of sweeping linearly, this Ionosonde consists of multiple high-slope 'subpulses' with short delays between each subpulse. Generating an Ionogram from this kind of transmission requires more detailed transmission information.

A full catalog of Ionosonde transmissions observed during this project can be found in appendix B.2.

7.2 Ionograms

Following the discovery of active Ionosondes, the data collected was used to create Ionograms. Ionograms detail changes in the Ionosphere by measuring the time it takes for an Ionosonde transmission to leave the transmitter, bounce off the ionosphere, and arrive at the receiver. This change in time and elements of the reflected signals detail the ionospheric layer changes, which can be seen in an Ionogram. Below is an Ionogram generated by our receiver station with the layers identified.

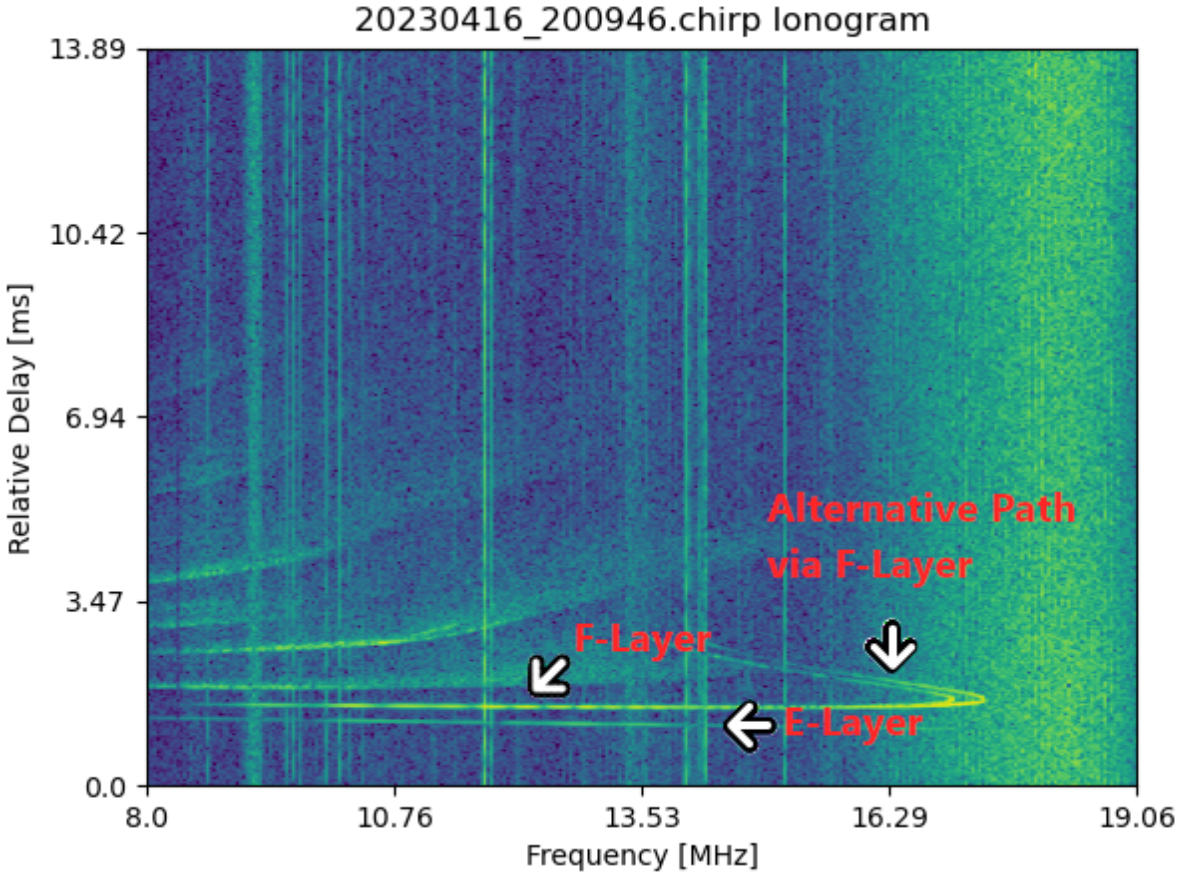

Figure 9: Ionogram

This figure shows the height at which these Ionospheric layers in terms of time reside during the recorded time. This will be explained more in the next section. The x-axis and y-axis represent frequency (in MHz) and time (in the form of relative delay, ms) respectively.

7.3 Ionosphere Layers

The E-layer is the lowest reflective layer of earth's ionosphere and is represented in the previous figure as the lowest straight line located under the F-Layer. The F-layer is usually distinguishable by the "tail-like" end that curves upward. The tail represents an alternative, longer path that the signal takes within the F-Layer. This observed behavior aligns with the existing body of research on Ionosondes and the ionosphere.

7.4 Ionogram Animation

The aggregate animation is a timelapse of multiple Ionograms over many hours. Thus, as previously mentioned, this animation will show the change in height of the Ionosphere and its respective layers.

A preview of this is shown below.

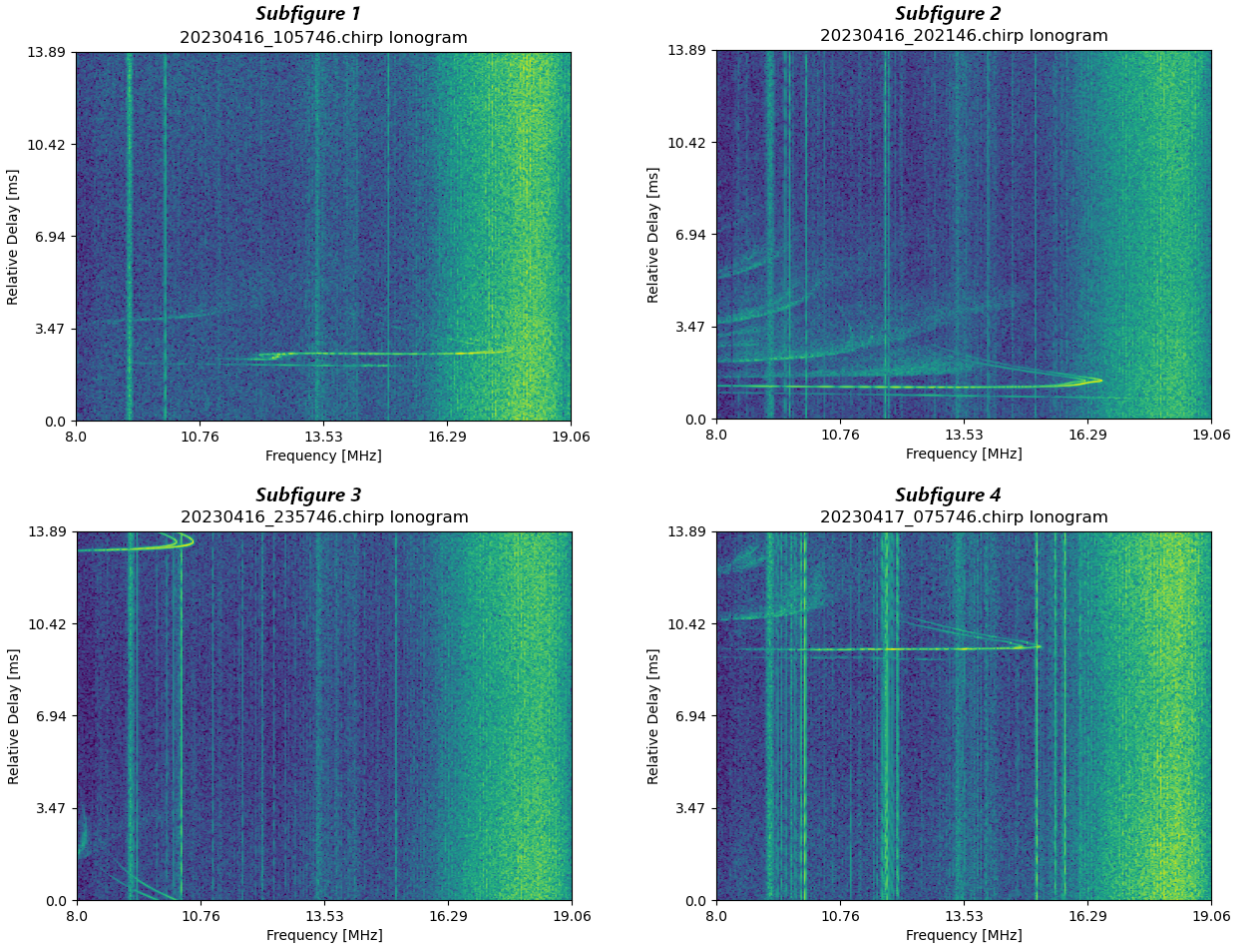

Figure 10: Ionogram Animation Example

The following figure contains multiple sub-figures that are part of an Ionogram timelapse. The first and fourth images are the beginning and end of that day's recording, respectively, while the second and third are closer to nighttime.

The most important aspect of the timelapse is the change of the MUF (Maximum Usable Frequency). The MUF increases during daylight hours and decreases at night. The change in MUF can be observed by noting the frequency at which the F layer tail begins to curve back onto itself, most notable in subfigures 1 and 3.

Another interesting atmospheric effect can be observed in Subfigure 2 and Subfigure 4: the 'wisps' above the aforementioned E and F layers. These wisps are caused by the signal reflecting off of the Earth's surface and back to the Ionosphere multiple times, causing an 'echo' effect on the Ionogram.

7.4.1 Timelapse 'Drift'

Timelapses generated by the current iteration of ChirpRX have a tendency to 'drift' downwards over time. This is the reason that the Ionosonde signal shown in figure 10 appears at several different heights in each subfigure[3].

This behavior is *not* a physical change in the Ionosphere, but is instead a problem with the way TimerTrigger clocks. TimerTrigger is currently configured to use the host machine's system clock, which is susceptible to small amounts of drift over time. As the timing requirements of this project are relatively strict, this causes a noticeable drift in our generated timelapses. The original goal of this project was to use a GPS-disciplined clock for the TimerTrigger, but this feature was not successfully implemented before project completion.

8 Budget

The project had a $500 budget provided by the Electrical Engineering (EE) department at New Mexico Tech. Most of the project's hardware components were available on loan from the EE department at no cost. The only exception is the last two items within the bill of materials (sets of BNC to SMA adapters).

All components used to recreate this project and their full costs can be found in the bill of materials (Appendix A).

9 Conclusion

The design team is happy to report that an automatic Ionosonde capture system has been built and is capable of receiving transmissions from oblique Ionosondes. Furthermore, the system is capable of using the recorded data to create Ionograms that show important atmospheric propagation conditions, including variations in E and F layer height over time, as well as changes in the Maximum Usable Frequency.

9.1 Summary of Project Goals

Most of the goals outlined in the project requirements were completed, however, three goals were not met.

First: due to the nature of Raytheon's ROTHR Ionosondes, the team was only able to create a 21-hour Ionogram animation, the reason for this is further described in Appendix B.1. Although many transmitters were observed, the ROTHR Ionosondes were the only transmitters with a reliable and discernible transmission schedule.

[3]Note that when the signal drifts off the bottom edge of the figure, it aliases back onto the top. This is why the bottom of the F layer seems to appear well above the tail in subfigure 3

The second missed goal was the disciplining of the TimerTrigger to a GPS clock. The current configuration relies only on the host machine's system clock, which generates the undesirable 'drifting' behavior described in section 7.4.1.

The final missing goal is the delivery of data in HDF5 format. It was incorrectly assumed while developing the project that packaging data into an HDF5 archive was of similar complexity to packaging as a *ZIP* or other compressed format. Unfortunately, this packaging step was left until the conclusion of the project, where it was discovered to be of significantly higher complexity. Data was instead delivered in standard *ZIP* archives, with the only metadata existing in the form of filenames as specified in section 5.1.3.

9.2 Future Work

Were this project to continue, it would be advantageous to create an auto-trigger script that would automatically detect Ionosondes and record their characteristics. Doing so would theoretically allow ChirpRX to record without operator intervention. Another possible update would be to evaluate an antenna that has better reception below 8MHz. ROTHR Ionosondes have a significant portion of their transmission in the 0-8MHz range, but the current station is unable to receive any useful data in this part of the spectrum. Updating the TimerTrigger block of ChirpRX to use a GPS-disciplined clock instead of the host system's clock would also be desirable. Doing so should fix the time drift observed in long-term (24hr+) recordings.

References

[1] Pieter-Tjerk de Boer. *Chirp reception and interpretation*. 2013. URL: http://websdr.ewi.utwente.nl:8901/chirps/article/.

[2] Raytheon Corporation. *Relocatable Over-the-Horizon Radar (ROTHR) for Homeland Security*. 2004. URL: http://www.mobileradar.org/Documents/ROTHR.pdf.

[3] Sarah Frazier and Lisa Tran. *10 Things to Know About the Ionosphere*. URL: https://solarsystem.nasa.gov/news/1127/10-things-to-know-about-the-ionosphere/.

[4] Whitham D. Reeve. *Modeling the Long Wavelength Array Crossed-Dipole Antenna*. 2014. URL: https://www.reeve.com/Documents/Long%5C%20Wavelength%5C%20Array/Reeve_LWA-Model.pdf.

Appendices

A Bill of Materials

Table 2: Bill of Materials

Model Number	Vendor Name	Distributor Model Number	Distributor Name	Part Description	Price per Unit	Qty	Total Cost
783144-01	Ettus Research	783144-01	Ettus Research	USRP X300 SDR	$7,597.00 USD	1	$7,597.00 USD
782750-01	Ettus Research	782750-01	Ettus Research	BasicRX Daughterboard 1-250 MHz Rx	$149.00 USD	1	$149.00 USD
TP3005T	Tekpower	B00ZBCLJSY	Amazon	Variable Linear DC Power Supply	$79.99 USD	1	$79.99 USD
LWA-SYS	Reeve	LWA-SYS	Reeve	LWA HF Antenna + Assembly	$1450.00 USD	1	$1450.00 USD + Shipping
TB-510+	Mini-Circuits	TB-510+	Mini-Circuits	Bias Tee	$75.84 USD	1	$75.84 USD + Shipping
B07YJHBVKW	RFAdapter	B07YJHBVKW	Amazon	Male to Male SMA Cables	$9.99 USD	1	$9.99 USD
785023-01	Ettus Research	785023-01	Ettus Research	USRP X300 SDR US Power Cord	$22.00 USD	1	$22.00 USD
LYSB01NCAL4DR-CMPTRACCS	Maxmoral	B01NCAL4DR	Amazon	N-Type Female to SMA Male Adapter	$7.49 USD	2	$14.98 USD
960	Adafruit	485-960	Mouser	GPS Antenna	$19.95 USD	1	$19.95 USD + Shipping
74752-1301	HPC Optics	B01N4VJP93	Amazon	SFP+ Male to SFP Male+	$54.99 USD	1	$54.99 USD
OL1183-2X	onelinkmore	B00VG5HWH2	Amazon	Two BNC Female to Banana Adapter	$6.78 USD	1	$6.78 USD
OL-2770-X	onelinkmore	B017SOLCMU	Amazon	Two SMA to BNC Adapter Kits	$6.59 USD	1	$6.59 USD

B Ionosonde Reception Catalog

B.1 Known Transmitters

Over the course of this project, only one group of transmitters was consistently identifiable. These oblique Ionosonde transmitters belong to the Relocatable Over The Horizon RADAR (ROTHR) program, operated by Raytheon for the US Navy [2]. Three sites are currently in operation, located in Virginia, Texas, and Puerto Rico, each with its own oblique Ionosonde.

While the design team considers the ROTHR Ionosondes as "consistently identifiable" it is important to note that the transmitters do cease transmitting for a period of time, roughly every 24 hours. When the transmitters begin transmitting again, the transmission period remains the same (12 minutes), but the start time appears to be random. Because of this, it is not possible to record more than about 21 hours of transmissions when using a time-based capture algorithm.

Table 3: ROTHR Ionosonde Specifications

	ROTHR A	ROTHR B
Transmission Period (Minutes)	12	12
Sweep Rate (kHz/sec)	100	∼81.37
Start Frequency (MHz)	2	5
Stop Frequency (MHz)	20	28
Continuous Sweep	Yes	No

B.2 Received Transmissions List

The following table is a manually recorded list of transmitters observed by the design team. It should be noted that when a start and stop frequency are listed in the table, the data were recorded by sight using the waterfall in GQRX. This start/stop frequency is not necessarily representative of the beginning/end of the transmission; instead, it usually represents the visual appearance and disappearance of the Ionosonde in the waterfall. Transmissions without a start and stop frequency listed were recorded aurally. Finally, the "Power" metric in the table represents the relative visibility of the Ionosonde in the waterfall. An entry of "INVIS" in the "Power" column represents an Ionosonde that was heard, but where the signal strength was too low to be identified visually in the waterfall.

Date (MT)	Time (MST/MDT) @ 12.5 MHz	Date (UTC)	UTC Time @ 12.5 MHz	Power	Start Frequency (MHz)	End Frequency (MHz)	Slope (kHz/sec)
1/11/23	22:24:57	1/12/23	5:24:57	LOW	8	14.3	78.75
1/11/23	22:28:58	1/12/23	5:28:58	MED	7.7	12.4	82.46
1/11/23	22:31:23	1/12/23	5:31:23	MED	5	8.1	103.33
1/11/23	22:36:56	1/12/23	5:36:56	LOW	7.4	12.8	84.38
1/11/23	22:40:55	1/12/23	5:40:55	LOW	9.3	11.7	80.00
1/11/23	22:43:32	1/12/23	5:43:32	LOW	5.7	7.5	81.82
1/11/23	22:49:08	1/12/23	5:49:08	LOW	7.7	12.3	64.79
1/11/23	22:52:46	1/12/23	5:52:46	LOW	8.2	12.8	102.22
1/11/23	23:00:54	1/12/23	6:00:54	LOW	7.6	12.5	85.96
1/12/23	13:29:02	1/12/23	20:29:02	LOW	12	22	87.72
1/12/23	13:29:14	1/12/23	20:29:14	LOW	12	20	106.67
1/12/23	13:41:04	1/12/23	20:41:04	LOW	14.54	24	87.59
1/12/23	13:41:15	1/12/23	20:41:15	LOW	14.527	20	107.31
1/12/23	13:53:12	1/12/23	20:53:12	MED	12.131	20	102.19
1/12/23	13:53:00	1/12/23	20:53:00	MED	12.2	27.98	83.94
1/12/23	13:55:33	1/12/23	20:55:33	HIGH	7.73	18.06	82.64
1/12/23	14:05:12	1/12/23	21:05:12	MED	12.27	20	101.71
1/12/23	14:05:00	1/12/23	21:05:00	MED	12.27	27.98	83.56
1/12/23	14:07:34	1/12/23	21:07:34	HIGH	8	17.8	83.76
1/15/23	18:07:34	1/16/23	1:07:34	MED	6.38	16.6	82.42
1/15/23	18:19:34	1/16/23	1:19:34	MED	6.38	16.4	82.13
1/15/23	18:31:31	1/16/23	1:31:31	MED	6.37	16.37	79.37
1/15/23	18:43:58	1/16/23	1:43:58	MED	5.7	16.3	90.60
1/15/23	18:55:30	1/16/23	1:55:30	MED	5.8	26.3	81.35
1/15/23	18:53:04	1/16/23	1:53:04	LOW	7.9	24.28	85.31
1/15/23	19:05:03	1/16/23	2:05:03	LOW	7.7	23.5	80.20
1/15/23	19:10:59	1/16/23	2:10:59	LOW	7.8	24.6	81.95
1/15/23	19:17:03	1/16/23	2:17:03	LOW	7.85	22.5	77.93
1/15/23	19:23:11	1/16/23	2:23:11	LOW	7.8	24.6	88.42
1/19/23	16:17:12	1/19/23	23:17:12	LOW	10.667	27.98	81.28
1/20/23	12:12:35	1/20/23	19:12:35	HIGH	8.19	27.97	83.81
1/20/23	12:12:48	1/20/23	19:12:48	MED	12.63	19.99	102.22
1/20/23	12:24:34	1/20/23	19:24:34	HIGH	7.96	27.96	82.99
1/20/23	12:24:47	1/20/23	19:24:47	MED	11.56	19.99	102.80
1/20/23	12:36:34	1/20/23	19:36:34	HIGH	8.13	27.98	83.05
1/20/23	12:36:47	1/20/23	19:36:47	LOW	8.83	18.9	101.72
3/23/23	9:11:33	3/23/23	15:11:33	INVIS			
3/23/23	9:15:16	3/23/23	15:15:16	LOW			
3/23/23	9:15:57	3/23/23	15:15:57	HIGH			
3/23/23	9:16:10	3/23/23	15:16:10	HIGH			
3/23/23	9:27:16	3/23/23	15:27:16	INVIS			
3/23/23	9:27:28	3/23/23	15:27:28	INVIS			

3/23/23	9:27:58	3/23/23	15:27:58	HIGH			
3/23/23	9:28:10	3/23/23	15:28:10	HIGH			
3/23/23	9:28:58	3/23/23	15:28:58	MED			
3/23/23	9:35:27	3/23/23	15:35:27	LOW			
3/23/23	9:39:16	3/23/23	15:39:16	LOW			
3/23/23	9:39:28	3/23/23	15:39:28	LOW			
3/23/23	9:39:57	3/23/23	15:39:57	MED			
3/23/23	9:40:11	3/23/23	15:40:11	MED			
3/23/23	9:41:36	3/23/23	15:41:36	INVIS			
3/23/23	9:44:28	3/23/23	15:44:28	LOW			
3/23/23	9:49:58	3/23/23	15:49:58	LOW			
3/23/23	9:51:16	3/23/23	15:51:16	LOW			
3/23/23	9:51:28	3/23/23	15:51:28	LOW			
3/23/23	9:51:57	3/23/23	15:51:57	MED			
3/23/23	9:52:10	3/23/23	15:52:10	MED			
3/27/23	8:49:00	3/27/23	14:49:00	MED			
3/27/23	8:50:27	3/27/23	14:50:27	LOW			
3/27/23	8:51:23	3/27/23	14:51:23	LOW			
3/27/23	8:51:35	3/27/23	14:51:35	LOW			
3/27/23	8:52:02	3/27/23	14:52:02	MED			
3/27/23	8:52:15	3/27/23	14:52:15	MED			
3/27/23	8:52:58	3/27/23	14:52:58	MED			
3/27/23	8:55:00	3/27/23	14:55:00	MED			
3/27/23	8:55:55	3/27/23	14:55:55	LOW			
3/27/23	8:58:59	3/27/23	14:58:59	MED			
3/27/23	9:00:59	3/27/23	15:00:59	LOW			
3/27/23	9:01:26	3/27/23	15:01:26	INIVS			
3/27/23	9:03:23	3/27/23	15:03:23	INVIS			
3/27/23	9:03:35	3/27/23	15:03:35	INVIS			
3/27/23	9:04:02	3/27/23	15:04:02	MED			
3/27/23	9:04:14	3/27/23	15:04:14	MED			
3/27/23	9:04:59	3/27/23	15:04:59	MED			
3/27/23	9:06:56	3/27/23	15:06:56	LOW			
3/27/23	9:06:59	3/27/23	15:06:59	INVIS			
3/27/23	9:10:58	3/27/23	15:10:58	MED			
3/27/23	9:12:26	3/27/23	15:12:26	INVIS			
3/27/23	9:13:00	3/27/23	15:13:00	LOW			
3/27/23	9:15:23	3/27/23	15:15:23	INVIS			
3/27/23	9:15:35	3/27/23	15:15:35	INVIS			
3/27/23	9:16:02	3/27/23	15:16:02	HIGH			
3/27/23	9:16:14	3/27/23	15:16:14	MED			
3/27/23	9:17:59	3/27/23	15:17:59	LOW			
3/27/23	9:17:56	3/27/23	15:17:56	INVIS			
3/27/23	9:19:00	3/27/23	15:19:00	LOW			
3/27/23	9:22:59	3/27/23	15:22:59	MED			
3/27/23	9:23:26	3/27/23	15:23:26	INVIS			

3/27/23	9:25:00	3/27/23	15:25:00	LOW			
3/28/23	9:07:28	3/28/23	15:07:28	HIGH			
3/28/23	9:13:59	3/28/23	15:13:59	HIGH			
3/28/23	9:16:05	3/28/23	15:16:05	MED			
3/28/23	9:16:17	3/28/23	15:16:17	MED			
3/28/23	9:18:29	3/28/23	15:18:29	MED			
3/28/23	9:24:58	3/28/23	15:24:58	HIGH			
3/28/23	9:28:04	3/28/23	15:28:04	MED			
3/28/23	9:28:16	3/28/23	15:28:16	LOW			
3/28/23	9:29:29	3/28/23	15:29:29	MED			
3/28/23	9:34:59	3/28/23	15:34:59	MED			
3/28/23	9:40:05	3/28/23	15:40:05	MED			
3/28/23	9:40:16	3/28/23	15:40:16	LOW			
3/28/23	9:40:29	3/28/23	15:40:29	MED			
3/30/23	11:40:11	3/30/23	17:40:11	LOW			
3/30/23	11:40:23	3/30/23	17:40:23	LOW			
3/30/23	11:52:11	3/30/23	17:52:11	LOW			
3/30/23	11:52:23	3/30/23	17:52:23	LOW			
3/30/23	12:00:59	3/30/23	18:00:59	INVIS			
3/30/23	12:04:10	3/30/23	18:04:10	LOW			
3/30/23	12:04:22	3/30/23	18:04:22	LOW			
3/30/23	12:04:11	3/30/23	18:04:11	LOW			
3/30/23	12:04:23	3/30/23	18:04:23	LOW			
3/30/23	12:16:11	3/30/23	18:16:11	LOW			
3/30/23	12:16:23	3/30/23	18:16:23	LOW			
3/31/23	18:22:01	4/1/23	0:22:01	LOW			
3/31/23	18:23:08	4/1/23	0:23:08	LOW			
3/31/23	18:23:19	4/1/23	0:23:19	MED			
3/31/23	18:23:32	4/1/23	0:23:32	LOW			
3/31/23	18:23:49	4/1/23	0:23:49	INVIS			
3/31/23	18:24:05	4/1/23	0:24:05	INVIS			
3/31/23	18:28:33	4/1/23	0:28:33	LOW			
3/31/23	18:35:03	4/1/23	0:35:03	LOW			
3/31/23	18:35:10	4/1/23	0:35:10	LOW			
3/31/23	18:35:21	4/1/23	0:35:21	MED			
3/31/23	18:35:23	4/1/23	0:35:23	LOW			
3/31/23	18:35:34	4/1/23	0:35:34	LOW			
3/31/23	18:35:51	4/1/23	0:35:51	LOW			
3/31/23	18:36:05	4/1/23	0:36:05	LOW			
3/31/23	18:41:32	4/1/23	0:41:32	LOW			
3/31/23	18:47:10	4/1/23	0:47:10	LOW			
3/31/23	18:47:21	4/1/23	0:47:21	MED			
3/31/23	18:47:23	4/1/23	0:47:23	LOW			
3/31/23	18:47:33	4/1/23	0:47:33	LOW			
3/31/23	18:47:51	4/1/23	0:47:51	INVIS			
3/31/23	18:48:02	4/1/23	0:48:02	LOW			

3/31/23	18:54:30	4/1/23	0:54:30	LOW			
3/31/23	18:59:10	4/1/23	0:59:10	LOW			
3/31/23	18:59:21	4/1/23	0:59:21	MED			
3/31/23	18:59:22	4/1/23	0:59:22	LOW			
3/31/23	18:59:33	4/1/23	0:59:33	LOW			
3/31/23	18:59:51	4/1/23	0:59:51	LOW			
3/31/23	19:00:04	4/1/23	1:00:04	LOW			
3/31/23	19:01:00	4/1/23	1:01:00	LOW			
4/4/23	13:32:21	4/4/23	19:32:21	INVIS			
4/4/23	13:35:18	4/4/23	19:35:18	MED			
4/4/23	13:35:31	4/4/23	19:35:31	MED			
4/4/23	13:44:21	4/4/23	19:44:21	INVIS			
4/4/23	13:47:20	4/4/23	19:47:20	MED			
4/4/23	13:47:32	4/4/23	19:47:32	LOW			
4/4/23	13:56:20	4/4/23	19:56:20	INVIS			
4/4/23	13:59:20	4/4/23	19:59:20	MED			
4/4/23	13:59:33	4/4/23	19:59:33	LOW			