GNU Radio Interface

Team GNU - It's Not Radio

Team Members:

Walker Frank

Quyen Nguyen

Max Oakes

Nick Wagner

Quentin Ward

Faculty Advisors:

Joseph Hoffbeck

James Schmidt

Industry Advisor:

Kyle Bernard

Table of Contents

1.0 Executive Summary 6
1.1 Rationale 6
1.2 Project Description 6
1.3 Technical Issues 6
1.3.1 Hardware Requirements 6
1.3.2 Software Requirements 6
1.4 Results of Project 7
1.5 Conclusion and Discussion 7
2.0 Introduction 7
3.0 Background 7
3.1 Goal 7
3.2 High Level Description 7
3.3 Current Condition 8
3.4 Problem and Solution 8
4.0 Discussion 8
4.1 Design Consideration Table 8
4.2 Design Criteria Table 9
4.3 Alternative Designs 10
4.4 Code Design 11
4.5 Testing Procedures 12
4.6 Engineering Standards 12
4.7 Professional Responsibility 12
5.0 Finances 13
6.0 Conclusions and Recommendations 16
7.0 References 17
8.0 Acknowledgments 17
9.0 Appendices 17
9.1 Weekly Status Report 17
9.1.1 Week 9/23/2016 17
9.1.2 Week 9/30/2016 18
9.1.3 Week 10/07/2016 18

9.1.4 Week 10/28/2016 19
9.1.5 Week 11/04/2016 19
9.1.6 Week 11/11/2016 20
9.1.7 Week 11/18/2016 20
9.1.8 Week 12/2/2016 21
9.1.9 Week 1/27/2017 21
9.1.10 Week 2/4/2017 21
9.1.11 Week 2/10/2017 22
9.1.12 Week 2/17/2017 22
9.1.13 Week 2/24/2017 22
9.1.14 Week 3/10/2017 23
9.1.15 Week 3/24/2017 23
9.1.16 Week 3/31/2017 23
9.1.17 Week 4/7/2017 24
9.2 Gantt Chart 24
9.3 RSA306 26
9.4 Tektronix Deliverables 26
9.5 Total Team Hours Spent 27

Tables and Figures

Table 1. Design Consideration Table 9
Table 2. Design Criteria Table 9
Table 3. Estimated Equipment Cost 14
Table 4. Rent Budget 14
Table 5. Developer Budget 14
Table 6. Consultant Budget 14
Table 7. Overall Development Cost 14
Table 8. Cost of unit per software package 14
Table 9. Total Expenses 15
Table 10. Total Revenue 15
Table 11. Income Statement 16
Table 12. Balance Sheet 16

Figure 1. Process Overview 8
Figure 2. Block Diagram of Current Design 11

Figure 3. Gantt Chart 25
Figure 4. RSA306 Diagram 26

1.0 Executive Summary

1.1 Rationale

Research in wireless communications is currently undergoing a revolution due to the increase in wireless communication devices (e.g. cell phones, drones, Wi-Fi) in everyday applications. The demand for cheaper and more powerful research devices is ever increasing as a result. The ability to connect to GNU Radio satisfies this demand by increasing the power and research potential of a cheaper research tool.

1.2 Project Description

The purpose of this project is to design and create a working interface between the Tektronix RSA and GNU Radio. This includes a method of reading blocks and streams of IQ and IF data types in such a way that they can be interpreted by the open source community GNU Radio, where other users that also possess an RSA can use the device easily.

The project is written mainly in C++ to interface with the RSA, and in Python to interface with GNU Radio. The program will have three main parts: IQ stream, IF stream and IQ block. For each of these parts, there are several different data types the output could be, including separate arrays for I and Q streams, for example. Once these parts are created, they can be 'plugged' into GNU Radio and then data from the RSA can be presented or modified in hundreds of ways.

Tektronix's RSA306 is a recent technology that has a lot of potential, but can be exclusively accessed by Tektronix software or smaller, more specific programs, limiting that it is able to do. However, with a bridge between the RSA and GNU Radio, major steps can be taken toward unlocking the devices potential since that community already has hundreds of tools to use the data that can be obtained from the RSA.

1.3 Technical Issues

1.3.1 Hardware Requirements

A full collection made by the RSA306A will produce approximately 200 MB per second. In order for real-time collection to be feasible a computer that can write at that speed. A Solid-State Drive (SSD) would provide us enough read/write ability for our project. The computer system provided in the Computer Science Laboratory (Shiley 304) is outfitted with an SSD and is capable of handling real-time operations. A RSA306A and RSA503A will be required to test if our code is compatible with the required devices.

1.3.2 Software Requirements

In order to develop C++ code, Mars Eclipse.2 4.5.2 with C/C++ IDE and the RSA API is required. The coding environment will be the Linux Ubuntu 15 Operating System (OS). To test data collected from the device to ensure accuracy, Tektronix SignalVu Software, and MATLAB

will be used. The computer system provided in the Computer Science Laboratory, as well as, personal coding development computers, have the required software installed.

1.4 Results of Project

While the project was successfully completed, the RSA still does not have a workable transmitter. This limits some possible applications of the RSA since it cannot act as a full transceiver. Tektronix, or a future senior design group at the University of Portland, will be left the responsibility of completing this task, as the scope of this project doesn't include a working transmitter.

1.5 Conclusion and Discussion

The creation of the GNU Radio blocks will bring a wide range of abilities for users of an RSA device. GNU Radio is a software defined radio and our block with an RSA will serve as the antenna for this radio. Since the RSA can read a wide spectrum, our block provides a powerful tool for GNU Radio.

2.0 Introduction

The Tektronix RSA306 USB Spectrum Analyzer is a full-featured spectrum analyzer for less than half the price of a conventional unit. It can be plugged into a computer using the USB port and, using Tektronix SignalVu software, can display capture signals in the frequency range of 9 kHz to 6.2 GHz. However, this device cannot connect to GNU Radio. GNU Radio is a powerful, open-source, coding depositary that is used by the research industry to conduct low cost signal analysis. The RSA306 Spectrum Analyzer would benefit greatly from this addition because it increases the applications of the RSA306 therefore opening new markets for Tektronix. This senior design project aims to create a C++ coding block for GNU Radio so the RSA306 Spectrum Analyzer can connect to and stream data to GNU Radio.

3.0 Background

3.1 Goal

The primary goal of this project is to increase the processing capability of the RSA306 Spectrum Analyzer by enabling the RSA306 to connect to the open-source coding platform GNU Radio. This will be done by creating a source block of C++ code that will be made available on the GNU Radio development environment. This code will be offered without cost to the user.

3.2 High Level Description

GNU Radio is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors. It is widely used in academic and commercial

environments to support wireless communications research as well as to implement real-world radio systems. The RSA306 [9,3] is a portable, 40 MHz Real-Time Spectrum Analyzer that contains an acquisition system inside a small module. The user interface and display resides on a user-provided PC running SignalVu-PC software. The host PC provides all power, control, and data signals over the USB 3.0 cable included with the instrument.

3.3 Current Condition

The RSA306 currently functions as a portable spectrum analyzer. It can receive and process wireless signals through SignalVu software, which only runs on Windows system. This is a limitation for Linux users to access data from the RSA306. Therefore, connecting RSA306 to GNU Radio will allow the RSA to fully function on both Windows and Linux environment, as well as give users the ability to access data off the RSA into GNU Radio.

3.4 Problem and Solution

There is no driver that allows the RSA306 to connect to GNU Radio. The solution to this is to design a driver for it. The overall process is shown in the following diagram. The dotted red area is what is missing from being able to work with GNU Radio.

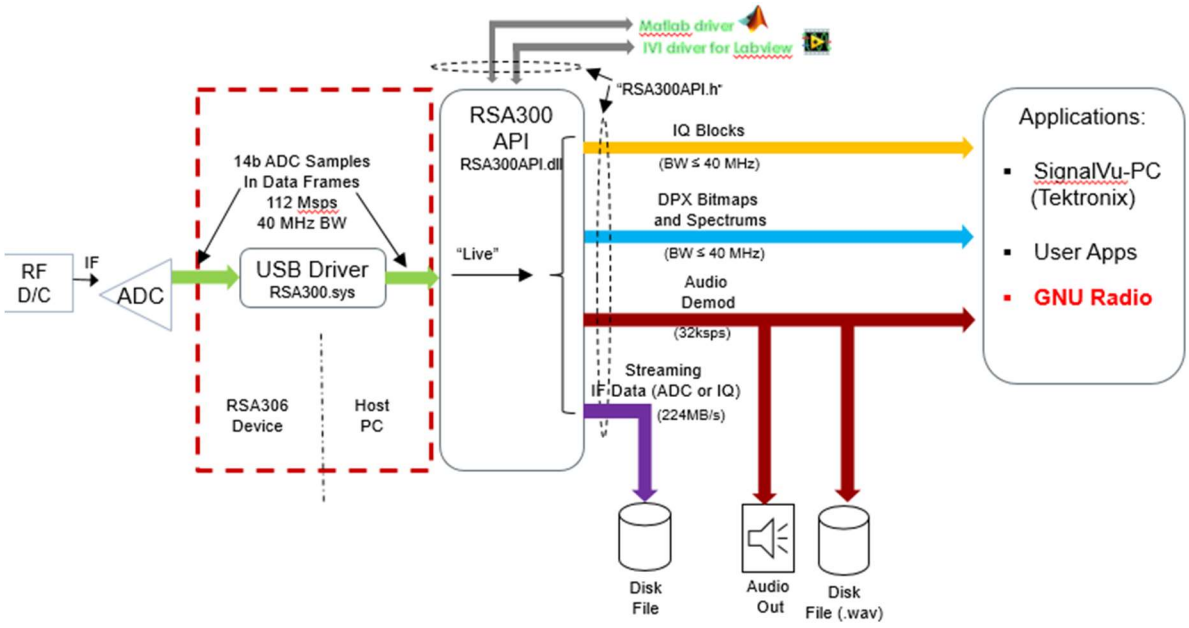

Figure 1. Process Overview

4.0 Discussion

4.1 Design Consideration Table

Table 1. Design Consideration Table

Category	Consideration
Economic – Industry User Cost	No cost
Economic – Tektronix Cost	No cost
Economic – University of Portland Cost	Less than give budget of $250
Ethical	Prevent unethical use of device to compromise personal security
Environmental	No environmental impact
Political	No political impact
Health and Safety	No overheating of device as it could damage personal health and industry equipment. Report errors that would damage equipment or personal health
Manufacturability	Must be easy to maintain after Tektronix assumes control after project completion
Sustainability	Project must be able to be deployable for new versions of GNURadio and Ubuntu
Social	No social impact

In order for our project to be successful our project must come at no cost to Tektronix (not including borrowed devices/components). This is because our project must be available to Tektronix's clients for free. The University of Portland has supplied us with a $250 budget for completing our project therefore, our costs to the university must not exceed this amount. The main design consideration for us was ethical implications. Our project has the potential to be used to breach personal security rights. Therefore, we needed to take this into account when designing our project that is both easily accessible and powerful. Our project must not harm the user and be able to report possible harm that the device may cause. Since our device is outputting a lot of data very quickly the device is at risk to overheat which may damage the device and the user. Therefore, in our project we had to keep in mind which errors must be reported in order to alert the user to problems that could cause bodily harm and/or machine damage. Our project needs to be sustainability in regards to how easy it is to upgrade to the next version. Both Linux Ubuntu OS and GNURadio upgrade regularly therefore, our code needs to be easily understandable and well commented such that a Tektronix computer scientist can upgrade our code to the new upgrades.

4.2 Design Criteria Table

Table 2. Design Criteria Table

Category	Requirement
Operating System	Linux Ubuntu 15
Coding Language	C++
Supported Devices	RSA300 and 500 Series
Code Compliance	GNURadio Standards 3.7.11
Data Outputs	IQ block, IF stream and IQ stream

Acquisition Status Reporting	Report errors and user settings

The above table is an abbreviated version of the requirements given to us by Tektronix. The full Tektronix requirements document can be found in the Appendix in Section 8.4.

Our project was required to be deployed in Linux Ubuntu 15 since GNURadio can only be used in the Linux Operating System. Our project had to be programmed in C++ as this is the underlying programming language for GNURadio. Tektronix has three different types of RSAs: 300, 500, and 700 series. For our project, Tektronix required that we supported the 300 and 500 series. In order for our project to be made available to Tektronix clients via GNURadio our code needed to pass GNURadio coding standards set out in their coding standards documentation. Our code also needed to support IQ block, IF stream, and IQ stream data output types as well as successfully report errors and user settings.

These requirements are discussed more fully in Section 4.4.

4.3 Alternative Designs

During our design process, we had a couple of alternative designs that were not chosen over the final design. One of these designs were having a different output data system. Instead of outputting I and Q together as one array as done in the final design, we would output I and Q separately. This meant our block would have two output ports. This idea was ultimately decided against because GNURadio only supports I and Q data outputted in a single array not separately. Another alternative design was the idea to combine all three data types (IQ block, IQ stream, and IF stream) into the same block such that we only create one GNURadio block. This was decided against because the GNURadio coding standards didn't support such a capability.

4.4 Code Design

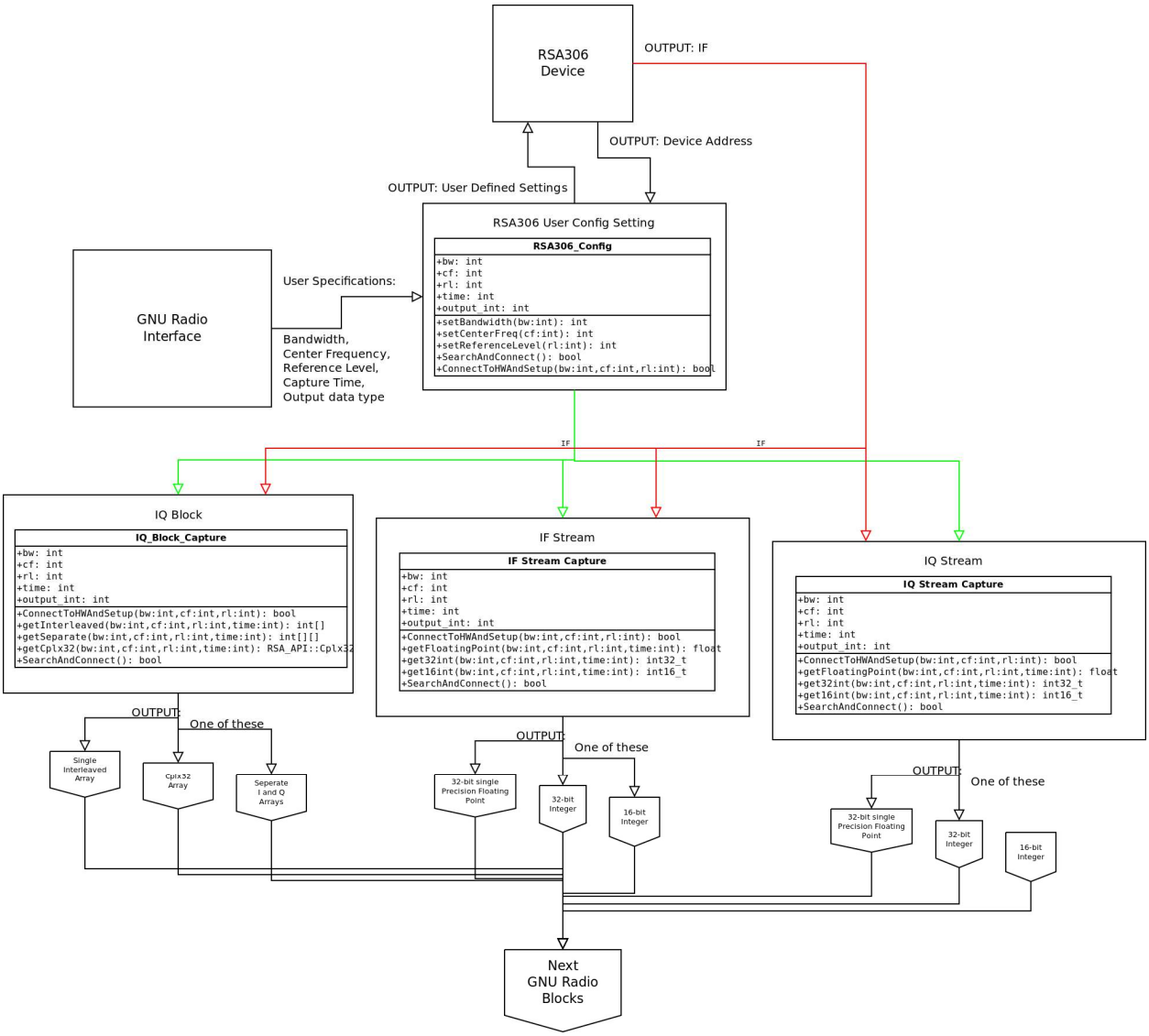

Figure 2. Block Diagram of Current Design

Upon the launching of GNU Radio, it will be up to the user to manage how the RSA306 blocks are used. The user will be able to choose from one of three blocks: IQ block capture, IQ Stream capture and IF Stream capture. In each block's interface, the user will be able to change the bandwidth, center frequency, reference level, capture time and output type. Once the information is entered into the block, and the block has an output, the GNU Radio program can start. Once the program starts, the user's capture specifications will be set on the RSA306.

Each block has relatively similar functionality. They all take bandwidth, center frequency, reference level, capture time and output type as inputs; only the output type will vary from block to block. Each block first acquires the RSA306 hardware and API. Then, the functionality of each block starts to differ.

For the IQ block capture, the RSA will be outputting blocks of *n* size every *t* milliseconds. The IQ block capture program is meant to take the output IF data from the RSA306 every *t* milliseconds. Once it has one block, it will convert that to the user-specified output. For the single interleaved array, it will put I in even numbered cells of an array: {0,2,4...2*j*}, and Q data in odd numbered cells: {1,3,5...2*k*+1}. For the Complex 32 array, there is no conversion needed, as that is the data that comes from the RSA306. For the separate I and Q arrays, the data will be split into two arrays, I and Q, where each cell number corresponds to the other array's equivalent cell.

For the IQ and IF streams, data will simply be converted to the user-specified data type. These data types include 32-bit float, 16-bit integer and 32-bit integer. For each output in IQ and IF stream, the data will be outputted and converted as fast as the RSA306 outputs the data, thus the need for a fast processor (and if the GNU radio user outputs to a file, they will need an SSD).

Once the data is outputted from any one of these capture programs, it is then passed to the next component in the user's GNU Radio setup.

4.5 Testing Procedures

In order to test our code, we had to develop test programs that used our GNURadio block in a GNURadio program. For both IQ block and IQ stream, we outputted the collected data into a text file, using GNURadio, then used a different programming environment, MATLAB, to extract the data and manually verified the outputted was correct. After verification, we then implemented a FM Radio using GNURadio. After we got a clear, audible sound we sent our code to Tektronix for further testing. Tektronix would then send us the data about errors that occurred and after fixing them we would repeat this process until no errors remained.

4.6 Engineering Standards

For GNURadio to accept a new piece of code in its library, the code first has to be inspected and pass their coding standards. These standards are put in place to ensure a uniform coding style that prevents errors and creates clear documentation so others may alter the code in the future. Therefore, in order for our project to be used in GNURadio we had to abide by their coding standards. These standards included how variables are named, how many comments must be in the program, and how long these comments can be.

4.7 Professional Responsibility

Whenever you are dealing with radio frequency research one needs to be aware of the security dangers that accompany it. Since the radio frequency spectrum is widely used in communication systems and other personal information carrying services security is an important ethical question. Our project enables a low-cost device with high-end capability thus opening the door to potential personal security violations (including cell phone hacking, car key fob hacking, wi-fi hacking, etc.). While we do not advise, or intend our project to be used for such means it is possible it will be. We can’t control what our project is used for therefore, we urge users to use

our product ethically, advocate for continuing ethical discussion surrounding radio frequency hacking, and advocate for continuing research and development in security systems to protect personal information.

5.0 Finances

Since Tektronix provided the RSA and the Shiley School of Engineering provided other equipment, this project was budget free. However, for investment purpose, the costs of those equipment are estimated as below.

Table 3. Estimated Equipment Cost

Item	Vendor	Quantity	Price
RSA306	Tektronix	1	$3,890.00
Wide Band Antenna	Sandia National Lab	1	$ 39.98
High End Computer with SSD	University of Portland	1	$1,700.00
SignalVu	Tektronix	1	$ -
Total Cost			$5,629.98

Table 4. Rent Budget

Lab/Room	Hrs/Week	Duration (Month)	Rent/month
306	5	8	$ 500.00
Total Rent Cost			$ 4,000.00

Table 5. Developer Budget

Developers	Hrs/Individual	Rate/Hr	Total Developer Budget
5	50	$50.00	$ 12,500.00

Table 6. Consultant Budget

Area of Need	Consultant	Hours Anticipated	Rate/Hr	Final Cost
Advising Meetings	Dr. Hoffbeck	32	$150.00	$4,800.00
CS Technical Support	Kyle Bernard	10	$150.00	$1,500.00
Total Consultant Budget				$6,300.00

Table 7. Overall Development Cost

Total Equipment Cost	$ 5,629.98
Consultant and developer cost	$12,500.00
Rent Cost	$ 4,000.00
Total Overall Development Cost	$22,129.98

Table 8. Cost of unit per software package

Item	Price
RSA306	$3,890.00
Software Package	$ 50.00
Actual Selling Price	$3,940.00
Final Selling Price	$3,999.99

Table 9. Total Expenses

Expenses			Year	2017	2018	2019	2020	2021	2022	2023
			Growth							
Quantity		20	20%	20	24	29	35	42	50	60
Cost of Goods Sold										
	Fixed	$ 79,999.80	2%	79999.8	$ 81,599.80	$ 83,231.79	$ 84,896.43	$ 86,594.36	$ 88,326.24	$ 90,092.77
	Variable	$ 1.00	2%	1	$ 1.00	$ 1.02	$ 1.04	$ 1.06	$ 1.08	$ 1.10
Total Cost of Goods				$ 80,019.80	$ 81,623.80	$ 83,261.37	$ 84,932.84	$ 86,638.93	$ 88,380.37	$ 90,159.01
Outside Services										
	20%			$ 16,003.96	$ 16,324.76	$ 16,652.27	$ 16,986.57	$ 17,327.79	$ 17,676.07	$ 18,031.80
Research and Development										
	17%			$ 13,603.37	$ 13,876.05	$ 14,154.43	$ 14,438.58	$ 14,728.62	$ 15,024.66	$ 15,327.03
Total Expenses				$ 109,627.13	$ 111,824.60	$ 114,068.08	$ 116,357.99	$ 118,695.33	$ 121,081.10	$ 123,517.85

Table 10. Total Revenue

		Year	2017	2018	2019	2020	2021	2022	2023
From expense Tab	Units		20	24	29	35	42	50	60
Cost of Goods Sold			$ 109,627.13	$ 111,824.60	$ 114,068.08	$ 116,357.99	$ 118,695.33	$ 121,081.10	$ 123,517.85
Target Gross	30%		$ 32,888.14	$ 33,547.38	$ 34,220.42	$ 34,907.40	$ 35,608.60	$ 36,324.33	$ 37,055.35
	Hours	Raw							
Revenue Consulting	42	150	$ 6,300.00	$ 6,300.00	$ 6,300.00	$ 6,300.00	$ 6,300.00	$ 6,300.00	$ 6,300.00
Consulting with markup	50%		$ 3,150.00	$ 3,150.00	$ 3,150.00	$ 3,150.00	$ 3,150.00	$ 3,150.00	$ 3,150.00
Revenue from Royalties			$ -	$ -	$ -	$ -	$ -	$ -	$ -
Total Revenue			$ 145,665.26	$ 148,521.98	$ 151,438.50	$ 154,415.39	$ 157,453.93	$ 160,555.43	$ 163,723.20

Table 11. Income Statement

Income Statement		Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7
Revenue		145,665	148,522	151,439	154,415	157,454	160,555	163,723
Operations & Maintenance		109,627	111,825	114,068	116,358	118,695	121,081	123,518
Operating Income (EBITDA)		36,038	36,697	37,370	38,057	38,759	39,474	40,205
Depreciation		563	563	563	563	563	563	563
Interest Expense		113	$108	$103	$98	$93	$88	$83
Net Income (EBIT)		35,475	36,134	36,807	37,494	38,196	38,911	39,642
Net Income (EBT)		35,363	36,026	36,704	37,396	38,102	38,823	39,559
Income Tax	37.02%	13,091	13,337	13,588	13,844	14,105	14,372	14,645
Net Income After Taxes		22,271	22,689	23,116	23,552	23,997	24,451	24,914

Table 12. Balance Sheet

Balance Sheet	End of	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7
Assets								
Gross Equipment		5,630	5,067	4,504	3,941	3,378	2,815	2,252
Accumulated Depreciation		(563)	(563)	(563)	(563)	(563)	(563)	(563)
Net Equipment		5,067	4,504	3,941	3,378	2,815	2,252	1,689
Total Assets		5,067	4,504	3,941	3,378	2,815	2,252	1,689
Liabilities and Owners Equity								
Common Equity		2,533	2,252	1,970	1,689	1,407	1,126	844
Debt		2,533	2,252	1,970	1,689	1,407	1,126	844
Deferred Taxes		-	-	-	-	-	-	-
Net Liabilites and OE		5,067	4,504	3,941	3,378	2,815	2,252	1,689
Total Liabilities		5,067	4,504	3,941	3,378	2,815	2,252	1,689

6.0 Conclusions and Recommendations

The GNU Radio Interface for RSA306 senior design project set out to increase the capability of the RSA306 Spectrum Analyzer by enabling it to connect to the open-source coding platform GNU Radio. This project was completed on schedule by April 11, 2017 in time to be presented at the University of Portland's Founder's Day. The primary objectives of devolving a block for GNURadio, IQ block capability, IQ stream capability, and RSA500 support were completed. The primary objective of IF stream capability was removed as a requirement by the client, Tektronix.

Our recommendations for continuing work on our project are to maintain our code as the Linux operating system, Ubuntu, and GNURadio upgrade, add support for the RSA700 series, and add IF streaming capability. Our project will be used in an upcoming University of Portland senior design project that seeks to use the RSAs, GNURadio, and our project to location a person using triangulation. This project can be extended in the future to include a smart phone app that, using

a system of RSAs, will locate a person on campus, display their location on their phone, and offer directions from their location to a building on campus.

7.0 References

"Guided GNU Radio Tutorials." *Guided Tutorials - GNU Radio*. GNU Radio, n.d. Web. 12 Sept. 2016. http://gnuradio.org/redmine/projects/gnuradio/wiki/Guided_Tutorials

"RSA306B USB Spectrum Analyzer." *RSA306B USB Spectrum Analyzer | Tektronix*. Tektronix, n.d. Web. 12 Sept. 2016. http://www.tek.com/spectrum-analyzer/rsa306

"System Requirements for MATLAB R2016b." *MathWorks*. MathWorks, n.d. Web. 28 Nov. 2016. http://www.mathworks.com/support/sysreq/current_release/index.html?sec=linux.

8.0 Acknowledgments

As with any engineering project we couldn't of completed our project without the generous help of many different people and institutions. We would like to thank our industry advisor, Kyle Bernard, whose technical advice helped guide our project through some rough patches and who helped secure the RSA306 and RSA506 for us to use in completing our project. Faculty advisors Dr. Joseph Hoffbeck and James Schmidt, for their technical advice. Our capstone instructor Christ Galati who not only helped us by aiding our project but helping us grow as professionals. Dean Sharon Jones of the Shiley School of Engineering for funding our project. We'd also like to thank Tektronix and Sandia National Laboratories for lending us equipment for development.

9.0 Appendices

9.1 Weekly Status Report

9.1.1 Week 9/23/2016

Accomplished:

- Met with Kyle Bernard (Tektronix) to go over the project requirements
- Received RSA306 and software installation

On Going:

- Get started with SignalVu and the RSA306
- Get familiar with API
- Reschedule weekly meeting
    - Group meeting on Monday at 6pm
    - Advisor meeting (with Dr. Hoffbeck) on Friday at 3:45 pm

Looking Forward:

- Get access to the CS LAB in 3rd floor Shiley

- Set up meeting with CS advisor

9.1.2 Week 9/30/2016

Accomplish:

- Contacted Dr. Schmidt (CS advisor) for software needs
    - Computer with Linux installed
    - SignalVu and Eclipse installed

On Going:

- Getting the website uploaded
- Filling IS tickets to get computer with software installed
    - Attempting to use personal computer to get the project started
- Researching on GNU radio and IQ block data handling
- Researching and applying Agile/Scrum to the development plan

Short Term:

- Have the hardware/software ready in the computer lab Shiley 304
- Establish a working connection between RSA306 and computer
- Have periodic reviews on each development process.

Long Term:

- Review and test the developed code (Beta release)

9.1.3 Week 10/07/2016

Accomplished:

- Filled out IS tickets for software access on UP
- Got access to the Website domain
- Met with AJ Allen for marketing consulting to present our product
- Establish a working connection between RSA306 and computer

On Going:

- Website skeleton created and design course for the final version
    - Give Oakes Max (Webmaster) access to the website
- Finish tutorials for the GNU Radio
    - Able to use SignalVu's functionality on Wi-Fi/radio signals

Short Term:

- Have the hardware/software ready in the computer lab Shiley 304

- Start to develop the IQ block.
- Have periodic reviews on each development process.

Long Term:

- Review and test the developed code (Beta release)

9.1.4 Week 10/28/2016

Accomplished:

- Finalized project budget
- Website
    - Link access: https://engineering.projects.up.edu/wagnern17/
    - Basic layout built
    - Team profiles completed
- Completed project's presentation for Capstone class.
- Enabled computer connection to the API and installed necessary software for code development.
- Learned and implemented Scum/Sprint.
    - Our first sprint is attached.
    - [For Kyle] I understand that our scrum will need to be expanded more, but we'd appreciate if you could give us some feedback on our first scrum?

On Going:

- API:
    - Enabled computer connection to the API.
    - Working on accessing the RSA306 using API.
- Website:
    - Finish preliminary development of team website.

Constraint:

- Couldn't reserve the desktop workstation in Shiley 304.
    - Development process is currently done on personal laptop.
    - We are working on obtain another computer with Ubuntu installed.

9.1.5 Week 11/04/2016

Accomplished:

- Set up computer in Shiley 304
- Modified the A3
- Updated Scum/Sprint, and website

On-going (next week):

- API
    - Working on resolving the connectivity between the API and computer
        - Getting error RSA500 share lib while compiling the API. [Max and Kyle are looking toward the solution]
    - Designing structure (classes, cases, etc.) for the API.
- Establish GitHub account
- Update Gant Chart
- Modify and update Sprint/Scrum with more specific details.

9.1.6 Week 11/11/2016

Accomplished:

- Updated Gantt Chart
- Updated Sprint/Scrum for this week
- Resolved major bug (RSA500 lib error)
- Established GitHub repository and team account
- Code design
    - Nick and Max discussed the coding structure, classes used for obtaining data
    - Reviewed the IF and IQ block diagram with Dr. Hoffbeck

On-going

- Code design
    - Continue to develop the structure, classes, used cases, etc.
    - Researching more on data stream format (IF and IQ)
- Outline the capstone project report
- Begin poster development

9.1.7 Week 11/18/2016

Accomplished:

- Beta stage poster completed
- Capstone Project report outline completed

On-going

- Code design
    - Continue define and construct classes, uses case, structure for the API
    - Researching more on data stream format (IF and IQ)
    - Attempt to test (decode) the received data using Matlab
- Create a short GitHub guide to eliminate ambiguity
- Team meeting with Kyle Bernard
    - Time: 4-5pm
    - Location: Shiley conference room

- Updating scrum/sprint for this week

9.1.8 Week 12/2/2016

Accomplished:

- Turned in the draft report
- Met with Kyle
    - Narrowed down the scope of the project
    - Got information on r3f file and streaming files

On-going

- Code design
    - Continue to define and construct classes, uses case, structure for the API
- Project report
    - Continue to work on sections/subsections
    - Focus on the executive summary and design portion
    - Report is due on Fri 9th
- Poster
    - Need to update the design portion to complete the poster
    - Poster presentation section: TBA
- Updating scrum/sprint for this week (attached)

9.1.9 Week 1/27/2017

Accomplished:

- Implementation:
    - Able to output the basic functionalities and device configurations from the API
        - Output .csv file (raw IQ data in complex 32 format)
    - Schedule
        - Updated the Gantt Chart/Sprint for Spring

On Going:

- Data Testing:
    - Verifying .csv data with Matlab/GNU Radio
- Acquisition Report
    - Implementing the input overage and RSA sample rate
- Project PowerPoint
    - Creating powerpoint to further explain the RSA306 and GNU Radio

9.1.10 Week 2/4/2017

Accomplished:

- Implementation:
    - Verified that output data was correct for 1 block of data
- Using input sources: captured radio station and sin wave from a function generator
    - Changed how output is saved to not lose any data during collection
- Data from blocks is not put into a buffer on disk, then written to a .csv afterwards, allowing us to keep up with the output data

On Going:

- Data Testing:
    - Verifying the output data from our code is correct for multiple block collection
    - Build and submit the GNU Radio code block to Tektronix for review

9.1.11 Week 2/10/2017

Accomplished:

- Implementation:
    - Verified that output data was correct for all block data
    - Converted clean data to .wav and clear heard the sampled signal
    - Completed data allocation on disk

On Going:

- Finish GNU Radio code block for Tektronix review
- Acquisition Report
    - Set up device acquisition settings

9.1.12 Week 2/17/2017

Accomplished:

- Added basic acquisition settings to IQ block
- Verified our IQ block was working correctly on two sets of computers
- Completed the IQ block and sent to Tektronix for code review
- Fixed small errors in IQ block after code review

On Going:

- Next week we will start working on the IQ streaming block
- We also will start looking at and planning some demos we can accomplish with GNU Radio and the RSA to show-off our project during Founder's Day presentations

9.1.13 Week 2/24/2017

Accomplished:

- Received IF streaming documentation from Tek
- Got a legitimate sample from the IQ stream before the buffer overloaded

On Going:

- Continuing development of the IQ Streaming Block
- Research GNURadio applications for demo's

9.1.14 Week 3/10/2017

Accomplished:

- Properly updated our comments, prints, and naming conventions in the IQ Stream block
- Properly implemented device/stream stop/starts on parameter changes

On Going:

- Get IQ Stream Buffer working properly
- Add Ubuntu 14.04 LTS functionality
- Look for miscellaneous compatibility errors in sending the block to Tek

9.1.15 Week 3/24/2017

Accomplished:

- Completed the FM Radio Demodulation in GNU Radio to allow streaming of FM Radio
- Worked on demodulation of TV signals

On Going:

- Bring source code up to date with GNU Radio coding guidelines

9.1.16 Week 3/31/2017

Accomplished:

- Updated our code to meet GNU Radio standards
- Worked on FM Radio demodulation for our presentation
    - Output speech is very clear and can switch between stations when running

On Going:

- Continuing to work on TV signal demodulation to create either a live streaming or streaming from a file demo

- Work on AM demodulation
- Work on Founder's Day Presentation (slide show and physical presentation)

9.1.17 Week 4/7/2017

Accomplished:

- Completed updating our code to meet GNU Radio standards
- Completed key fob demo for our presentation
- Completed wifi demo for our presentation
- Completed our presentation slides and submitted them to Dr. Hoffbeck for review/editing
  - Edited and resubmitted
- Allowed our code to be accessed by the RSA500
- Assigned presentation roles to each member to begin presentation dry runs

On Going:

- Complete multiple presentation dry runs to ensure smooth transitions and no awkward slides
- Founders Day Presentation at 3:15 in Shiley 301
- Resolve any residual errors to complete RSA500 compatibility

9.2 Gantt Chart

A virtual version of the Gantt chart can be accessed through the team website at the following link: https://engineering.projects.up.edu/wagnern17/schedule/.

The Gantt chart can be seen below:

GNU Radio Interface

Team GNU

Today's Date: 4/28/2017 Friday

(vertical red line)

Project Lead: Nick Wagner

Start Date: 9/5/2016 Monday

First Day of Week (Mon=2): 2

WBS	Tasks	Task Lead	Start	End	Duration (Days)	% Complete	Working Days	Days Complete	Days Remaining
1	**Primary**	Quentii	9/5/16	9/20/16	16	100%	12	16	0
1.1	Organize Weekly Meetings		9/5/16	9/6/16	2	100%	2	2	0
1.2	Decide Leadership Roles		9/6/16	9/7/16	2	100%	2	2	0
1.3	Finalize Budget		9/6/16	9/13/16	8	100%	6	8	0
1.4	Finalize Delegate Responsibilities		9/8/16	9/20/16	13	100%	9	13	0
1.5	Construct Website		9/12/16	9/15/16	4	100%	4	4	0
1.6	Upload Team		9/15/16	9/22/16	8	100%	6	8	0
1.6.1	Upload		9/15/16	9/18/16	4	100%	2	4	0
1.6.2	Upload Schedule		9/15/16	9/18/16	4	100%	2	4	0
1.6.3			9/17/16	9/20/16	4	100%	2	4	0
2	**Prep for**	Quentii	9/23/16	11/4/16	43	100%	31	43	0
2.1	Get Computer for Development		9/23/16	11/4/16	43	100%	31	43	0
2.1.1	Install Development		11/1/16	11/4/16	4	100%	4	4	0
2.2	Familarize team with SignalVu		9/29/16	10/13/16	15	100%	11	15	0
2.3	Learn and Implement Scrum Development		10/20/16	10/29/16	10	100%	7	10	0
2.4	Establish Github account		11/1/16	11/2/16	2	100%	2	2	0
3	**Design**	Quentii	11/1/16	1/15/17	76	100%	54	76	0
3.1	Enable Computer Connection to API		11/1/16	11/14/16	14	100%	10	14	0
3.2	Primary Design		11/9/16	11/11/16	3	100%	3	3	0
3.3	1st Draft of Design		11/14/16	11/20/16	7	100%	5	7	0
3.4	2nd Draft of Design		11/20/16	12/3/16	14	100%	10	14	0
3.4.1	First Review Session		12/5/16	12/18/16	14	100%	10	14	0
3.5	Final Semester		11/20/16	12/7/16	18	100%	13	18	0
3.6	3rd Draft of Design		12/18/16	1/15/17	29	100%	20	29	0
4	**Code**	Nick W	1/16/17	4/9/17	84	100%	60	84	0
4.1	Enable Communication to		1/16/17	1/22/17	7	100%	5	7	0
4.2	Get IQ Block Data		1/22/17	2/4/17	14	100%	10	14	0
4.2.1	2nd Code		2/4/17	2/27/17	24	100%	16	24	0
4.3	Get IQ Stream Data		2/27/17	3/27/17	29	100%	21	29	0
4.3.1	3rd Code Review		3/12/17	3/21/17	10	100%	7	10	0
4.4	Make Code GNU Capable		3/27/17	4/1/17	6	100%	5	6	0
4.5	RSA500		4/12/17	5/1/17	20	100%	14	20	0
5	**Founder's Day Preperations**	Nick W	3/27/17	5/1/17	36	100%	26	36	0
5.1	Create		4/1/17	4/7/17	7	100%	5	7	0
5.2	Demonstrations		3/27/17	4/9/17	14	100%	10	14	0
5.3	Practice		4/7/17	4/10/17	4	100%	2	4	0
5.4	Final Checks		4/9/17	4/10/17	2	100%	1	2	0
5.5	Founder's Day		4/11/17	4/11/17	1	100%	1	1	0
5.6	Create Tektronix Presentation		4/12/17	4/28/17	17	100%	13	17	0
5.7	Tektronix Visit		4/28/17	4/28/17	1	100%	1	1	0
5.8	Final Report		4/12/17	5/1/17	20	100%	14	20	0

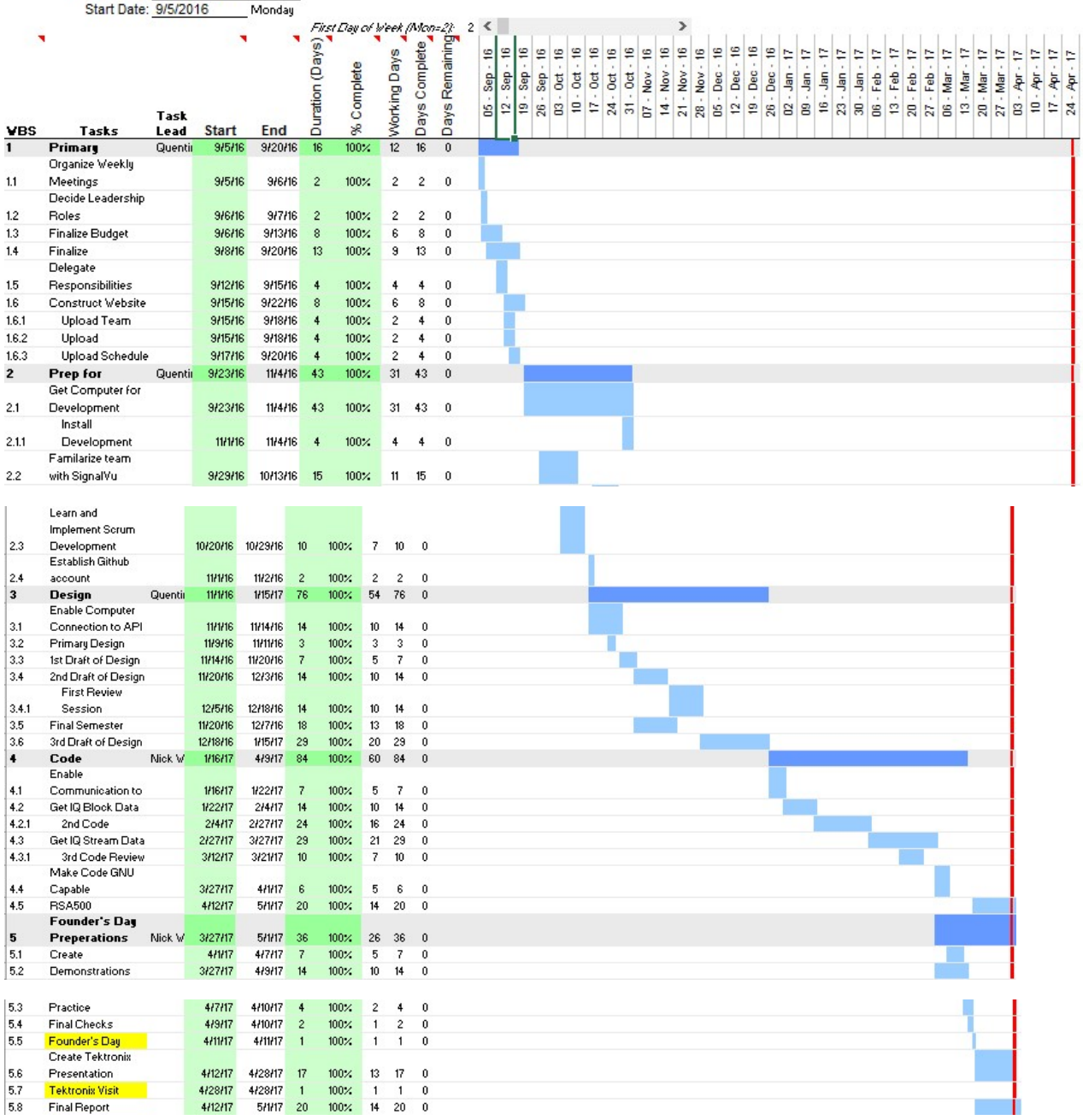

Figure 3. Gantt Chart

9.3 RSA306

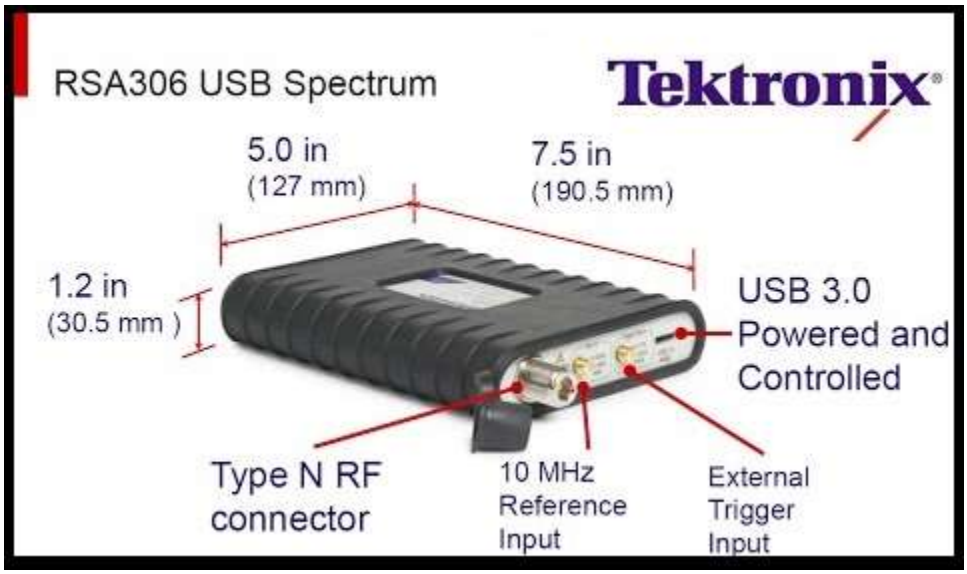

Figure 4. RSA306 Diagram

9.4 Tektronix Deliverables

Priority E=Essential, D=Desired SE=Excluded	MVP (min. viable product)	Description	Requirement	Notes
	General			
E	MVP	GNURadio source module	GNURadio OOT source block(s) shall be developed which support Tek RSA acquisition devices.	
	Source Data			
E	MVP	IQ Block	GNURadio Tek RSA source module shall supply voltage scaled IQ block data	Separate GNURadio OOT source block
E	MVP	IQ Block data format	single interleaved array	
E		IQ Block data format	Cplx32 array	
E		IQ Block data format	separate I and Q array	
E	MVP	IF Stream	GNURadio Tek RSA source module shall supply voltage scaled IF streamed data	Separate GNURadio OOT source block. Tek to add this to the RSA API.
E	MVP	IF Stream auxiliary info	actual CF at IF, gain scaling, channel EQ data,	Auxiliary info is supplied by the RSA API for frequency correction, data scaling to voltage, and channel equalization (if user selects). Tek to add this to the RSA API.
		IF Stream data format	32-bit single precision floating point	Tek to add this to the RSA API?
		IF Stream data format	32-bit integer	Tek to add this to the RSA API?
E	MVP	IF Stream data format	16-bit integer	Tek to add this to the RSA API
D		IQ Stream	GNURadio Tek RSA source module shall supply voltage scaled IQ streamed data	Separate GNURadio OOT source block
D		IQ Stream data format	32-bit single precision floating point	
D		IQ Stream data format	32-bit integer	
D		IQ Stream data format	16-bit integer	
	Coding/Deployment			
E	MVP	C++	Tek RSA source module(s) shall be written in C++	
E	MVP	Source code style	Module/block source code shall follow the GNURadio Coding Guide	GNR Coding Guide
E	MVP	Deployed source code	Module/block source code shall be publically deployed as Open Source code	
E	MVP	Publishing	Tek RSA source module(s) shall be published as a standard GNURadio OOT module	Configuring GNURadio and OOT modules
E	MVP	Operating system	Tek RSA source module shall be deployed for Ubuntu 15	
D		Operating system	Tek RSA source module shall be deployed for Ubuntu 14	
	Acquisition Device Configuration			
E	MVP	Center Frequency	Settable at run time over full range of the connected RSA device.	
E	MVP	Reference Level	Settable at run time over full range of the connected RSA device.	
E	MVP	Acquisition Bandwidth	Settable at run time over full range of the connected RSA device.	
E	MVP	Channel Equalization	Settable at connect time to on/off	IF streaming only - user selects if channel equalization is enabled.
E	MVP	Block Length	Settable at run time over full range of the connected RSA device.	IQ block source
E		Trigger Source	Settable at run time to values: Freerun, IF Power, External	
E		IF Trigger Level	Settable at run time over full range of the connected RSA device.	
E		Trigger Transition	Settable at run time to values: L>H, H>L, L<>H	
E		Trigger Position	Settable at run time to position within the IQ block record	
E	MVP*	RF Attenuation	Settable at run time to values: Auto or specific manual setting	RSA500/600 only. * User control not required for MVP but must Default to Auto
E	MVP	RF preamp	Settable at run time to values: On/Off	RSA500/600 only. Default = Off
	Acquisition Devices Supported			
E	MVP	RSA300 series devices	RSA306, 306B	
E	MVP	RSA500 series devices	RSA503, 507, 603, 607	

	Device Functions			
E	MVP	Search and connect	Search USB bus for all connected RSA devices	
E	MVP	Search and connect	Connect to a single selected RSA device	
E		Report device info	FPGA version, FW version, HW version, device nomenclature, device serial number, API version	
E	MVP	Alignment	Manual alignment	User chooses when to perform alignment. Alignment status reported to user. Note: For RSA306 (and 306B) input signals can corrupt alignment, so need to prompt user to disconnect input.
D		Alignment	Auto alignment (RSA500/600 only)	Detect when alignment is needed, pause flow, perform alignment, resume flow.
	Acquisition Status Reporting			
E	MVP	Acquisition status - **current data output from source block**	Input overrange	acqStatus as described in the RSA API documentation.
E	MVP		RSA sample rate	sample rate used for current data
E			Timestamp	timestamp of the first sample returned
E			Trigger indicies	list of sample indicies where trigger events occurred
E			Trigger timestamps	list of timestamps where trigger events occurred
E	MVP	Acquisition status - **overall for source block**	USB data stream discontinuity	acqStatus as described in the RSA API documentation.
E	MVP		Input buffer overflow	acqStatus as described in the RSA API documentation.
E	MVP		Output buffer overflow	acqStatus as described in the RSA API documentation.
E			batteryOverTemperature	RSA500 only. Power Info as described in the RSA API documentation.
E			batteryHardwareError	RSA500 only. Power Info as described in the RSA API documentation.
E			RSA device over temperature operation	as in DEVICE_GetOverTemperatureStatus
	Time Functions			
SE		Basic reference time	Set/Report reference time (basic)	Automatic in the latest API
SE		High precision reference time	Set/Report reference time based on GPS/GNSS event (e.g., 1 PPS)	Automatic in the latest API

9.5 Total Team Hours Spent

As listed in Table 5, the total team hours spent on this project is 250 hours.