REDHAWK is new, open source software based on the Software Communications Architecture (SCA) 2.2.2 designed to create applications to simplify the rapid development of real-time Software Defined Radio (SDR) and systems. The REDHAWK Integrated Development Environment (IDE), based on Eclipse, provides a complete environment for all aspects of SDR development including: source code generation, graphical drag-and-drop waveform construction, runtime introspection of systems, integration with debuggers, and advanced signal visualization [1]. Waveform applications that use the RTL 2832-based SDR were developed and tested using custom and available REDHAWK signal processing components. Although initial REDHAWK documentation has been released, it is still under development; therefore, processes were documented and shared to broaden the community of users through YouTube tutorials and written documentation.

Increased documentation on REDHAWK will increase the amount of information in the open source environment. In addition to creating more advanced waveforms, an increase of information will also help users to troubleshoot problems. If the user is able to increase the readily amount of information on REDHAWK the process of fixing errors to create more accurate radio systems will be enhanced.

This research project increased the documentation for REDHAWK by creating custom components and an FM receiver. Troubleshooting problems in the domain manager were also documented. The information was recorded in written documentation and a YouTube channel.

2. BACKGROUND

Based in part on Virginia Tech’s OSSIE software, The REDHAWK Core Framework extends the US DOD Joint Tactical Radio System (JTRS) SCA version 2.2.2. The REDHAWK IDE, based on Eclipse, provides a complete environment for all aspects of SDR development, including: source code generators, drag-and-drop construction of applications, runtime introspection of systems, integration with debuggers, and advanced signal visualization [1]. Eclipse is an open source project to create powerful IDEs for well known high level languages like Java or C++. The REDHAWK framework and Integrated Development Environment (IDE) are designed to be user-friendly environments.

REDHAWK is a new open source software created by the United States Government (USG) Department of Defense (DoD) designed to facilitate the rapid development of SDR applications and systems. The first version of REDHAWK was released in March 2013 so information will be limited. A review of internet tutorials on REDHAWK showed 12,600 documents related to SDR tutorials. Of these documents, 3,600 were related to GNU Radio. The Axios REDHAWK YouTube channel at Axios Engineering.com contains two videos on the installation and the basics of the REDHAWK IDE. The REDHAWK Manual 1.8.4 is the main source of information for learning.
REDHAWK. “It is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems.[2]” “GNU Radio REDHAWK offers a software package called GNUHAWK which contains a library and shared components that enable GNU Radio blocks to be used within the REDHAWK software framework[3].”

3. METHODOLOGY

This project proceeds through these five phases.

![Figure 2. Phases of the project](image)

3.1. Learn how to use REDHAWK

Starting with Phase I the research group learned how to use REDHAWK. In order to learn how to use REDHAWK the REDHAWK Manual 1.8.4 and REDHAWK Overview Manual 1.8.4 was read. Besides the reading, the two video tutorials of REDHAWK created by Axios found on YouTube were studied and additional documentation of basic components in the REDHAWK git repositories was used as reference.

3.2. Create waveforms and components. Create an FM receiver in the Sandbox IDE.

After the group learned how to use REDHAWK, Phase II started. During Phase II the participants tested the REDHAWK IDE creating different waveforms and components using the basic components from REDHAWK like: SigGen, HardLimit and components from GnuHawk integration. Using these components simple waveforms was created but since the library of components is limited, creating different waveforms is also limited. In order to fulfill the main objective which is creating an FM Receiver it was fundamental to learn the basics and then learn to create processing components to be able to have all the components needed to create an FM Receiver. The ConverterFloatToShort component was needed to convert from the float data type to short data type because the AudioSink input port receives only the short data type. With the creating of the ConverterFloatToShort all the components needed to build an FM receiver were available.

To start explain the Radio Receiver created first the theory behind the FM Receiver will be explained. FM radio waves encode the information varying the frequency of the carrier signal and leaving the amplitude is constant. The amplitude will normally vary as a result of noise.

The general idea of an FM Receiver is to receive an FM signal that has noise because when the wave is travelling from the FM transmitter to the FM receiver the signal will be combined with noise. After the signal is received the limiter will remove the amplitude variation that is produced by noise and then it will output the demodulated signal in form of audio to the sound port.

For the project an FM Receiver was built using the REDHAWK IDE. The R820T SDR & DVB-T from StartTech.com was used with the adapted antenna provided to the group to receive the FM signal. The rtl_tcp application is part of the rtl_sdr software and is used to receive the FM spectrum. The FM Receiver was built in the Chalkboard from the Sandbox REDHAWK IDE. The chalkboard can be used to create a waveform using the components available to connect them and then launch the waveform. The chalkboard is very simple to use because it provides you a tab to add components to build the waveform using block diagrams. All components from REDHAWK IDE have a properties tab where you can change them in real time or before running a waveform. A block diagram of the FM Receiver is shown below.

The first component of the diagram is the RTLTcp_Souce component and the source of the FM Receiver. The RTLTcp_Source component is a component created by Axios and it needs to have the rtl_tcp application to be running in order to be used. The properties changed from default of the RTLTcp_Source component are the frequency, frequency correction and gain. The frequency is set to the FM station in Hz that the user wants to listen to.
The frequency correction is set to calibrate the SDR and the units are PPM (Parts per million). The gain is set to 20.1 dB to increase the strength of the signal from the input.

The second component is the TuneFilterDecimate. This component reduces the bandwidth of the signal and in such, reduces the sample rate so that the demodulator is required to demodulate fewer samples. The property changed from the default is the DesiredOutputRate which is changed to 150000 Hz.

The third component that is connected to the receiver is the AMFMPMBasebandDemodulator. This component can demodulate FM, AM or PM signals.

The next component is the AM FM PM Demod Port Selector. This component is a component created to receive the three types of demodulated signals from the AMFMPMBasebandDemod component to select the one you want to listen to. The purpose of this component is to be able to change the types of demodulation in real time for future work when the receiver is upgraded to an AM/FM Receiver. For the purpose of this project the FM selection is chosen.

The fifth component is another TuneFilterDecimate. The purpose of this second TuneFilterDecimate is to lower the sample rate to something that the sound card can handle. The DesiredOutputRate was set to 48000 Hz and the FilterBW to 15000 Hz.

The following component is the multiply_const_ff. This component is used to amplify the signal because after filtering the signal twice the signal is not that strong. The property k was set to 1e9 to amplify the signal.

After the signal is amplified the ConverterFloatToShort component is connected. This component was created to convert from a float data type to the data type short in order to be received by the AudioSink component which only accepts the short data type. Finally the AudioSink component was connected and this component outputs the audio to the sound port.

### 3.3. Create written documentation

During Phase III written documentation was created in order to fill some of the gaps of processes that are not explained with the REDHAWK Manual 1.8.4, the Redhawk website or the Axios video tutorials, but are not candidates for a video tutorial. Creating this written documentation is indispensable to share information that might be useful for the REDHAWK users.

### 3.4. Create YouTube tutorials

During the Phase IV the video tutorials were created to transmit the learned processes during the programs and uploaded to YouTube in order to share the knowledge with the REDHAWK community or with any audience interested in learning about some processes that they are not familiarized with in REDHAWK. In order to fulfill one of project’s objectives of documenting procedures, video tutorials were created to share them with the community. This method was chosen because it is easy to access and YouTube is a website that is used globally. There is documentation of REDHAWK but because the program was recently developed it needs further documentation, hence the video tutorials. When the group researched online for tutorials, only two video tutorials from Axios were found in YouTube. One of the videos shows how to install REDHAWK and the second one builds a waveform in the Sandbox IDE.

The key point of video tutorials is to use an effective way to transmit procedures in a visual and practical manner. Processes like creating signal processing components, build an FM receiver, Test a component using the Sandbox and creating a simple waveform are a few of the processes that the research group agreed are significant enough to learn and to get familiarized with the REDHAWK framework.

The process of creating video tutorials was done with the following order: learn a process, create a script, record...
the screen with audio, edit the video and upload the video tutorial to the YouTube Channel.

First the process was learned in order to give a tutorial. Some of the video tutorials were created using written tutorials as guidance. After the process is learned a script of the process is created and then the video is recorded using the FreeScreenToVideo program with the audio. After the video is recorded the Windows Live Movie Maker is used to add music to the tutorial and to do final edits of the video tutorial. When the video is ready it is uploaded to the YouTube channel.

3.5. Improvements for the FM Radio Receiver

Phase V goal is to make improvements to the FM receiver. When the first version of the FM Receiver was tested the FM Receiver was only using one TuneFilterDecimate and the sample rate property of the RTLTcpSource was set to 1024000. After looking at the properties closely the FM Receiver needed to receive the default value of sample rates in order to get a better audio and a second TuneFilterDecimate was added to filter the signal after the demodulating was done to set the signal to that the sound card can handle since the computer was used to output the Audio. After doing these modifications FM Receiver sounded just like a normal FM Radio Receiver.

4. RESULTS/DISCUSSION

This section presents the results of the project in detail. The results that are going to be explained in detail are: the written documentation, YouTube video tutorials and the FM Radio Receiver. The results were achieved during phase III, phase IV and phase V of the project.

4.1. FM Radio Receiver

The FM Receiver is main objective of the project. With this FM Receiver you can listen to local FM Stations. The FM Radio Receiver was tested listening to the stations that used 105.3 MHz and 90.7 MHz respectively in Blacksburg, VA. Using the Sandbox REDHAWK IDE to create this FM Receiver shows the potential REDHAWK framework has as a development environment. REDHAWK has features to output the FFT and Amplitude vs. Time graphs from the output ports of each component. Some of these graphs are shown below and explained in this section.

Graph 1 shows the signal spectrum that the RTLTcpSource component receives. This is a FFT graph. It can show the range of signal that the SDR receives from the frequency that is set to. In this case it was set to 105.3 MHz. You can also observe that the signal is very noisy. After the signal spectrum is received, the FM Receiver filters the signal, demodulates the signal, then filters the signal again to set the desired output rate to a rate that the sound card of the computer can handle. Graph 2 shows the signal after the second TuneFilterDecimate component filters the signal. You can also observe that there are fewer samples per time.

After the signal is completely processed, the output of the ConverterFloatToShort component is shown in the Graph 3. The audio is sent to the sound port after it is outputted by the ConverterFloatToShort component. This in an Amplitude vs. Time graph. You can see how the sound varies in the form of sine waves depending on how the
sound is transmitted and if it is shown with more or less samples per time.

4.2. Written documentation

The written documentation was created in order to increase the knowledge of REDHAWK for users so they have guidance during some processes that are essential to use, thus maximizing the benefits of REDHAWK. The finished written documentation can be found in the following link: http://ossie.wireless.vt.edu/download/labs/RH_Tutorials/

The first written document created is the FM Receiver details. This document has a step by step explanation of the steps to follow to create a FM Receiver, assuming that the ConverterFloatToShort has been created before starting the process. This document walks you through the process to start the REDHAWK IDE and add components to the Chalkboard (Part of the Sandbox IDE) while explaining the purpose of each component added. After reading this written documentation, the reader is expected to understand the component's functionality, be able to create an FM Receiver and listen to music using the radio.

The second written documentation created is the Domain Manager issues. This written documentation explains the procedure of how to fix some issues that could appear while using the domain manager by deleting log files and restarting the 'omniNames' and 'omniEvents'. This is not addressed in the REDHAWK Manual 1.8.4 or in the REDHAWK website. After the reader is finished reading this written documentation it will be able to solve an issue that appears when the domain manager fails to connect. This prevents the user to be confused about how to fix the problem and spend time trying to find a solution for this problem.

The third written documentation is about testing components and waveforms in a python session. This gives simple instructions on how to test a component that has been created or any other component that is in the TargetSDR (library of components) or test a waveform using a python session. This type of procedure can add more functionality to a programmer and gives an option of using REDHAWK outside of the Integrated Development Environment. After the reader is finished with this document he will be able to know how to use simple commands to test components and waveforms to control them.

The last written documentation is creating an AMFMPMDemodPortSelector signal processing component. This component is part of the future work, as the FM Receiver can be upgraded to an AM/FM Receiver, the AMFMPMDemodPortSelector is a component that works in real time where you choose which type of demodulation is desired, which helps the user decide the type of signal is going to be used without having to stop the execution to change the signal type. This written tutorial gives you step by step instructions of how to create this signal processing component. After reading this document the reader would have a component that can improve the FM Receiver to a Receiver that has more versatility.

4.3. YouTube video tutorials

There are five YouTube video tutorials uploaded to the YouTube Channel, REDHAWKWaveforms. The titles of the YouTube videos are the following: Build an FM receiver using REDHAWK IDE Sandbox, Create a Simple Waveform (Creating an SCA Waveform Project) using REDHAWK IDE Tutorial, Using the REDHAWK Sandbox to test a component tutorial, Creating a Python Processing Component for REDHAWK Tutorial and Details on Creating a FM Receiver in the REDHAWK IDE.

These video tutorials were created in order to give visual instructions to REDHAWK users to help them learn how to use REDHAWK correctly and in such, help the REDHAWK community to grow. Creating video tutorials is a practical option to share the knowledge. After learning these procedures they can be a step ahead to create complex components and expand REDHAWK’s potential.

The expected result after viewing the video named “Build and FM receiver using REDHAWK IDE Sandbox” is that the user learns how to build an FM Receiver using some components of the TargetSDR. After the user learns how to create a simple FM Receiver the user can move on to create a better version of the Receiver. For the “Create a Simple Waveform video tutorial” the purpose served is to teach the user, step by step, how to build a waveform with basic components and launch that simple waveform in the REDHAWK IDE. After the user finishes watching the “Using the REDHAWK Sandbox to test a component tutorial” video the user will be able to start a python session and test a component connecting it with others and change the properties of the component. “Creating a Python processing component for REDHAWK tutorial” video shows how to create a Python processing component creating a SCA Component Project, adding an input port and an output port to the component, adding a property to the component that can be set during running time or before starting the component and modifying the service function of the component to add functionality to the component. Finally the “Details on Creating a FM Receiver in the REDHAWK IDE” video tutorial is an extended version of the “Build an FM receiver using REDHAWK IDE Sandbox” video tutorial created in order to explain each component
details for users that are not very familiarized with these components.

5. CONCLUSION

REDHAWK can be a powerful tool given that the user has enough knowledge to correctly utilize it. Through this idea, the work that was put into this iteration was to serve future REDHAWK users this knowledge through a thorough documentation, step by step videos of how to create components and utilize the IDE and create components that can simplify the user experience in REDHAWK. With this research, we open the door for future works with REDHAWK IDE while helping develop the REDHAWK community with tutorials and documentation that could help new users to understand the program with much more ease.

6. FUTURE WORK

Future research topics using REDHAWK are available since REDHAWK is still relatively new. Furthering research in this software framework will continue to increase the number of users and to provide feedback for troubleshooting the software. Troubleshooting in the REDHAWK framework was a challenge during this research due to the lack of documentation. Through our work it is expected that users learn to work on REDHAWK and in turn create their own documentation; being recently developed software, it will need further work on stability and its knowledge base.

The project needs to continue and emphasize on troubleshooting, creating customized component and documentation, as this software has great potential. Several suggestions to upgrade this project include upgrading the FM radio receiver to an AM/FM receiver, creating more YouTube tutorials, generating more custom components to add to the components library and creating more advanced waveforms including cognitive controlled waveforms.

7. REFERENCES

