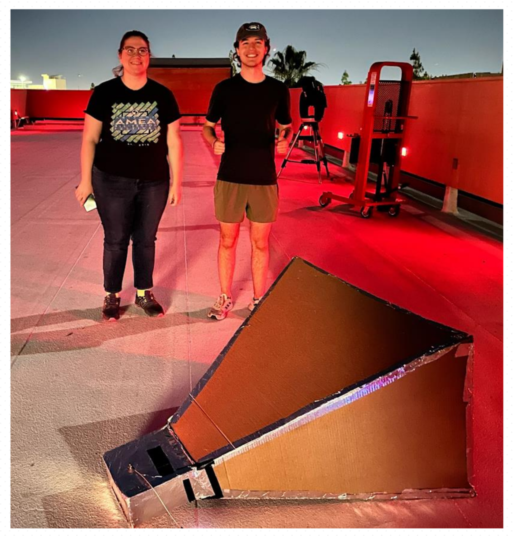
Visualization of Signal Processing for Radio Astronomy

Kaitlyn Ashcroft
School of Earth and Space Exploration

Abstract

The Completely Hackable Amateur Radio Telescope (CHART), is an initiative to create a radio telescope that is both low tech and low cost, making it easily accessible to anyone. The base design is optimized to look at the Milky Way and capture the hydrogen 21 cm line. The CHART project includes a base design that consists of a cardboard telescope and an RTL-SDR module programmed with GNURadio. In a prior iteration the data capture was done using a GnuRadio python script. Here we describe a new version built in GNURadio companion with the aim of improving student understanding of the signal processing steps and encouraging modification. The template in a visual programming language can be easily replicated by new users. Custom blocks simplify the signal flow , and the use of GRC makes the signal processing steps clear and provides a fun and accessible tool set for anyone interested in radio astronomy.



Introduction

The homemade telescope consists of the horn that is pointed at the sky, a low frequency amplifier, a RTL-SDR (or Software Defined Radio), and a raspberry pi. This set up has a program that is very intuitive and easy to use but does not give a great insight into how the signal is being processed by the program. My set up consists of the horn, amplifier, RTL-SDR, and computer. This is for the more inquisitive students who want to see how the signal is being transmitted.

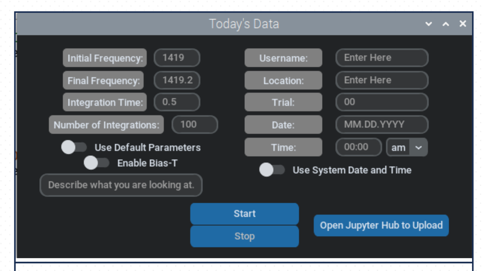


Figure 1: The original program used for collecting data on a raspberry pi.

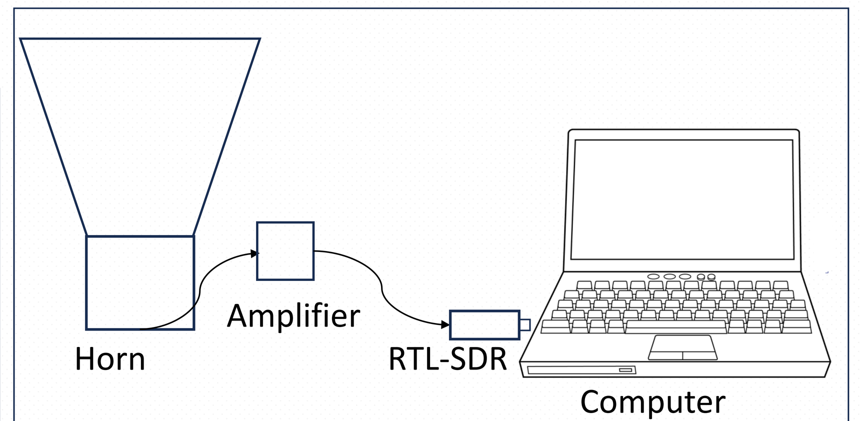
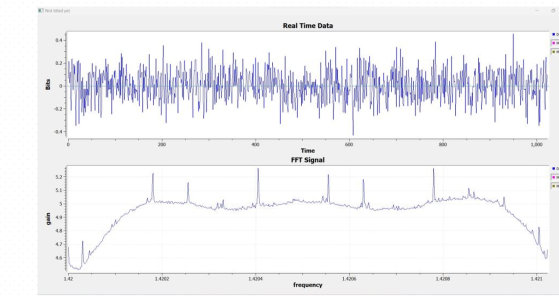


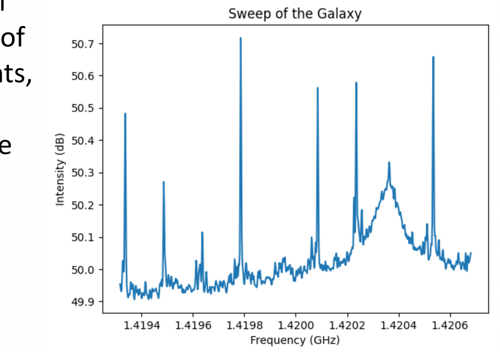
Figure 2: The set up for the computer version of the signal processing

Results

When connected to the telescope we can see the 21 cm. line of the galaxy in a live graph as well as save the data to the computer. The three graphs that we see in figure 3 are the Real Time data, which allows us to determine the state of the data coming in before the FFT. The other two other graphs are post FFT transformation and show us the frequencies the are being picked up by the telescope.



With the GNURadio program we can run an experiment that maps the Milky Way across the sky. The experiment consisted of starting the telescope parallel to the ground, facing where the galaxy is, and taking, for example, 30 seconds of data. Then you start moving the telescope up vertically by 15 degree increments, repeating the same steps of the first parallel experiment. We are then able to save the data to the computer and are then able to process it to look like figure 4.



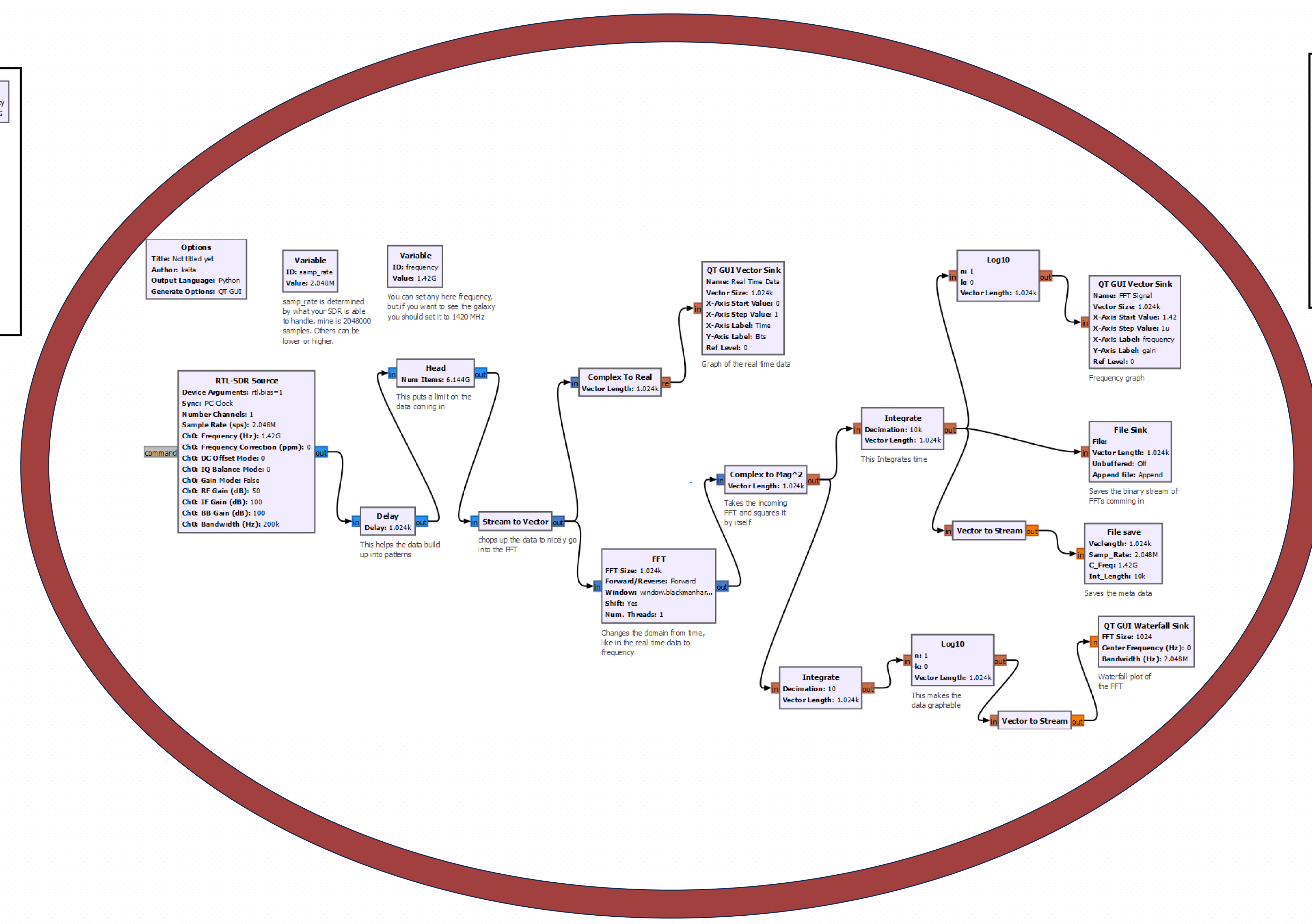
Conclusion

The Completely Hackable Amateur Radio Telescope (CHART), is an initiative to create a radio telescope that is both low tech and low cost, making it easily accessible to anyone. The base design is optimized to look at the Milky Way and capture the hydrogen 21 cm line. The CHART project includes a base design that consists of a cardboard telescope and an RTL-SDR module programmed with GNURadio. In a prior iteration the data capture was done using a GnuRadio python script. Here we describe a new version built in GNURadio companion with the aim of improving student understanding of the signal processing steps and encouraging modification. The template in a visual programming language can be easily replicated by new users. Custom blocks simplify the signal flow , and the use of GRC makes the signal processing steps clear and provides a fun and accessible tool set for anyone interested in radio astronomy.

References

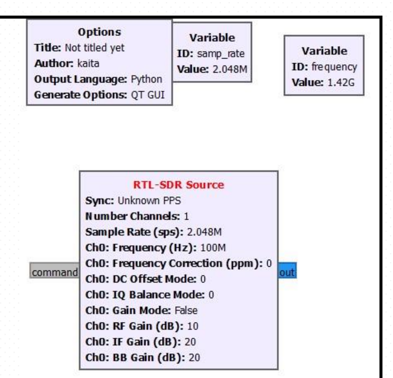
https://astrochart.github.io/
https://www.gnuradio.org/

The Flowgraph



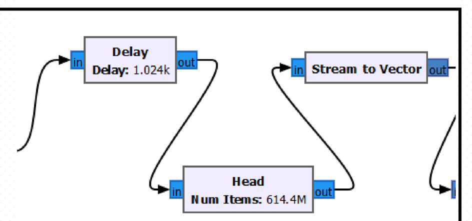
Step 1:

- Variables:
  - Samp_rate: Based on the sample of the SDR device
  - Frequency: the frequency you want to measure at
- RTL-SDR: The block allows the computer to connect to the SDR that is plug into the computer



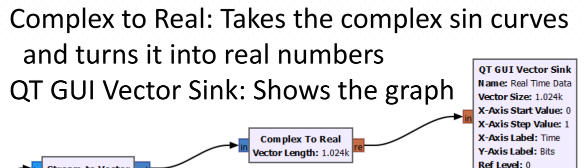
Step 2:

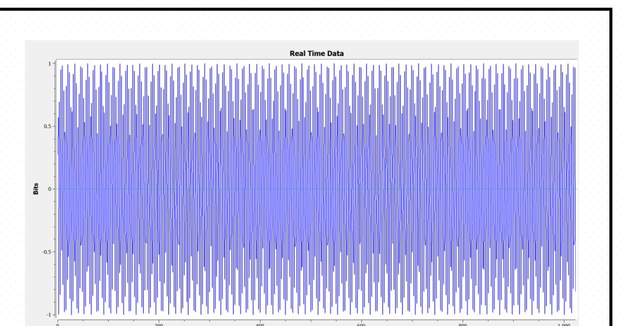
- Delay: holds the data and allows it to build up
- Head: Dictates the amount of data points you want
- Stream to Vector: Takes the stream of sample points and chops it into slices.



Step 3: Graph 1: Real Time Data

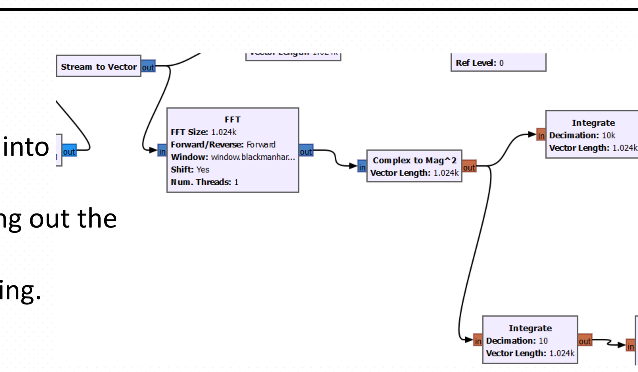
- Complex to Real: Takes the complex sin curves and turns it into real numbers
- QT GUI Vector Sink: Shows the graph





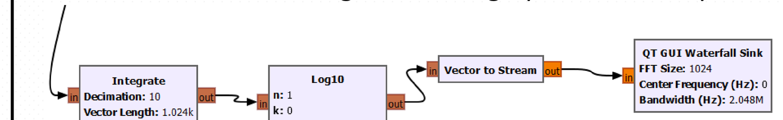
Step 4:

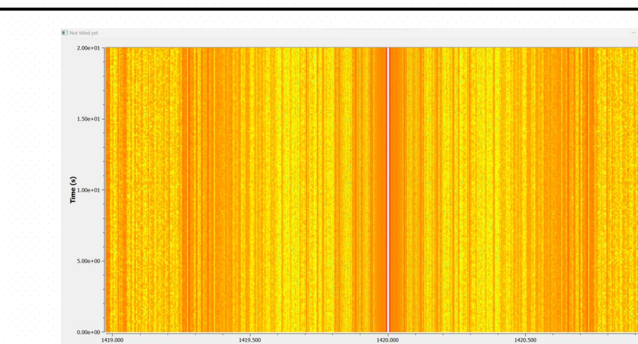
- FFT: Transfers the signal from a time domain into a frequency domain
- Complex to Mag^2: squares the signal to bring out the correct frequencies
- Integrate: integrates over time to do something.



Step 5: Graph 2: Waterfall Plot

- Log10: Transforms the data into making it prettier for graphing
- Vector to Stream: Turns the vectors into a stream so that there is time
- QT GUI Waterfall Sink: generates a graph from the input





Step 6: Graph 3: Frequency Graph

- Log10: Transforms the data into making it prettier for graphing
- QT GUI Vector Sink: generates a graph from the input
- File Sink: saves data to your computer.
- File Save: a custom python embedded block that saves meta data to your computer.