Subject: Event Capture in ADALM Pluto - Extensions to PlutoWeb
Memo: 23, Revision 2
From: Glen Langston,
Date: 2021 February 2

Summary: Adding event capture to an SDR to allow cosmic ray detection. Python and C code are additions to Plutoweb. .

The Science Aficionados can now detect cosmic rays with their own Radio Astronomy Telescope. We document additions to present new blocks that record observations and are also resistant to short duration radio interference, often seen in city locations. Also these new blocks are free to download. The blocks calibrate astronomical observations, so the galaxy can be seen in just a few seconds of observations. The Aficionados will record their observations using ascii data writing block, so they can map the spiral structure of our Milky Way Galaxy. The spectral line data format is described in LightWork Memo 18 and the mapping software, for reducing the observations to images of the sky, is described in LightWork Memo 19.

Background

The Gnuradio Companion (GRC) is provided as a standard part of the Gnuradio installation, so is relatively easy to obtain (see https://wiki.gnuradio.org/index.php/InstallingGR). GRC is fun-to-use visual programming tool. There are many guides online for using different aspects of GRC (see https://wiki.gnuradio.org/index.php/GNURadioCompanion for instance). Here the new radio astronomy blocks are presented (for background on Radio Astronomy see: https://www.cv.nrao.edu/course/astr534/Introradastro.html). Radio Astronomy is also introduced in a Great Course by Dr. Jay Lockman[1].

For a very short, and enjoyable, introduction to GRC take a look our Youtube “Nsf Listens” video (https://bit.ly/2HsFndr), staring Sophie, Evan and myself.

Giant Crab Pulses

The event detection software is tested by observations of Giant Pulses from the Crab Pulsar. These pulses are as bright as 45,000 Jy at 1400 MHz.

We first calculate the sensitivity of a 3-foot diameter horn for detection of pulses, then compute effect of dispersion reducing the peak brightness in a band.

The 3-foot horn has an effective area of 0.5 m^2 . The combination of amplifiers and feed probe has an effective system temperature of 120 K. We use Boltzmann’s constant to convert the system temperature times area into a system equivalent flux density in Janskys. Jansky units are units of power in watt seconds per meter squared. 1 Jy = 10-26 Watts - seconds or 10-26 Watts-Hz/m2.

[1]https://www.thegreatcourses.com/courses/radio-astronomy-observing-the-invisible-universe.html

K/Jy = Ae/2k = 0.5 10 -26/ 2. * 0.25 * / 1.38 x 10-23 J/K = 1.8e+22

K/Jy = 0.0018 K/Jy.

Therefore a 1 Jy source increases the system temperature by 0.0018K, a very small amount. A 1000 Jy source raises the system temperature by 1.8 K.

An extremely bright sources, such as one of the most extreme Crab giant pulses (45,000 Jy), would increase the system temperature by 81 K. This is still small compared with the system temperature for our horn, 120 K. The total brightness temperature, 200K, is 1.4 times the average system temperature.

Dispersion Measure

The interstellar medium contains an ionized plasma. This plasma has the effect of slowing down radio waves as they pass from distant objects to our telescopes. The delay in the signal is not constant, but instead depends on the frequency of observations.

The amount of plasma along the line of sight is related to a measurement we can make with our radio telescopes. The bigger delay the bigger the amount of Dispersion Measure (DM). For the Crab Pulsar the measured DM is 56.5. The total delay in a 6 MHz band at 1.420 GHz is t = 8.3 6. 56.5 /1.42^3 = 982.7 microseconds or 0.0009827 seconds.

The crab pulsar has a period of 0.033089 seconds. So the pulsar is on for about 3% of the time at 1.420 GHz, with a 6 MHz bandwidth.

At 6 MHz bandwidth (12 MHz samples), the total time a giant pulse is present is 12 * 10^6 * 0.0009827 = 11,792 samples. If the band was divided into 128 channels, a pulse would be seen in 92 spectra.

Login to ADALM Pluto
From the Linux command line, you can log into the SDR

ssh root@pluto.local

(Password: *analog*).

Saving the Observations: **Ra_Ascii_Sink (RAS)**

The Aficionado observations are written in simple, ascii, format files. Each of these files includes a summary of the observing setup. Figure 1 shows a plot of the average of a set of ascii files. The file setup information is also written to the notes file, Watch.not, before the observations start. The file is then updated with the time and date of the observations along with the coordinates describing the direction the telescope is pointed.

The post processing software knows how to interpret the observations and average related data. The calibration process is described LightWork Memo 4. The calibration is done by looking at the ground with your telescope (the hot load), and recording a negative elevation. Elevation = -90 indicates your telescope is pointed straight down. A cold load observation is also needed, and the software looks through all your observations for data recorded far from the galactic plane and at high elevation.

Putting the GRC files all together

The GRC system is a great tool for playing with frequencies and time plots. With almost any computer you can create a GRC design that creates interesting tones. (For example, see our video: https://bit.ly/2HsFndr)

The system we're presenting here is designed to be integrated into your own version of GRC, and the steps to include these files are listed at Github. **Figure 3** shows the GRC design used to record astronomical observations. There are many GRC variables in the full design, and this figure excludes variables to simplify the display. The observing setup is completely recorded via GRC "Variable Config" blocks and the Note file.

The observing setup is completely restored each time GRC is restarted. Also note that the data acquisition design can be directly run as a python program, without the GRC interface, further reducing the CPU load.

Conclusion

This memo has described a new GRC design for astronomical observations that allow sensitive observations of the Milky Way. It takes a while to learn how to use GRC and to build your own radio telescope, but taking these steps allows you to see the Universe from your own back yard.

The telescope and software are sensitive enough to see our Galaxy. With 10 minutes of observations, you can start to discover our galaxy. Give these commands a try with your own telescope and collect your own observations of the Milky Way!

Thanks to my family and friends for their support for this project.

Appendix A Median Filter Test Setup

A key feature of the new GRC blocks is that they can be effectively tested through simulated spectra built within GRC. The vector median and average blocks were tested with a GRC design used to create a time sequence at a rate of 1 MHz. This test sequence consisted of three signals, 1) 0.3 MHz complex sine wave with amplitude of 0.15 units. The second test signal 2) was a 0.2 MHz complex sine wave with amplitude of 0.075 units. The third component of the test signal 3) was a gaussian distributed noise source with amplitude of 1.0. The noise source nominated the time sequence and the sin waves were barely visible.

The average blocks are used to reduce the noise, while preserving the wave signals. The design is shown in **Figure A.1**.

Testing New Blocks: **VA** and **VM**

The first new GRC blocks I created are an advance on existing GRC capabilities for observing average spectra in the presence of interference. These blocks are very simple vector processing blocks:

- **Vector Average (VA)**: Average a number of input vectors and output a single vector
- **Vector Median (VM)**: Median average a number of input vectors and output a single vector

Both of these blocks take a single input vector stream and output filtered vectors at a slower rate, after either averaging or median filtering the input vectors. These are both “decimation” blocks. The decimation factor reduces the computer load after the block. A series of these blocks further reduces the computer load.

The **VM** block is implemented by waiting for GRC to gather **N** vectors (usually 4) and then finding the min and max of each of **N** vectors for each of the many channels (usually 1024 or 2048 channels), then subtracting the min and max from the average of all **N** values. The VM block is exactly the mathematical median for a decimation factor of 3 or 4. Decimation factors of 1 or 2 are not allowed. For decimation factors greater than 4, the middle values are still averaged after removing the high and low values. For large N, the median calculation approaches the same result as the VA average block.

The VA block allows averaging with decimation factors between 1 and 15. The GRC scheduler does not allow capturing more than 15 vectors before calling the block.

Figure A.2 shows a comparison of outputs of the VA and VM blocks. The points to note in **Figure A.2** are **1)** both the median and average blocks significantly reduce the noise in the spectra, compared with a single input spectrum. Note **2)** that the median shows about the same noise level as the average block. The median block is much more resistant to short-term interference, with only a small noise penalty.

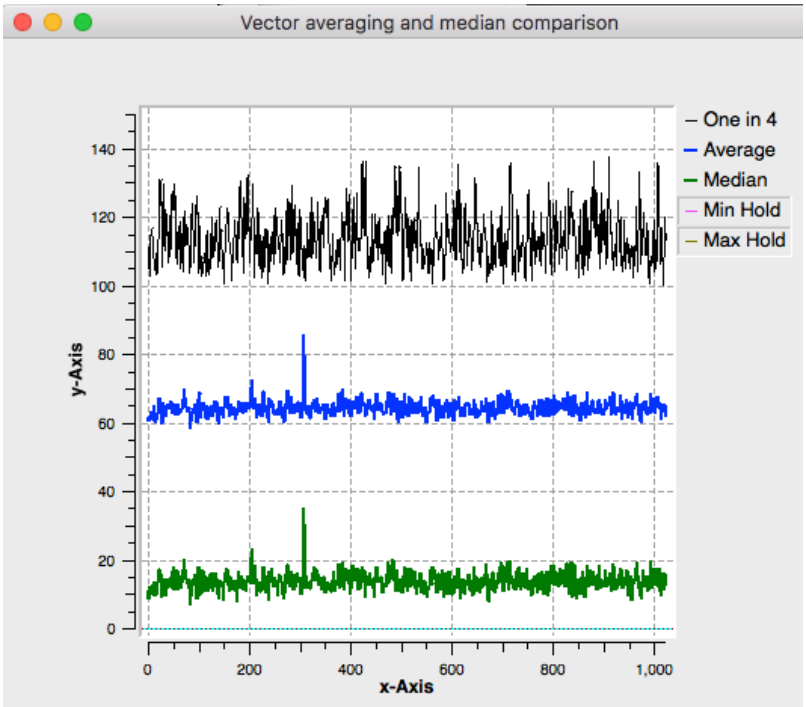
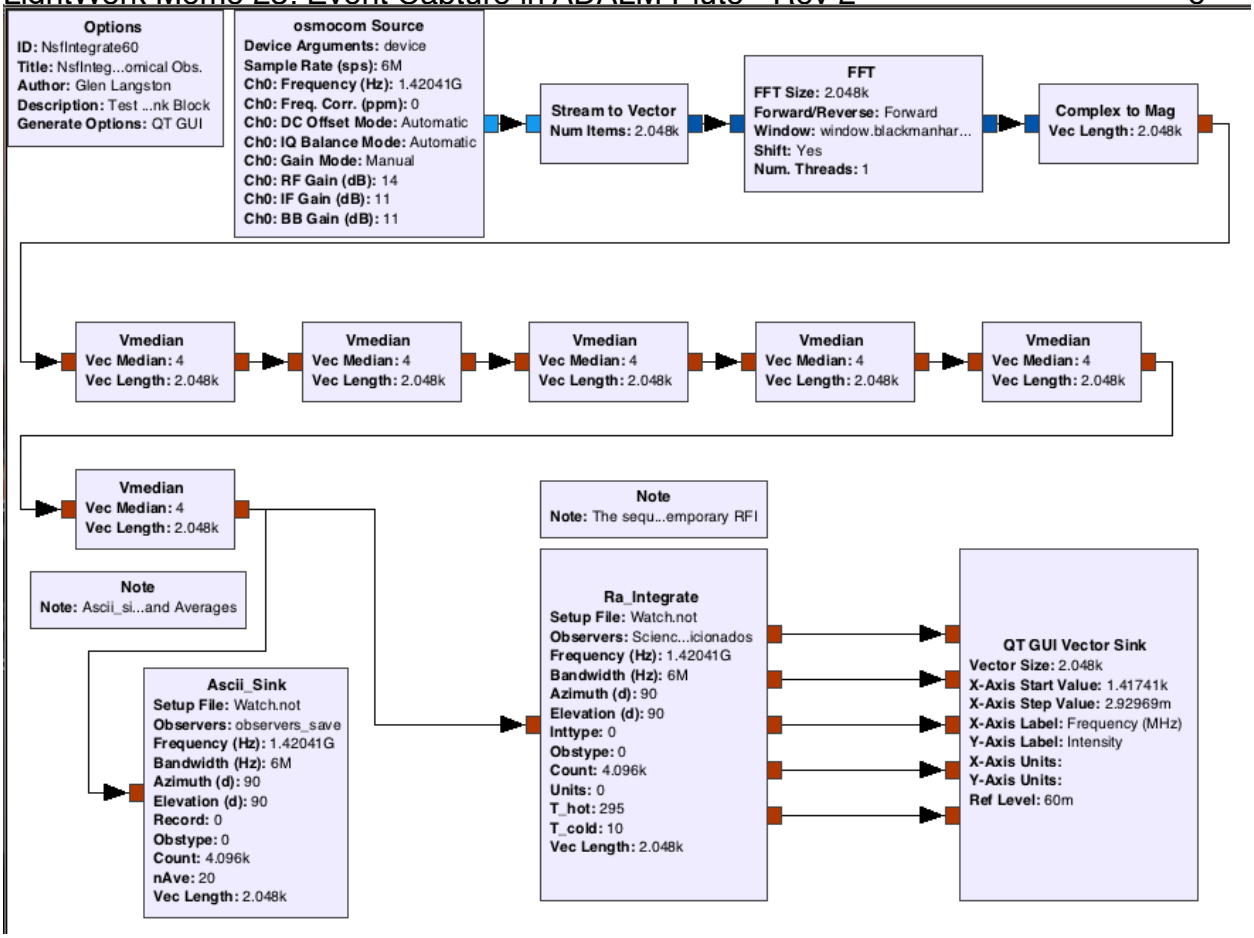


Figure A.2: Comparison of Vector Average and Vector Median blocks relative to an input model spectrum. *The bottom (green) curve shows the Vector Median output after the test spectra pass through two median blocks, each with decimation factor of 4. The middle plot (blue) shows the average of spectra passing through two vector average blocks each with a decimation factor of 4. The top (black) plot shows a single input spectrum before passing through the vector processing blocks. counts. The curves are offset to allow comparison.*

Appendix B: NsfIntegrate.grc Installation Guide

This section is underdevelopment, as I prepare to place all the components in Github.


Options
ID: NsfIntegrate60
Title: NsfInteg...omical Obs.
Author: Glen Langston
Description: Test ...nk Block
Generate Options: QT GUI
osmocom Source
Device Arguments: device
Sample Rate (sps): 6M
Ch0: Frequency (Hz): 1.42041G
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Automatic
Ch0: IQ Balance Mode: Automatic
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 14
Ch0: IF Gain (dB): 11
Ch0: BB Gain (dB): 11
Stream to Vector
Num Items: 2.048k
FFT
FFT Size: 2.048k
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1
Complex to Mag
Vec Length: 2.048k
Vmedian
Vec Median: 4
Vec Length: 2.048k
Vmedian
Vec Median: 4
Vec Length: 2.048k
Vmedian
Vec Median: 4
Vec Length: 2.048k
Vmedian
Vec Median: 4
Vec Length: 2.048k
Vmedian
Vec Median: 4
Vec Length: 2.048k
Vmedian
Vec Median: 4
Vec Length: 2.048k
Note
Note: The sequ...emporary RFI
Note
Note: Ascii_si...and Averages
Ascii_Sink
Setup File: Watch.not
Observers: observers_save
Frequency (Hz): 1.42041G
Bandwidth (Hz): 6M
Azimuth (d): 90
Elevation (d): 90
Record: 0
Obstype: 0
Count: 4.096k
nAve: 20
Vec Length: 2.048k
Ra_Integrate
Setup File: Watch.not
Observers: Scienc...icionados
Frequency (Hz): 1.42041G
Bandwidth (Hz): 6M
Azimuth (d): 90
Elevation (d): 90
Inttype: 0
Obstype: 0
Count: 4.096k
Units: 0
T_hot: 295
T_cold: 10
Vec Length: 2.048k
QT GUI Vector Sink
Vector Size: 2.048k
X-Axis Start Value: 1.41741k
X-Axis Step Value: 2.92969m
X-Axis Label: Frequency (MHz)
Y-Axis Label: Intensity
X-Axis Units:
Y-Axis Units:
Ref Level: 60m

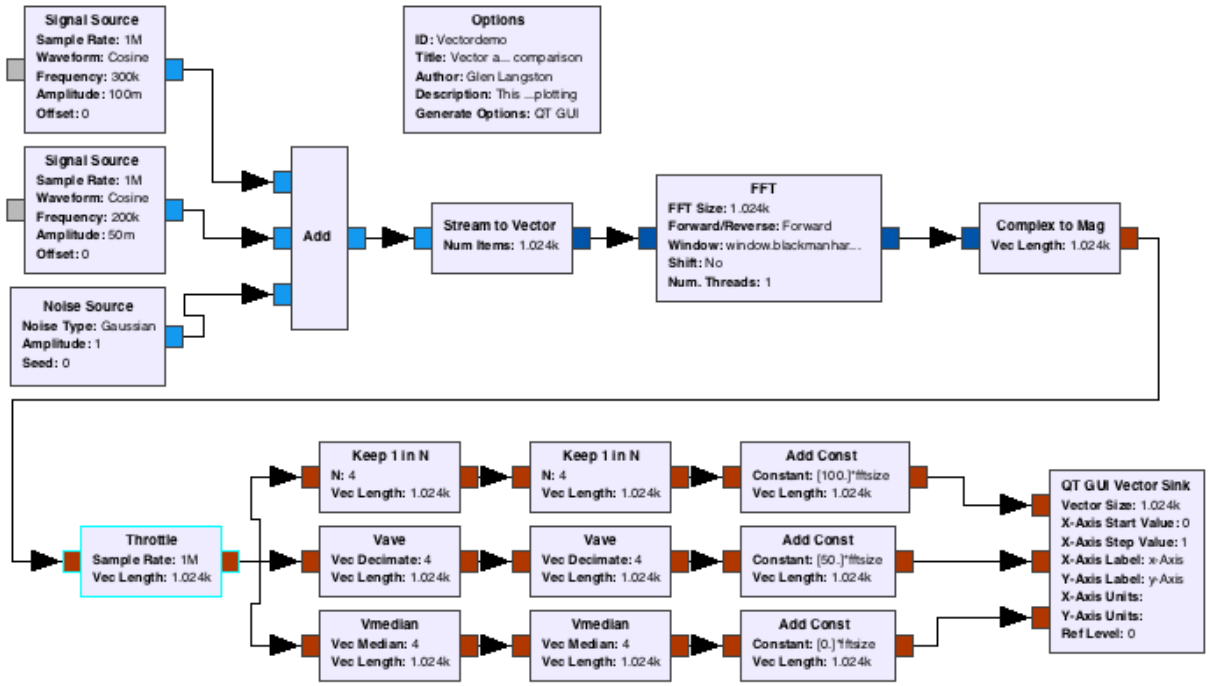


Figure A.1: GRC design Vectordemo.grc. *This figure shows a screen capture of the GRC design used to test both the vector average and median blocks. The design has a throttle block to control the computer load. The output passes through three sets of processing blocks. The top set just keeps one of the 16 spectra passing through. The middle set averages the 16 spectra passing through. The bottom block median filters the 16 spectra passing through. The QT GUI Vector Sink plots the spectra. A different constant was added to each of the three spectra so the spectra could be compared on a single plot.*