

QRAAT backend software manual

Todd Borrowman, Christopher Patton
John Muir Institute for the Environment
November 2012

This document describes how to use the backend software originally developed by Todd Borrowman for the QRAAT system. The software handles the immediate signal processing of the RMG receiver as well as visualization and configuration for the target transmitters. We refer to these tools collectively as the RMG module.

Obtaining the software

The RMG module can be downloaded via git at [[link not yet available](#)]. The package includes C++ sources for the processing block – as it is known in Gnu Radio terminology – that performs the low level signal processing (located in the directories `lib/` and `include/`). Also included in this package are the Python wrappers for the processing block (`swig/`) and an implementation of the data flow graph from the RMG module to the fully processed signal (`python/`). Finally, the runnable programs are located in `apps/`.

Installation

The software package includes scripts for downloading and building the Universal Hardware Driver (UHD) from Ettus Research¹ (driver for the RMG receiver), the Gnu Radio system, and building our module from source. As of this writing, these scripts only work for the Fedora and Ubuntu GNU/Linux distributions. (However, since these software packages use `cmake`, installation on Windows, Mac, or other Linux distributions should be relatively straight forward.) On these platforms, invoking

```
$ ./build-gnuradio2 -v all
```

will build Gnu Radio as well as the UHD driver. On most systems, this takes about an hour. To take advantage of a multicore system to speed up the build process, type

```
$ ./build-gnuradio -v -ja all
```

Installation of the RMG module is similarly simple and takes less than a minute:

```
$ ./install-rmg -v install
```

Lastly, it is a good idea to add your user account to the group with read/write permissions for the serial port. Otherwise you will have to use the following programs with `sudo` which will produce data files that belong to root. Simply add your user name to the 'dialout' group:

```
$ sudo adduser <username> dialout
```

and restart your computer.

Usage

We've developed the runnable programs to support standard desktop systems as well as headless installations (no monitor) that are power limited. The four main programs are:

`4channel_spectrum` – provides visualization of the four channel rapid multichannel goniometer,

1 See: http://files.ettus.com/uhd_docs/manual/html/ for details.

2 We distribute the build script written and maintained by Marcus Leech. The most recent version can be found at <http://www.sbrac.org/files/build-gnuradio>.

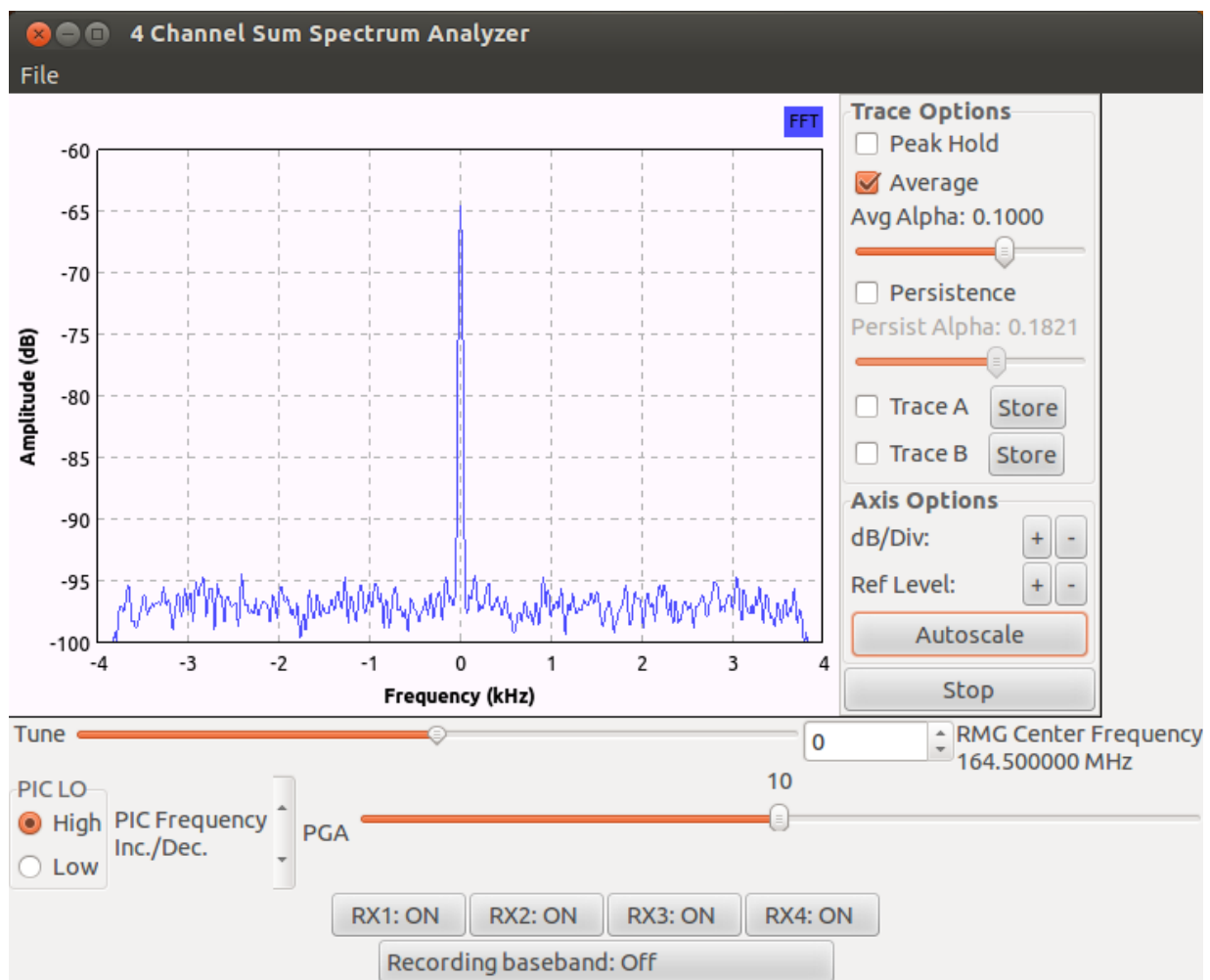
tx_setup – set up transmitters and calculate the tuning parameters for the module, and

run_rmg – invokes the signal processing graph and outputs pulses in files.

active_rmg – runs run_rmg, but also displays activity information.

Four Channel Sum Spectrum Analyzer

Simply type this into the terminal to open the window. Note that it may be necessary to run the program with administrator privileges (ie. “\$ sudo 4channel_spectrum”)



The top section controls the audio filter. The radio buttons select the upper side band (USB) or the lower side band (LSB). The sliders select the cutoff frequencies (in Hz) for the audio filter. The central graph displays the sum of the absolute value of the activated RMG receiver channels. The graph is an 8 kHz wide signal mixed down to baseband by the frequency displayed on the frequency

slider (see below). The yaxis is displayed in dB. The controls for the graph are to the right.
Note: In the current configuration the RMG has a large DC bias. This is inserted after the A/D and does not effect the dynamic range of the device.

Tuning:

Peak Hold toggles on/off the green plot of the peak values of the graph .
Average toggles on/off averaging of the spectrum .
Avg Alpha the time constant used in the averaging, larger → less values used in average .
dB/div radio buttons determine the size of the y-axis divisions .
Ref Level +/- determines the minimum y-axis value .
Autoscale sets the graphs division size and reference automatically .
Run/Stop Starts or Stops the graph display .
Tune slider and spin control – determines the center frequency displayed in the graph .
PGA slider determines the value of the PGA (Programmable Gain Amplifier, 020 dB) .
Vol slider determines the volume of the audio output .
RX# toggle buttons – toggles on/off the individual channels (antennas) of the RMG in the graph, audio and recording .
Recording button toggles on/off the recording, saved to a file which is named based on the time the recording starts (YYYYMMDDhhmmss.tdat) and is displayed on the button when recording is on .

Headless systems (4channel_spectrum_server/4channel_spectrum_client):

For embedded installations, we provide a visualization scheme that utilizes a network connection between the headless computer that is phsically connected to the RMG and a standard desktop or laptop with graphical user interface capabilities. Essentially, we divide the four channel spectrum analyzer into server and client components. On the embedded system, invoke the server with its IP address, for example:

```
# 4channel_spectrum_server -n 192.168.1.1
```

Running without specifying an IP address defaults to the system's loopback interface (127.0.0.1, or 'localhost' in Unix terms). On your desktop or laptop, you must specify the server's IP address:

```
$ 4channel_spectrum_client -n 192.168.1.1
```

Again, this defaults to the loopback interface. This establishes a TCP connection between your desktop or laptop and the embedded system for passing tuning parameters from the client to the server.. Data transmission commences shortly after on a parallel UDP socket³. The spectrum analyzer window will appear on your computer with virtually the same functionality.

3 Parameter passing and data transmission occur on ports 50006 and 50007 respectively.

Transmitter Setup

tx_setup provides a graphical user interface (GUI) to input transmitter information and generate a parameter file which controls the RMG receiver. *[NOTE: this program no longer does tuning calculation. It is simply an interface for creating .csv files.]*

Front-End Frequency Parameters
PreAmp - Minimum Frequency: 162.000000 MHz, Maximum Frequency: 167.000000 MHz
First IF Stage - Center Frequency: 70.000000 MHz, Bandwidth: 500.000000 kHz
Second LO Frequency: 80.700000 MHz
Second IF Stage - Center Frequency: 10.700000 MHz, Bandwidth: 250.000000 kHz
PLL/VCO Tuning Range - Min: 218.500000 MHz, Max: 248.000000 MHz, Step: 100.000000 kHz, Offset: 0.000000 kHz
Number of Baseband Channels: 32

Directory for .det files ...

Select	Name	Frequency in MHz	Type	Pulse Width in ms	Rise Trigger	Fall Trigger	Filter Alpha
<input checked="" type="checkbox"/>	mouse1	164.500000	Pulse	20.0	3.00	2.00	0.010
<input checked="" type="checkbox"/>	noisy	164.700000	Pulse	20.0	1.01	1.00	0.010
<input checked="" type="checkbox"/>	robin0	164.400000	Pulse	10.0	3.00	2.00	0.010

At the top of window is a list of front-end frequency parameters. These depend on the hardware being used. Currently these parameters can only be set within the code, however the user should not need to change these in the course of entering the transmitter list. See the documentation for `rmg_setup.py` for more information.

The text box is used to name the directory the user wants the generated '.det' files to be saved in. The button to the right of the box, labeled with ellipsis (...), brings up a dialog box with which the user may choose a directory. If the directory in the text box does not exist, the directory (and associated path) will be created when the RMG is run.

The spreadsheet in the center of the window contains information on the transmitters to be tracked. The name can be any sequence of alphanumeric characters and/or symbols except for \ : * ? | < > /. This name is used within the file name of any '.det' or '.tdat' files created for this transmitter. The frequency of the transmitter should be entered in MHz. The type of transmitter can be pulse, continuous, or other. The pulse type uses the pulse detector to filter out the pulses from the received signal and saves them as '.det' files. Both the continuous and other types produce a '.tdat' file which records the baseband signal from the polyphase filter in the software-defined radio. For pulse type transmitters a pulse width should be included in milliseconds (ms). This information is used in the pulse detector to determine the integration period. For continuous and other types this field is not used. The buttons at the bottom of the window have the following functions:

<i>Check All</i>	Selects all of the transmitters in the spreadsheet.
<i>Uncheck All</i>	Deselects all of the transmitters in the spreadsheet.
<i>Invert Selection</i>	Selects all of the transmitters that are not selected and deselects all of them which currently are selected.

<i>Delete Selected</i>	Removes the selected transmitters from the spreadsheet.
<i>Add New Transmitter</i>	Adds a new transmitter to the spreadsheet with default values
<i>Calculate Tuning</i>	Calculates the tuning parameters needed to record only the selected transmitters. These parameters are saved to a '.dfc' file and a dialog box pops up for the user to name the file and save it to the desired directory. This is the file the RMG operation program uses to receive and record the transmitters. <i>[NOTE: This functionality has been deprecated. The tuning is now calculated when run_rmg is invoked]</i>

The menu has the following items:

File →

- Load Loads a '.csv' into the GUI. This file also saves the front-end parameters listed at the top of the window so be careful when using files that were setup for different systems.
- Save Saves the current GUI information to a '.csv' file.
- Exit Closes the GUI

Help →

- About Generic information about the program.

Headless systems:

Because we have moved the tuning calculation from the transmitter setup to the main program, there is no longer a need for a text-based equivalent of this program. It is simple enough to create your own '.csv' file in your favorite editor. The csv has the following format (no whitespace permitted):

```
use,name,freq,type,pulse_width,rise_trigger,fall_trigger,filter_alpha
Yes,mouse1,164.5,Pulse,20,3.0,2.0,0.01
No,noisy,164.7,Pulse,20,1.010,1.005,0.01
Yes,robin0,164.4,Pulse,10,3.0,2.0,0.01
```

Running the RMG module

To run the signal processing graph, we specify the transmitter configuration file generated by tx_setup:

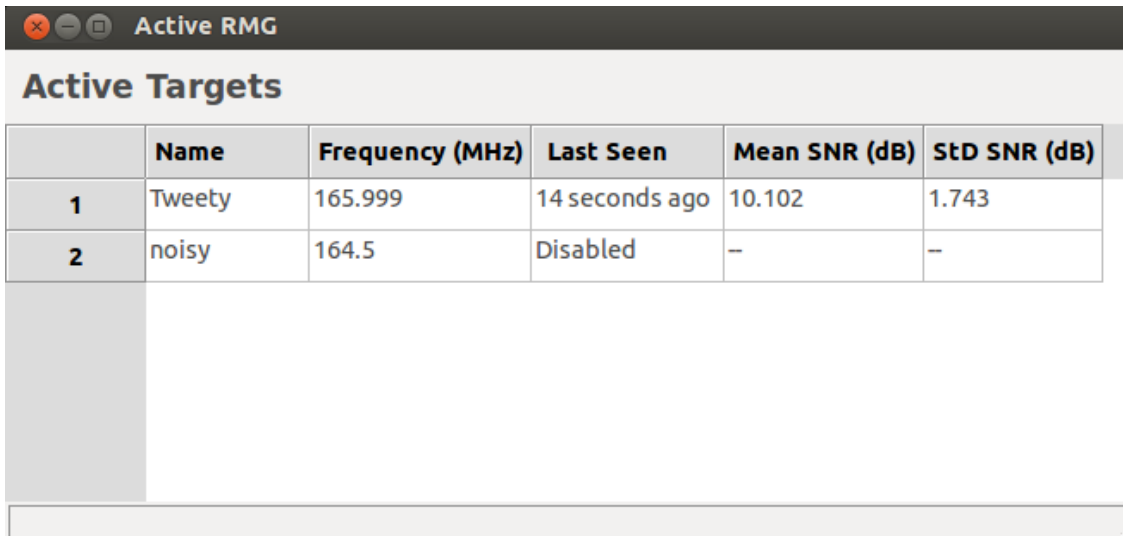
```
$ run_rmg -f my_tx.csv -d det_files -b 1
```

If no file is provided, the program defaults to 'tx.csv' in the working directory. The option -b determines the number of basebands that will be configured. The maximum number, and the default for this program, is 32 basebands. If 1 is specified, this disables the polyphase filter, which adds too much load on certain power limited systems.

Active RMG

In addition to this text-based program, we've developed an application for visualizing the activity levels of target transmitters. This is run in the same way as run_rmg, but has a different name:

```
$ active_rmg -f my_tx.csv -d det_files
```



The screenshot shows a window titled "Active RMG" with a table of active targets. The table has six columns: an index column, Name, Frequency (MHz), Last Seen, Mean SNR (dB), and StD SNR (dB). There are two rows of data. The first row shows a target named "Tweety" at 165.999 MHz, last seen 14 seconds ago, with a mean SNR of 10.102 dB and a standard deviation of 1.743 dB. The second row shows a target named "noisy" at 164.5 MHz, which is disabled, with a mean SNR of - and a standard deviation of -.

	Name	Frequency (MHz)	Last Seen	Mean SNR (dB)	StD SNR (dB)
1	Tweety	165.999	14 seconds ago	10.102	1.743
2	noisy	164.5	Disabled	—	—

The window displays the time when each transmitter was last detected as well as the mean and standard deviation of the SNR value. This serves as a proxy for the activity of the target animals.