Documentation Re-Org and CGRAN Update

Marc Lichtman

GNU Radio Conference 2018

Outline

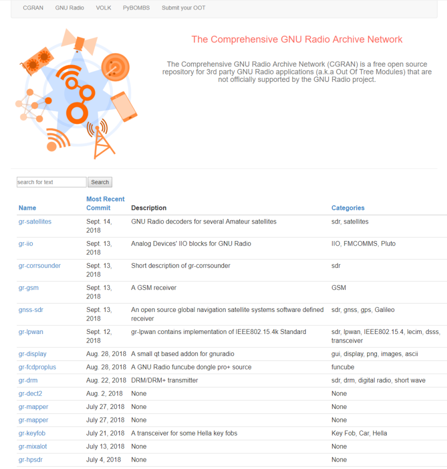
1. CGRAN Update
2. Documentation “re-org”

What is CGRAN?

- The Comprehensive GNU Radio Archive Network
- Repository for 3rd party GNU Radio applications that are not officially supported by the GNU Radio project
- (a.k.a Out Of Tree Modules)
- E.g. OOTs: gr-fosphor, gr-gsm, gr-ieee802-11, gr-ofdm

The New CGRAN

- New version of CGRAN is live at cgran.org
- It’s actually up to date now
- Includes date of most recent commit
- Search bar feature
- New logo!

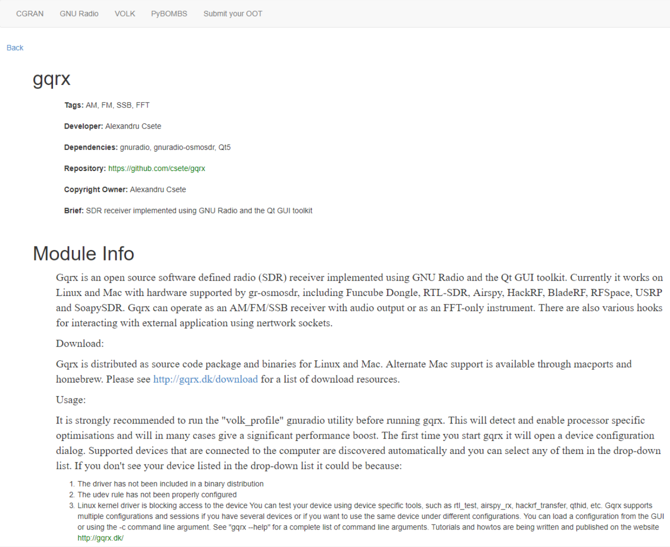


The Comprehensive GNU Radio Archive Network

The Comprehensive GNU Radio Archive Network (CGRAN) is a free open source repository for 3rd party GNU Radio applications (a.k.a Out Of Tree Modules) that are not officially supported by the GNU Radio project.

Name	Most Recent Commit	Description	Categories
gr-satellites	Sept. 14, 2018	GNU Radio decoders for several Amateur satellites	sdr, satellites
gr-iio	Sept. 13, 2018	Analog Devices' IIO blocks for GNU Radio	IIO, FMCOMMS, Pluto
gr-corrsounder	Sept. 13, 2018	Short description of gr-corrsounder	sdr
gr-gsm	Sept. 13, 2018	A GSM receiver	GSM
gnss-sdr	Sept. 13, 2018	An open source global navigation satellite systems software defined receiver	sdr, gnss, gps, Galileo
gr-lpwan	Sept. 12, 2018	gr-lpwan contains implementation of IEEE802.15.4k Standard	sdr, lpwan, IEEE802.15.4, lecim, dsss, transceiver
gr-display	Aug. 28, 2018	A small qt based addon for gnuradio	gui, display, png, images, ascii
gr-fcdproplus	Aug. 28, 2018	A GNU Radio funcube dongle pro+ source	funcube
gr-drm	Aug. 22, 2018	DRM/DRM+ transmitter	sdr, drm, digital radio, short wave
gr-dect2	Aug. 2, 2018	None	None
gr-mapper	July 27, 2018	None	None
gr-mapper	July 27, 2018	None	None
gr-keyfob	July 21, 2018	A transceiver for some Hella key fobs	Key Fob, Car, Hella
gr-mixalot	July 13, 2018	None	None
gr-hpsdr	July 4, 2018	None	None

Example OOT Listing



gqrx

Tags: AM, FM, SSB, FFT

Developer: Alexandru Csete

Dependencies: gnuradio, gnuradio-osmosdr, Qt5

Repository: https://github.com/csete/gqrx

Copyright Owner: Alexandru Csete

Brief: SDR receiver implemented using GNU Radio and the Qt GUI toolkit

Module Info

Gqrx is an open source software defined radio (SDR) receiver implemented using GNU Radio and the Qt GUI toolkit. Currently it works on Linux and Mac with hardware supported by gr-osmosdr, including Funcube Dongle, RTL-SDR, Airspy, HackRF, BladeRF, RFSpace, USRP and SoapySDR. Gqrx can operate as an AM/FM/SSB receiver with audio output or as an FFT-only instrument. There are also various hooks for interacting with external application using nertwork sockets.

Download:

Gqrx is distributed as source code package and binaries for Linux and Mac. Alternate Mac support is available through macports and homebrew. Please see http://gqrx.dk/download for a list of download resources.

Usage:

It is strongly recommended to run the "volk_profile" gnuradio utility before running gqrx. This will detect and enable processor specific optimisations and will in many cases give a significant performance boost. The first time you start gqrx it will open a device configuration dialog. Supported devices that are connected to the computer are discovered automatically and you can select any of them in the drop-down list. If you don't see your device listed in the drop-down list it could be because:

1. The driver has not been included in a binary distribution
2. The udev rule has not been properly configured
3. Linux kernel driver is blocking access to the device You can test your device using device specific tools, such as rtl_test, airspy_rx, hackrf_transfer, qthid, etc. Gqrx supports multiple configurations and sessions if you have several devices or if you want to use the same device under different configurations. You can load a configuration from the GUI or using the -c command line argument. See "gqrx --help" for a complete list of command line arguments. Tutorials and howtos are being written and published on the website http://gqrx.dk/

Current Limitations

Still has issues:

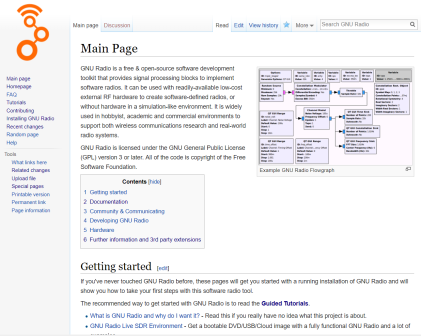
- Only shows OOTs that have a PyBOMBS recipe (so we are likely missing some valuable OOTs)
- Gets its info by parsing OOT's MANIFEST files, many of which don't exist
- The parser has trouble with OOTs not hosted on github
- No way to know if a PyBOMBS recipe is actually an OOT

Documentation

Documentation

Current state of GNU Radio's Documentation:

- Doxygen "C++ Manual/API"
- Sphinx "Python Manual/API"
- Wiki
- Tom's blog & other random stuff



Main Page

GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems.

GNU Radio is licensed under the GNU General Public License (GPL) version 3 or later. All of the code is copyright of the Free Software Foundation.

Example GNU Radio Flowgraph

Contents [hide]

1 Getting started
2 Documentation
3 Community & Communicating
4 Developing GNU Radio
5 Hardware
6 Further information and 3rd party extensions

Getting started [edit]

If you've never touched GNU Radio before, these pages will get you started with a running installation of GNU Radio and will show you how to take your first steps with this software radio tool.

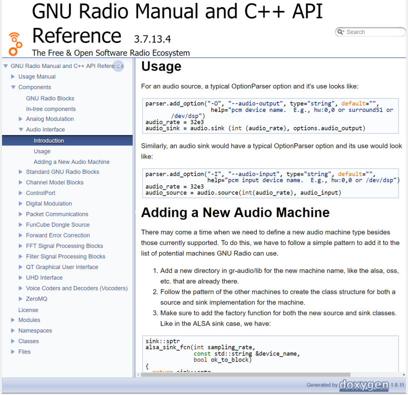
The recommended way to get started with GNU Radio is to read the **Guided Tutorials**.

- What is GNU Radio and why do I want it? - Read this if you really have no idea what this project is about.
- GNU Radio Live SDR Environment - Get a bootable DVD/USB/Cloud image with a fully functional GNU Radio and a lot of

Doxygen Manual

Doxygen Contains:

- Usage Manual
- Block specific docs
- Snippets of other stuff here and there
- Lots of auto-generated descriptions



GNU Radio Manual and C++ API Reference 3.7.13.4

The Free & Open Software Radio Ecosystem

Usage

For an audio source, a typical OptionParser option and it's use looks like:

```
parser.add_option("-O", "--audio-output", type="string", default="",
                  help="pcm device name.  E.g., hw:0,0 or surround51 or
      /dev/dsp")
audio_rate = 32e3
audio_sink = audio.sink (int (audio_rate), options.audio_output)
```

Similarly, an audio sink would have a typical OptionParser option and its use would look like:

```
parser.add_option("-I", "--audio-input", type="string", default="",
                  help="pcm input device name.  E.g., hw:0,0 or /dev/dsp")
audio_rate = 32e3
audio_source = audio.source(int(audio_rate), audio_input)
```

Adding a New Audio Machine

There may come a time when we need to define a new audio machine type besides those currently supported. To do this, we have to follow a simple pattern to add it to the list of potential machines GNU Radio can use.

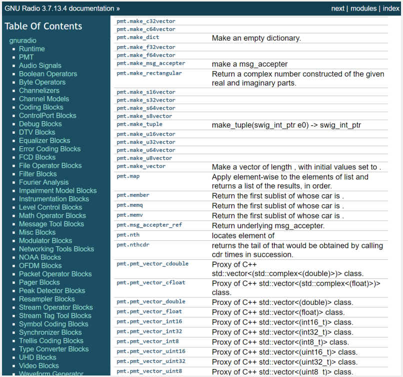
1. Add a new directory in gr-audio/lib for the new machine name, like the alsa, oss, etc. that are already there.
2. Follow the pattern of the other machines to create the class structure for both a source and sink implementation for the machine.
3. Make sure to add the factory function for both the new source and sink classes. Like in the ALSA sink case, we have:

```
sink::sptr
alsa_sink_fcn(int sampling_rate,
              const std::string &device_name,
              bool ok_to_block)
{
  return sink::sptr
```

Sphinx Manual

Sphinx Contains:

- Auto-generated Sphinx stuff
- That's it



GNU Radio 3.7.13.4 documentation » next | modules | index

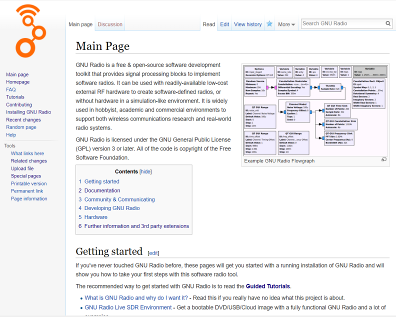
Table Of Contents

`pmt.make_c32vector`	
`pmt.make_c64vector`	
`pmt.make_dict`	Make an empty dictionary.
`pmt.make_f32vector`	
`pmt.make_f64vector`	
`pmt.make_msg_accepter`	make a msg_accepter
`pmt.make_rectangular`	Return a complex number constructed of the given real and imaginary parts.
`pmt.make_s16vector`	
`pmt.make_s32vector`	
`pmt.make_s64vector`	
`pmt.make_s8vector`	
`pmt.make_tuple`	make_tuple(swig_int_ptr e0) -> swig_int_ptr
`pmt.make_u16vector`	
`pmt.make_u32vector`	
`pmt.make_u64vector`	
`pmt.make_u8vector`	
`pmt.make_vector`	Make a vector of length , with initial values set to .
`pmt.map`	Apply element-wise to the elements of list and returns a list of the results, in order.
`pmt.member`	Return the first sublist of whose car is .
`pmt.memq`	Return the first sublist of whose car is .
`pmt.memv`	Return the first sublist of whose car is .
`pmt.msg_accepter_ref`	Return underlying msg_accepter.
`pmt.nth`	locates element of
`pmt.nthcdr`	returns the tail of that would be obtained by calling cdr times in succession.
`pmt.pmt_vector_cdouble`	Proxy of C++ std::vector<(std::complex<(double)>)> class.
`pmt.pmt_vector_cfloat`	Proxy of C++ std::vector<(std::complex<(float)>)> class.
`pmt.pmt_vector_double`	Proxy of C++ std::vector<(double)> class.
`pmt.pmt_vector_float`	Proxy of C++ std::vector<(float)> class.
`pmt.pmt_vector_int16`	Proxy of C++ std::vector<(int16_t)> class.
`pmt.pmt_vector_int32`	Proxy of C++ std::vector<(int32_t)> class.
`pmt.pmt_vector_int8`	Proxy of C++ std::vector<(int8_t)> class.
`pmt.pmt_vector_uint16`	Proxy of C++ std::vector<(uint16_t)> class.
`pmt.pmt_vector_uint32`	Proxy of C++ std::vector<(uint32_t)> class.
`pmt.pmt_vector_uint8`	Proxy of C++ std::vector<(uint8_t)> class.

Wiki

Wiki Contains:

- Installation guides
- Getting started guides
- Many tutorials
- Community info
- Hardware info
- Misc articles of various quality



Main Page

GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems.

GNU Radio is licensed under the GNU General Public License (GPL) version 3 or later. All of the code is copyright of the Free Software Foundation.

Example GNU Radio Flowgraph

Contents [hide]

1 Getting started
2 Documentation
3 Community & Communicating
4 Developing GNU Radio
5 Hardware
6 Further information and 3rd party extensions

Getting started [edit]

If you've never touched GNU Radio before, these pages will get you started with a running installation of GNU Radio and will show you how to take your first steps with this software radio tool.

The recommended way to get started with GNU Radio is to read the **Guided Tutorials**.

- What is GNU Radio and why do I want it? - Read this if you really have no idea what this project is about.
- GNU Radio Live SDR Environment - Get a bootable DVD/USB/Cloud image with a fully functional GNU Radio and a lot of

Current Issues

Biggest issues (besides blocks with missing docs):

1. Information is spread out among many locations
2. People new to GR *very* often have issues finding stuff
3. Contributions to the doxygen content might be too intimidating for some, or require too many steps
4. Python (Sphinx) manual is primarily auto-generated content

Possible Solutions

We need to define what documentation goes where, so people know where to look

- One solution is to make the Doxygen manual only contain block/class/function specific docs
- If it's not block/class/function specific, it's in the wiki
- Usage manual is the main thing that would be removed (wiki usage manual could be exported and available offline)
- There are also some descriptions in the top-level components sections

Possible Solutions

Sphinx related:

- Figure out how people really use the Python API
- Find a better way to present that information
- Do we just need a list of the Python-version of everything? Categorized in a logical manner

Possible Solutions

Ease-of-contributions:

- Need better (or more easily found) instructions for how to modify Doxygen
- Github's built in text editor and commit system
- Perhaps a more automatic way of submitting changes, e.g. with a form
- Use wiki for anything appropriate

Efforts to Date

So far I have:

- Converted usage manual doxygen to wiki format, although there are still touch-ups needed
- Organized some wiki content, still plenty left
- Wrote a script to export certain wiki pages (future usage manual) so they can be included in the main repo and available offline

Input

Questions for the audience:

1. Would it be a problem if the Usage Manual is (only) on the Wiki?
2. I’m curious who uses the Python API and in what way
3. What would make people more likely to contribute doc-related changes?

Conclusion

Conclusion:

- Use CGRAN! Post on Slack if you have issues with it
- Reach out to me if you want to help with the docs re-org