

# 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](http://cgran.org)
- It's actually up to date now
- Includes date of most recent commit
- Search bar feature
- New logo!

CGRAN GNU Radio VOLK PyFIMMS Submit your OOT



**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.

search for text  Search

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

Navigation icons: back, forward, search, etc.

# Example OOT Listing

[CGRAN](#)
[GNU Radio](#)
[VOLK](#)
[PyBOMBS](#)
[Submit your OOT](#)

[Back](#)

## gr-drm

**Tags:** sdr, drm, digital radio, short wave  
**Developer:** Felix Wunsch <felix.wunsch@kit.edu>  
**Dependencies:** None  
**Repository:** <https://github.com/kit-celigr-drm>  
**Copyright Owner:** Felix Wunsch  
**Brief:** DRM/DRM+ transmitter

### Module Info

## GR-DRM

A SOFTWARE DRM/DRM+ TRANSMITTER FOR GNU RADIO

### Contents

- 1: Installation
- 2: Usage
- 3: Features
- 4: (Current) Constraints

## Current Limitations

Still has issues:

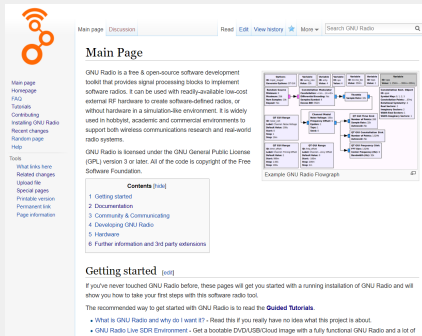
- Only shows OOTs that have a PyBOMBS recipe (so we are likely missing some valuable OOTs)
- Gets its info by parsing OOT's MANIFEST files, many of which don't exist
- The parser has trouble with OOTs not hosted on github
- No way to know if a PyBOMBS recipe is actually an OOT

# Documentation

# Documentation

## Current state of GNU Radio's Documentation:

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



The screenshot shows the GNU Radio Main Page. At the top left is the GNU Radio logo. Below it is a sidebar with navigation links: Main page, Homepage, FAQ, Tutorial, Contributing, Installing GNU Radio, Recent changes, Random page, and Help. Under 'Tools', there are links for 'What links here', 'Related changes', 'Upload file', 'Special pages', 'Printable version', 'Permanent link', and 'Page information'. The main content area has a search bar and a 'Main Page' heading. Below the heading is a paragraph of introductory text about GNU Radio as a free & open-source software development toolkit. To the right of this text is a flowchart titled 'Example GNU Radio Flowgraph'. Below the text is a table of contents with links: 1 Getting started, 2 Documentation, 3 Community & Communicating, 4 Developing GNU Radio, 5 Hardware, and 6 Further information and 3rd party extensions. Below the table of contents is a 'Getting started' section with a sub-heading and introductory text. At the bottom of the page, there are navigation icons for back, forward, and search.



# 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

- ▼ GNU Radio Manual and C++ API Reference
- ▶ Usage Manual
- ▼ Components
  - GNU Radio Blocks
  - In-line components
  - Analog Modulation
  - ▼ Audio Interface
- Introduction
- Usage
  - Adding a New Audio Machine
  - Standard GNU Radio Blocks
  - Channel Model Blocks
  - ControlPort
  - Digital Modulation
  - Packet Communications
  - FunCube Dongle Source
  - Forward Error Correction
  - FFT Signal Processing Blocks
  - Filter Signal Processing Blocks
  - QT Graphical User Interface
  - UHD Interface
  - Voice Coders and Decoders (Voocoders)
  - Zeromq
- License
- Modules
- Namespaces
- Classes
- Files

### 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="pcn device name. E.g., hi:0,0 or surround1 or
                 /dev/dsp")
audio_rate = 32000
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="pcn input device name. E.g., hi:0,0 or /dev/dsp")
audio_rate = 32000
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 static_cast<
```

Generated by doxygen 1.8.13

# Sphinx Manual

## Sphinx Contains:

- Auto-generated Sphinx stuff
- That's it

GNU Radio 3.7.13.4 documentation » next | modules | index

### Table Of Contents

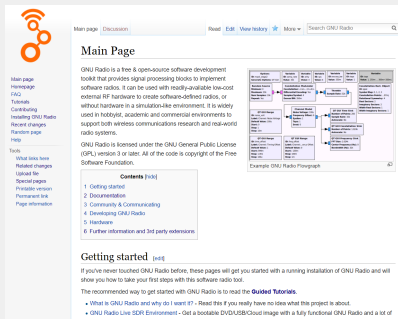
- gprutils
- Runtime
  - PMT
  - Audio Signals
  - Boolean Operators
  - Byte Operators
  - Channelizers
  - Channel Models
  - Coding Blocks
  - ControlPort Blocks
  - Debug Blocks
  - DTV Blocks
  - Equalizer Blocks
  - Error Coding Blocks
  - FCD Blocks
  - File Operator Blocks
  - Filter Blocks
  - Fourier Analysts
  - Impairment Model Blocks
  - Instrumentation Blocks
  - Level Control Blocks
  - Math Operator Blocks
  - Message Tool Blocks
  - Misc Blocks
  - Modulator Blocks
  - Networking Tools Blocks
  - NDAAR Blocks
  - OFDM Blocks
  - Packet Operator Blocks
  - Pager Blocks
  - Peak Detector Blocks
  - Resampler Blocks
  - Stream Operator Blocks
  - Stream Tag Tool Blocks
  - Symbol Coding Blocks
  - Synchronizer Blocks
  - Trellis Coding Blocks
  - Type Converter Blocks
  - UHD Blocks
  - Video Blocks
  - Waveform Generator

ptt_make_c32vector	
ptt_make_c64vector	
ptt_make_dict	Make an empty dictionary.
ptt_make_f32vector	
ptt_make_f64vector	
ptt_make_msg_accepter	make a msg_accepter
ptt_make_rectangular	Return a complex number constructed of the given real and imaginary parts.
ptt_make_s16vector	
ptt_make_s32vector	
ptt_make_s64vector	
ptt_make_u16vector	
ptt_make_u32vector	
ptt_make_u64vector	
ptt_make_uvector	
ptt_make_vector	Make a vector of length <code>n</code> , with initial values set to <code>val</code> .
ptt_msg	Apply element-wise to the elements of <code>lst</code> and returns a list of the results, in order.
ptt_number	Return the first sublist of whose car is <code>num</code> .
ptt_nseq	Return the first sublist of whose car is <code>num</code> .
ptt_nseq	Return the first sublist of whose car is <code>num</code> .
ptt_msg_accepter_ref	Return underlying msg_accepter.
ptt_nth	locates element of <code>lst</code> at index <code>n</code> .
ptt_nthcar	returns the tail of that would be obtained by calling <code>cdr</code> times in succession.
ptt_ptt_vector_cdouble	Proxy of C++ <code>std::vector&lt;std::complex&lt;(double)&gt;&gt;</code> class.
ptt_ptt_vector_cfloat	Proxy of C++ <code>std::vector&lt;std::complex&lt;(float)&gt;&gt;</code> class.
ptt_ptt_vector_double	Proxy of C++ <code>std::vector&lt;(double)&gt;</code> class.
ptt_ptt_vector_float	Proxy of C++ <code>std::vector&lt;(float)&gt;</code> class.
ptt_ptt_vector_int16	Proxy of C++ <code>std::vector&lt;int16_t&gt;</code> class.
ptt_ptt_vector_int32	Proxy of C++ <code>std::vector&lt;int32_t&gt;</code> class.
ptt_ptt_vector_int8	Proxy of C++ <code>std::vector&lt;int8_t&gt;</code> class.
ptt_ptt_vector_uint16	Proxy of C++ <code>std::vector&lt;uint16_t&gt;</code> class.
ptt_ptt_vector_uint32	Proxy of C++ <code>std::vector&lt;uint32_t&gt;</code> class.
ptt_ptt_vector_uint8	Proxy of C++ <code>std::vector&lt;uint8_t&gt;</code> class.

# Wiki

## Wiki Contains:

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



The screenshot shows the GNU Radio Wiki Main Page. At the top, there is a navigation bar with links for 'Main page', 'Discussion', 'Read', 'Edit', 'View history', and 'More'. A search box is located on the right. The main heading is 'Main Page'. Below this, there is a paragraph describing GNU Radio as a free & open-source software development toolkit for implementing software radios. To the right of this text is a flowchart titled 'Example GNU Radio Flowgraph'. Below the flowchart is a 'Contents' table of contents with links for 'Getting started', 'Documentation', 'Community & Communicating', 'Developing GNU Radio', 'Hardware', and 'Further information and 3rd party extensions'. Below the contents is a 'Getting started' section with a link to '[edit]'. The text below 'Getting started' explains that if you've never touched GNU Radio before, these pages will get you started with a running installation. It also lists recommended steps: reading the 'Guided Tutorials', reading 'What is GNU Radio and why do I want it?', and getting the 'GNU Radio Live SDR Environment'.

# 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
- Link in GRC to Github's built in text editor and commit system for that specific block
- An option for an indirect way of submitting changes, e.g. with a form, for block experts
- Use wiki for anything appropriate since it's easy to change

## Efforts to Date

- 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 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