

As a Newbie

GNU Radio for SDR Developers

Exploring the Open-Source DSP Framework
Andrew Lammas, Flinders University



Why GNU Radio?

- Free & open-source DSP environment for SDR
- Alternative to
 - MATLAB/Simulink,
 - LabVIEW,
 - Vendor SDKs
- Supports rapid prototyping and teaching/research
- Broad hardware support

Research & Industry Applications

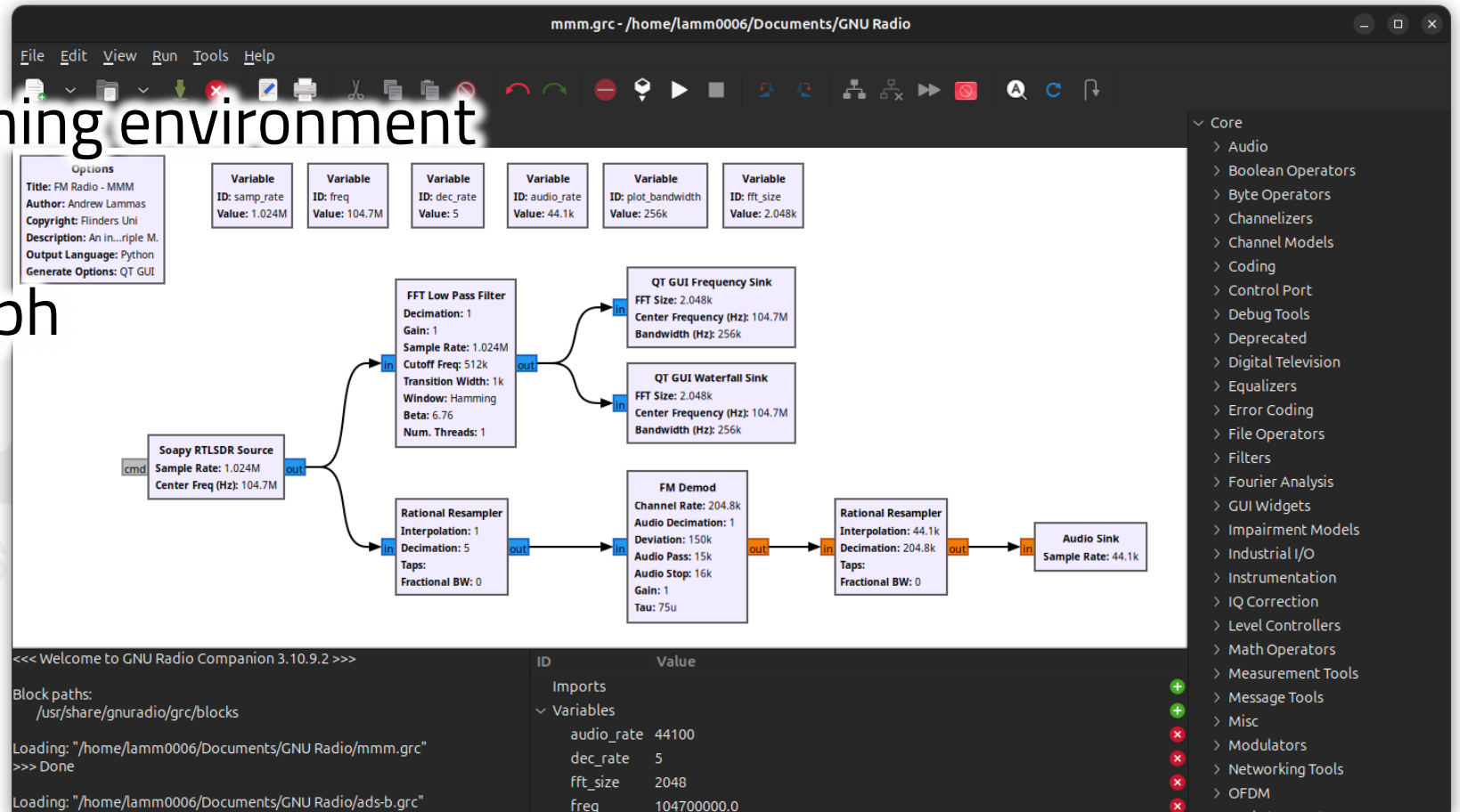
- Software-Defined Radio (SDR) Development
- Wireless Communications
- Spectrum Monitoring and Analysis
- Education and Training
- Satellite and Space Applications
- Defence and Security
- Amateur Radio and Hobbyist Projects

Architecture Overview

- Flowgraph Design:
 - Python API
 - GRC GUI
- DSP Kernels: Optimized C++ blocks
- Scheduler: Thread-per-block model
- Hardware Interfaces
 - UHD
 - SoapySDR
 - RTL-SDR

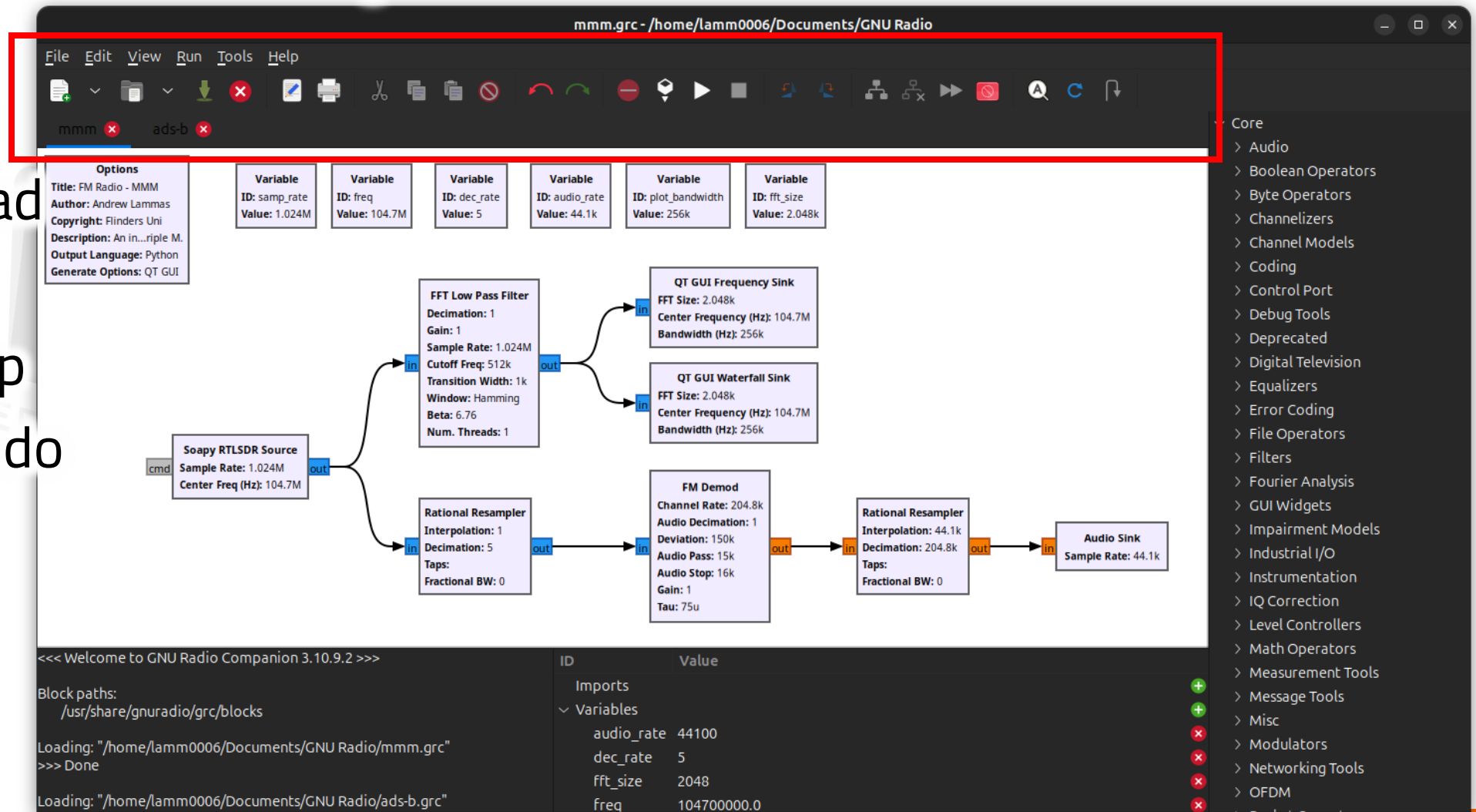
GNU Radio Companion

- Graphical programming environment
- Block diagram
- Dataflow - Flowgraph



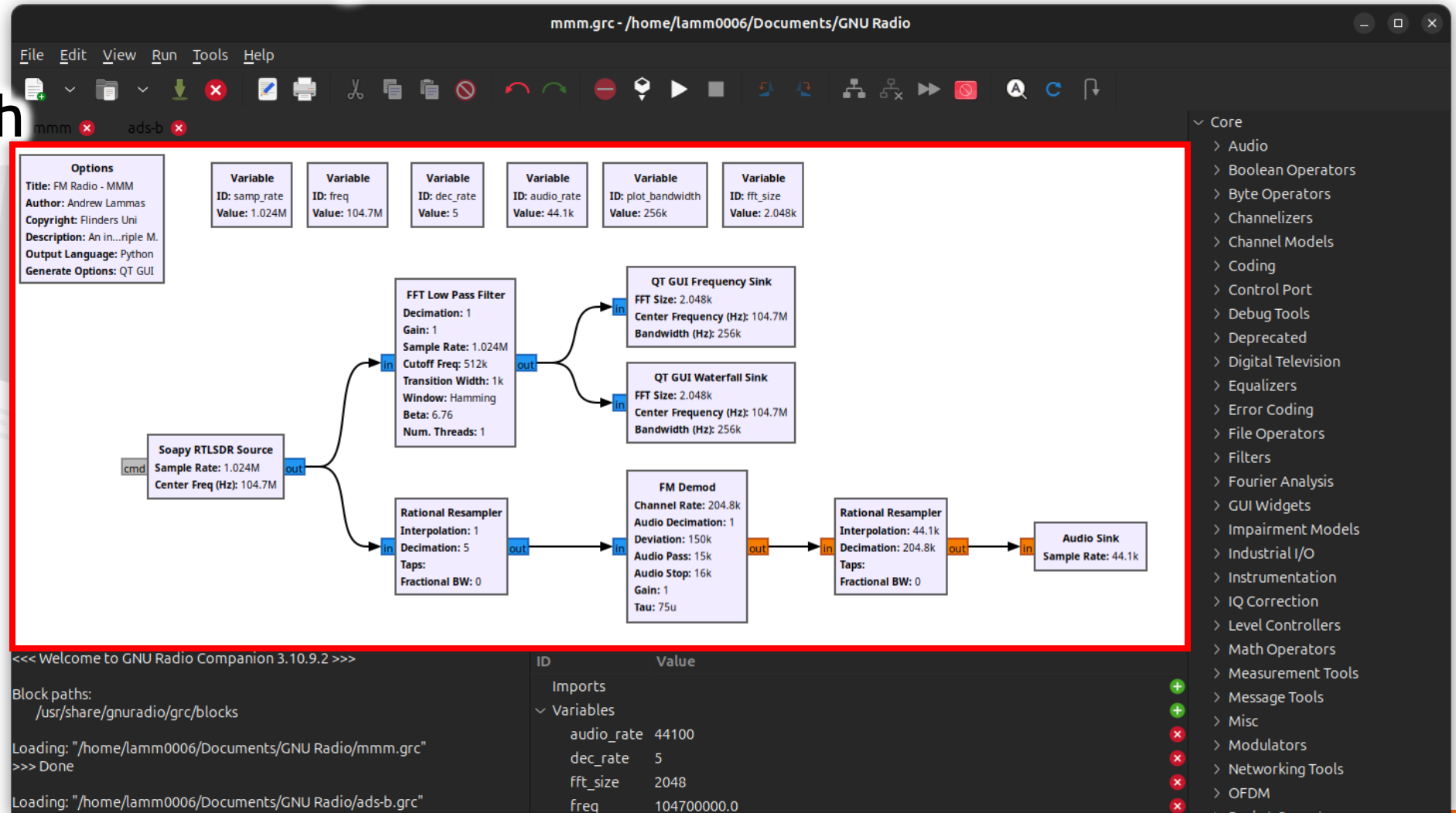
GNU Radio Companion

- Controls
- Save/Load
- Print
- Play/Stop
- Redo/Undo



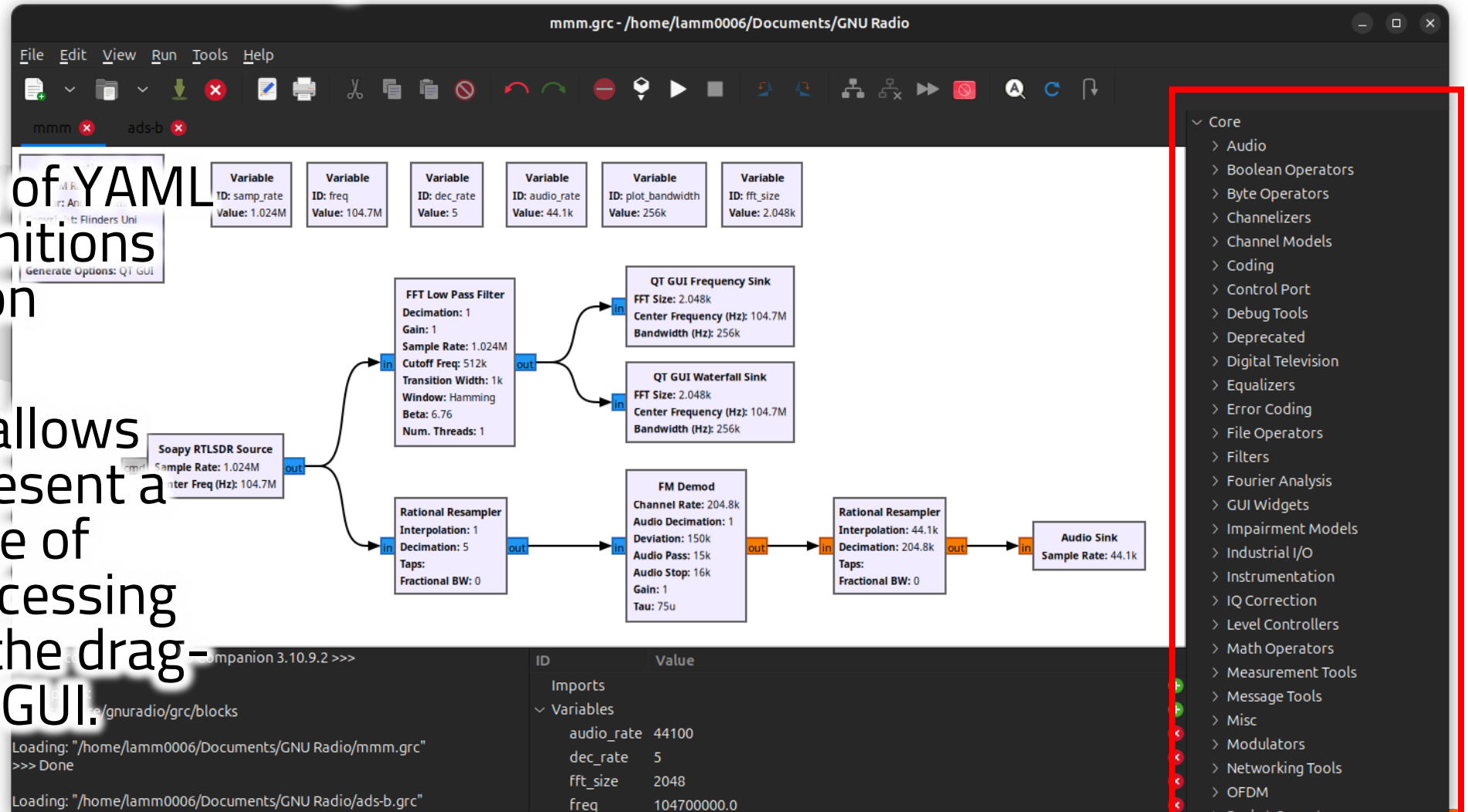
GNU Radio Companion

• Flowgraph



GNU Radio Companion

- Library
- Collection of YAML block definitions and Python scripts.
- It's what allows GRC to present a wide range of signal processing blocks in the drag-and-drop GUI.



GNU Radio Companion

- Terminal
- Shows
 - run command
 - outputted text
 - errors

The screenshot displays the GNU Radio Companion (GRC) interface. The main window shows a block diagram for an FM radio receiver. The blocks are connected as follows: Soapy RTLSDR Source (Sample Rate: 1.024M, Center Freq (Hz): 104.7M) connects to an FFT Low Pass Filter (Decimation: 1, Gain: 1, Sample Rate: 1.024M, Cutoff Freq: 512k, Transition Width: 1k, Window: Hamming, Beta: 6.76, Num. Threads: 1). The output of the filter splits into two paths: one to a QT GUI Frequency Sink (FFT Size: 2.048k, Center Frequency (Hz): 104.7M, Bandwidth (Hz): 256k) and another to a QT GUI Waterfall Sink (FFT Size: 2.048k, Center Frequency (Hz): 104.7M, Bandwidth (Hz): 256k). The other path from the filter goes to a Rational Resampler (Interpolation: 1, Decimation: 5, Taps: Fractional BW: 0), which then connects to an FM Demod (Channel Rate: 204.8k, Audio Decimation: 1, Deviation: 150k, Audio Pass: 15k, Audio Stop: 16k, Gain: 1, Tau: 75u). The output of the FM Demod goes to another Rational Resampler (Interpolation: 44.1k, Decimation: 204.8k, Taps: Fractional BW: 0), which finally connects to an Audio Sink (Sample Rate: 44.1k).

At the top, there are several variable blocks with the following values:

ID	Value
samp_rate	1.024M
freq	104.7M
dec_rate	5
audio_rate	44.1k
plot_bandwidth	256k
fft_size	2.048k

On the left, an 'Options' panel shows the title 'FM Radio - MMM', author 'Andrew Lammas', and description 'An in...riple M. Output Language: Python T GUI'.

At the bottom, a terminal window is open, showing the following text:

```
<<< Welcome to GNU Radio Companion 3.10.9.2 >>>
Block paths:
  /usr/share/gnuradio/grc/blocks
Loading: "/home/lamm0006/Documents/GNU Radio/mmm.grc"
>>> Done
Loading: "/home/lamm0006/Documents/GNU Radio/ads-b.grc"
```

On the right, a sidebar lists various tool categories, including Core, Audio, Boolean Operators, Byte Operators, Channelizers, Channel Models, Coding, Control Port, Debug Tools, Deprecated, Digital Television, Equalizers, Error Coding, File Operators, Filters, Fourier Analysis, GUI Widgets, Impairment Models, Industrial I/O, Instrumentation, IQ Correction, Level Controllers, Math Operators, Measurement Tools, Message Tools, Misc, Modulators, Networking Tools, and OFDM.

GNU Radio Companion

- Variables
- Name
- Value

The screenshot displays the GNU Radio Companion (grc) interface. The main window shows a block diagram for an FM radio receiver. The blocks are connected as follows: Soapy RTLSDR Source (Sample Rate: 1.024M, Center Freq (Hz): 104.7M) connects to an FFT Low Pass Filter (Decimation: 1, Gain: 1, Sample Rate: 1.024M, Cutoff Freq: 512k, Transition Width: 1k, Window: Hamming, Beta: 6.76, Num. Threads: 1) and a Rational Resampler (Interpolation: 1, Decimation: 5, Taps: Fractional BW: 0). The FFT Low Pass Filter output connects to two sinks: QT GUI Frequency Sink (FFT Size: 2.048k, Center Frequency (Hz): 104.7M, Bandwidth (Hz): 256k) and QT GUI Waterfall Sink (FFT Size: 2.048k, Center Frequency (Hz): 104.7M, Bandwidth (Hz): 256k). The Rational Resampler output connects to an FM Demod (Channel Rate: 204.8k, Audio Decimation: 1, Deviation: 150k, Audio Pass: 15k, Audio Stop: 16k, Gain: 1, Tau: 75u) and another Rational Resampler (Interpolation: 44.1k, Decimation: 204.8k, Taps: Fractional BW: 0). The FM Demod output connects to the second Rational Resampler, which then connects to an Audio Sink (Sample Rate: 44.1k).

At the top, there are several variable blocks with the following values:

ID	Value
samp_rate	1.024M
freq	104.7M
dec_rate	5
audio_rate	44.1k
plot_bandwidth	256k
fft_size	2.048k

On the right side, there is a sidebar with a tree view of the block categories. The 'Core' category is expanded, showing sub-categories like Audio, Boolean Operators, Byte Operators, Channelizers, Channel Models, Coding, Control Port, Debug Tools, Deprecated, Digital Television, Equalizers, Error Coding, File Operators, Filters, Fourier Analysis, GUI Widgets, Impairment Models, Industrial I/O, Instrumentation, IQ Correction, Level Controllers, Math Operators, Measurement Tools, Message Tools, Misc, Modulators, Networking Tools, and OFDM.

At the bottom, there is a terminal window showing the following output:

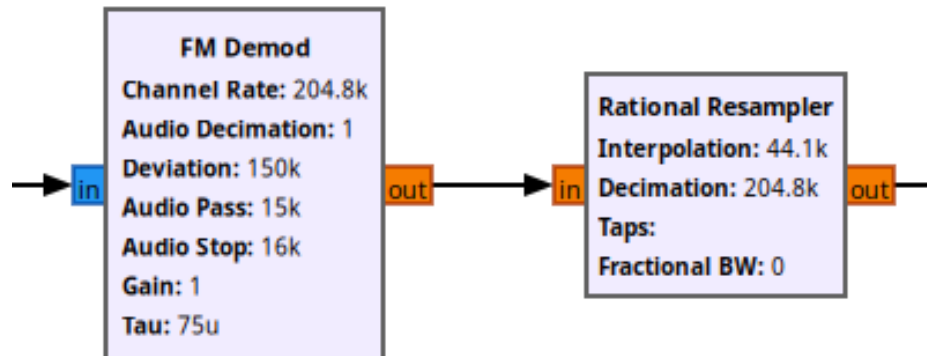
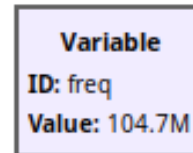
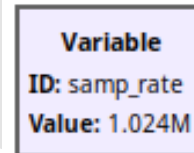
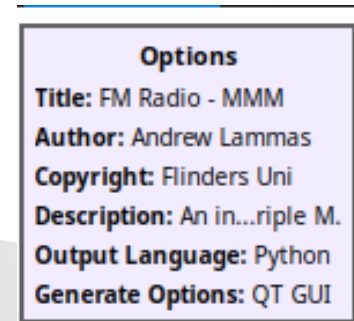
```
<<< Welcome to GNU Radio Companion 3.10.9.2 >>>
Block paths:
  /usr/share/gnuradio/grc/blocks
Loading: "/home/lamm0006/Documents/GNU Radio/mmm.grc"
>>> Done
Loading: "/home/lamm0006/Documents/GNU Radio/ads-b.grc"
```

Below the terminal window, there is a table showing the current values of the variables:

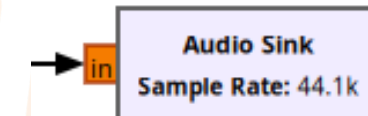
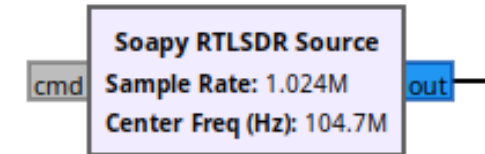
ID	Value
Imports	
Variables	
audio_rate	44100
dec_rate	5
fft_size	2048
freq	104700000.0

GNU Radio Companion Blocks

- Options
- Variables
- Processing

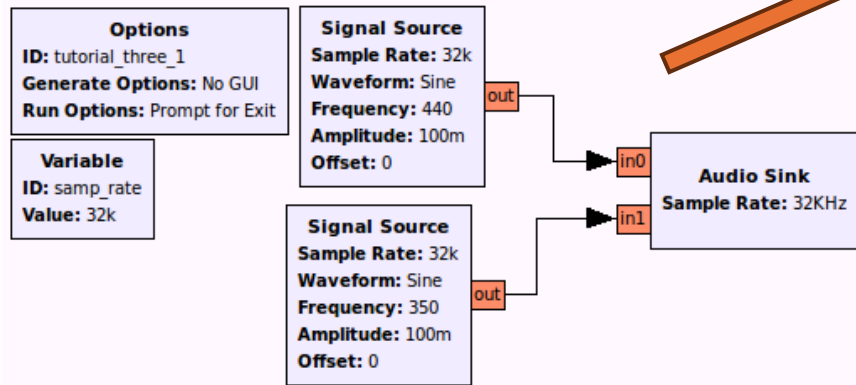


- Sources
 - Log files
 - Receivers
- Sinks
 - Transmitters
 - Log files
 - Audio
 - Visualisations



What Happens When You Press Play?

- Flowgraph is compiled into code
 - Python (default)
 - C++
- Compiled code can be run without companion



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  #
5  # SPDX-License-Identifier: GPL-3.0
6  #
7  # GNU Radio Python Flow Graph
8  # Title: tutorial_three_1
9  # GNU Radio version: 3.8.0.0
10
11 from distutils.version import StrictVersion
12
13 if __name__ == '__main__':
14     import ctypes
15     import sys
16     if sys.platform.startswith('linux'):
17         try:
18             x11 = ctypes.cdll.LoadLibrary('libX11.so')
19             x11.XInitThreads()
20         except:
21             print("Warning: failed to XInitThreads()")
22
23 from gnuradio import analog
24 from gnuradio import audio
25 from gnuradio import gr
26 from gnuradio.filter import firdes
27 import sys
28 import signal
29 from PyQt5 import Qt
30 from argparse import ArgumentParser
31 from gnuradio.eng_arg import eng_float, intx
32 from gnuradio import eng_notation
33
```

Extending GNU Radio

- Out-of-tree (OOT) modules with custom DSP kernels
- Python bindings auto-generated
- Use gr-modtool to scaffold blocks
- Community libraries:
 - gr-lte,
 - gr-dvb,
 - satellite comms,
 - security tools

Strengths

- Incredibly easy to get started
- Large open-source ecosystem
- Hardware abstraction via UHD/SoapySDR
- Mix of GUI prototyping - Python/C++ coding
- Active community & GNU Radio Conference

Workflow Example: FM Receiver

Options
 Title: FM Radio - MMM
 Author: Andrew Lammass
 Copyright: Flinders Uni
 Description: An in...rple M.
 Output Language: Python
 Generate Options: QT GUI

Variable
 ID: samp_rate
 Value: 1.024M

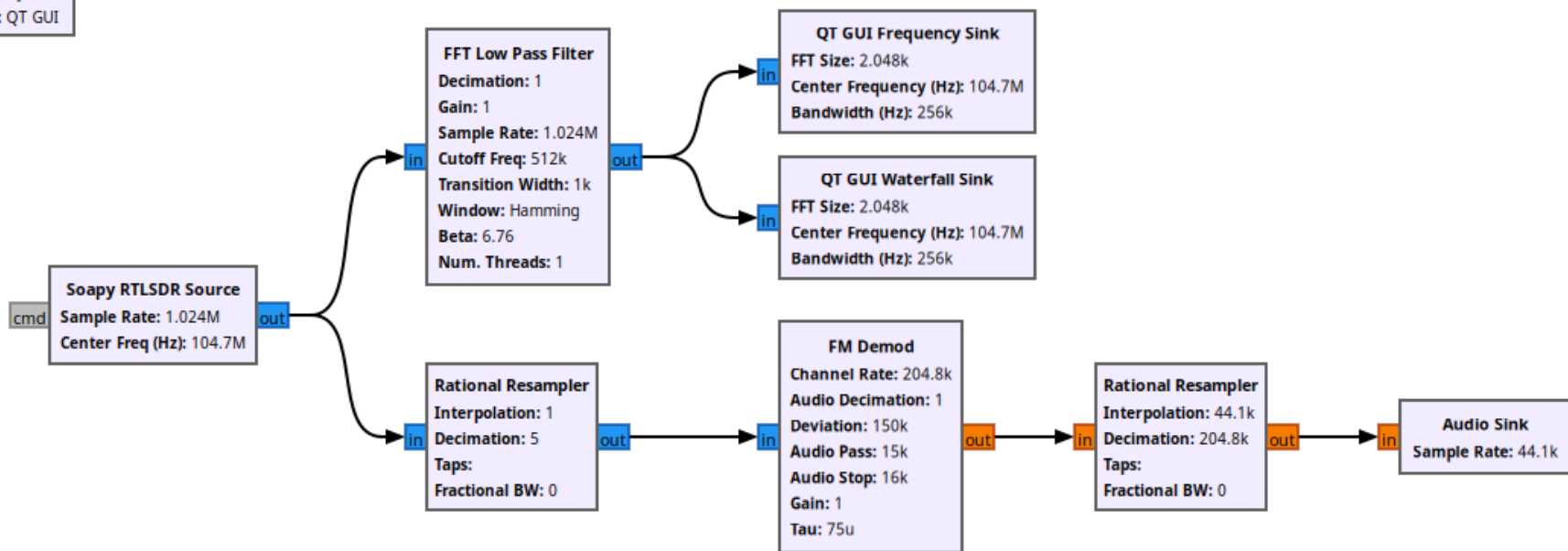
Variable
 ID: freq
 Value: 104.7M

Variable
 ID: dec_rate
 Value: 5

Variable
 ID: audio_rate
 Value: 44.1k

Variable
 ID: plot_bandwidth
 Value: 256k

Variable
 ID: fft_size
 Value: 2.048k



<<< Welcome to GNU Radio Companion 3.10.9.2 >>>

Block paths:
 /usr/share/gnuradio/grc/blocks

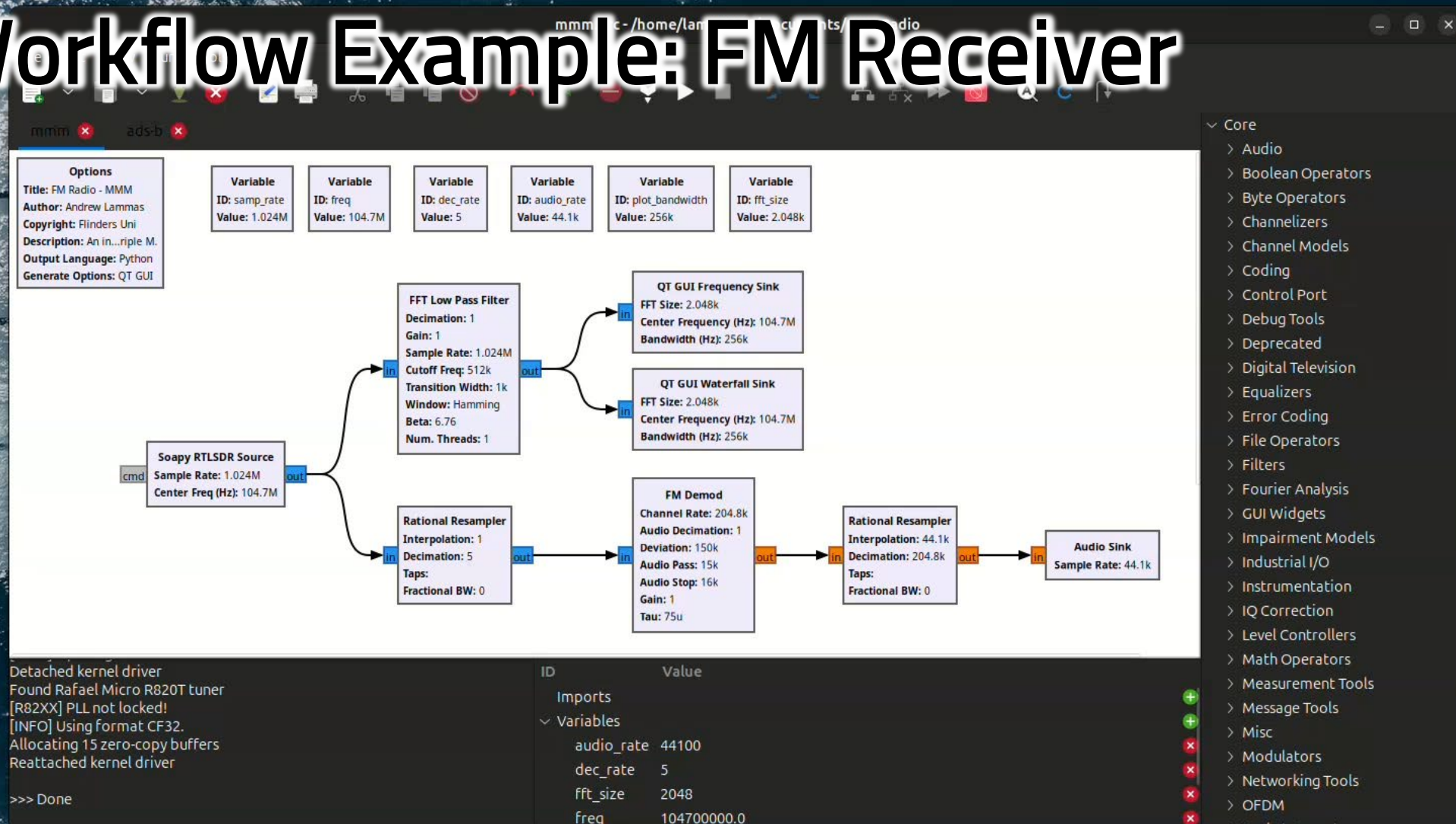
Loading: "/home/lamm0006/Documents/GNU Radio/mmm.grc"
 >>> Done

Loading: "/home/lamm0006/Documents/GNU Radio/ads-b.grc"

ID	Value
Imports	
Variables	
audio_rate	44100
dec_rate	5
fft_size	2048
freq	104700000.0

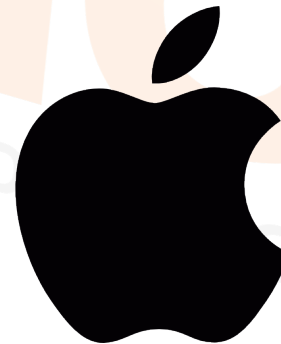
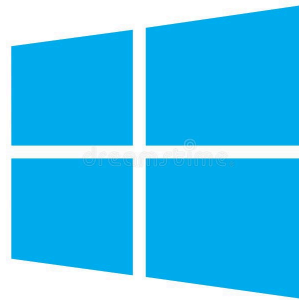
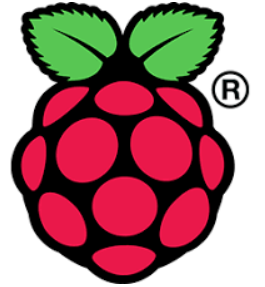
- Core
 - > Audio
 - > Boolean Operators
 - > Byte Operators
 - > Channelizers
 - > Channel Models
 - > Coding
 - > Control Port
 - > Debug Tools
 - > Deprecated
 - > Digital Television
 - > Equalizers
 - > Error Coding
 - > File Operators
 - > Filters
 - > Fourier Analysis
 - > GUI Widgets
 - > Impairment Models
 - > Industrial I/O
 - > Instrumentation
 - > IQ Correction
 - > Level Controllers
 - > Math Operators
 - > Measurement Tools
 - > Message Tools
 - > Misc
 - > Modulators
 - > Networking Tools
 - > OFDM

Workflow Example: FM Receiver



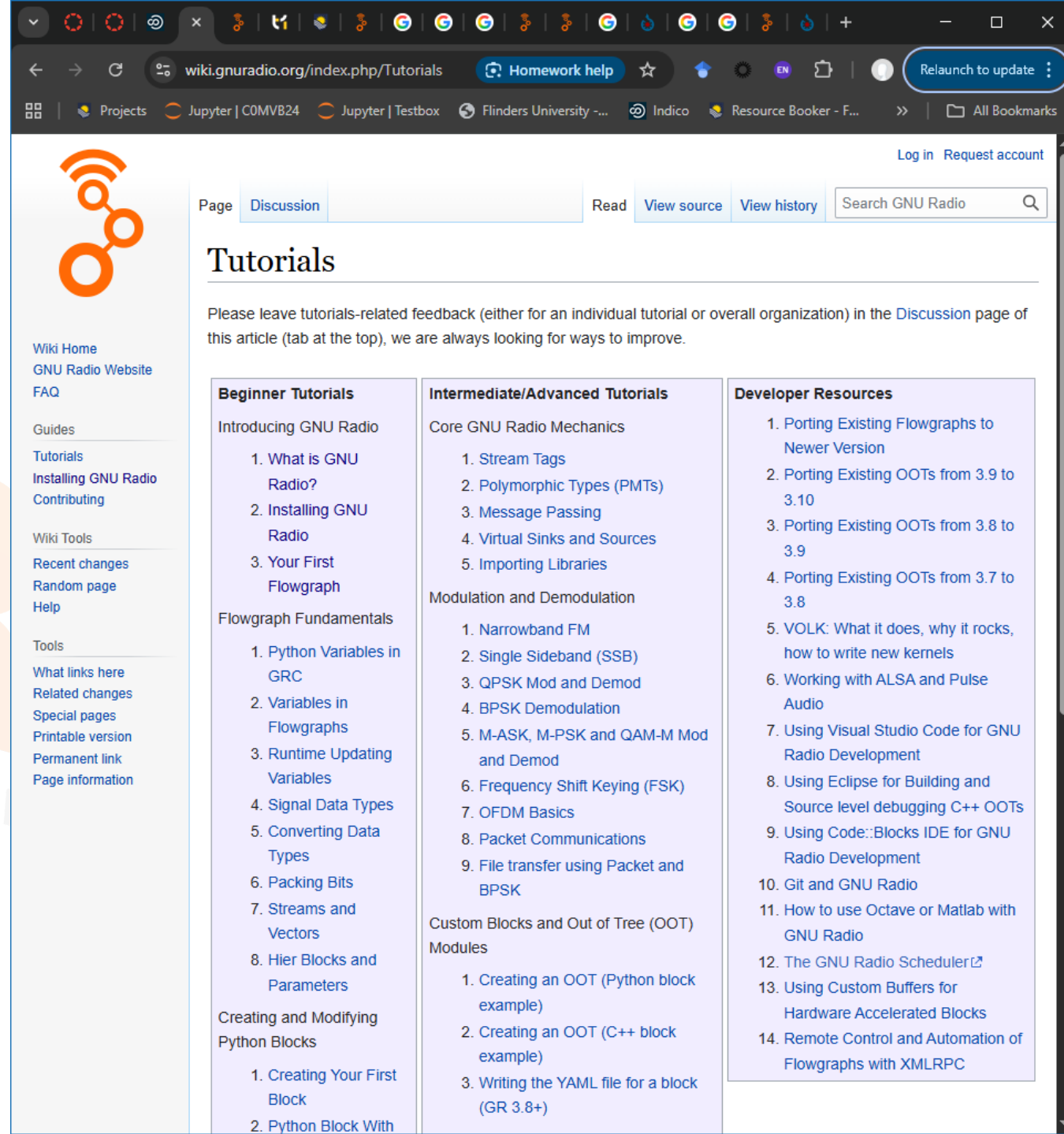
Getting Started

- Best on Linux
 - Ubuntu/Debian/Mint
 - Fedora
 - Other Linux distros
 - <https://repology.org/project/gnuradio/badges>
- Can be installed on
 - Windows
 - macOS
 - Via Radioconda



Further Work

- Explore gr- add-ons
- Resources
 - wiki.gnuradio.org
 - Tutorials
 - GNU Radio Conference talks



The screenshot shows the GNU Radio website's tutorial page. The browser's address bar displays wiki.gnuradio.org/index.php/Tutorials. The page features a sidebar with navigation links such as 'Wiki Home', 'GNU Radio Website', 'FAQ', 'Guides', 'Tutorials', 'Installing GNU Radio', 'Contributing', 'Wiki Tools', 'Recent changes', 'Random page', 'Help', 'Tools', 'What links here', 'Related changes', 'Special pages', 'Printable version', 'Permanent link', and 'Page information'. The main content area is titled 'Tutorials' and includes a search bar and tabs for 'Page', 'Discussion', 'Read', 'View source', and 'View history'. A paragraph encourages users to provide feedback in the 'Discussion' tab. The page is organized into three columns: 'Beginner Tutorials' (including 'Introducing GNU Radio' and 'Flowgraph Fundamentals'), 'Intermediate/Advanced Tutorials' (including 'Core GNU Radio Mechanics' and 'Modulation and Demodulation'), and 'Developer Resources' (listing 14 items related to porting, development, and automation).

Page [Discussion](#) [Read](#) [View source](#) [View history](#)

Tutorials

Please leave tutorials-related feedback (either for an individual tutorial or overall organization) in the [Discussion](#) page of this article (tab at the top), we are always looking for ways to improve.

Beginner Tutorials	Intermediate/Advanced Tutorials	Developer Resources
Introducing GNU Radio <ol style="list-style-type: none">1. What is GNU Radio?2. Installing GNU Radio3. Your First Flowgraph	Core GNU Radio Mechanics <ol style="list-style-type: none">1. Stream Tags2. Polymorphic Types (PMTs)3. Message Passing4. Virtual Sinks and Sources5. Importing Libraries Modulation and Demodulation <ol style="list-style-type: none">1. Narrowband FM2. Single Sideband (SSB)3. QPSK Mod and Demod4. BPSK Demodulation5. M-ASK, M-PSK and QAM-M Mod and Demod6. Frequency Shift Keying (FSK)7. OFDM Basics8. Packet Communications9. File transfer using Packet and BPSK Custom Blocks and Out of Tree (OOT) Modules <ol style="list-style-type: none">1. Creating an OOT (Python block example)2. Creating an OOT (C++ block example)3. Writing the YAML file for a block (GR 3.8+)	<ol style="list-style-type: none">1. Porting Existing Flowgraphs to Newer Version2. Porting Existing OOTs from 3.9 to 3.103. Porting Existing OOTs from 3.8 to 3.94. Porting Existing OOTs from 3.7 to 3.85. VOLK: What it does, why it rocks, how to write new kernels6. Working with ALSA and Pulse Audio7. Using Visual Studio Code for GNU Radio Development8. Using Eclipse for Building and Source level debugging C++ OOTs9. Using Code::Blocks IDE for GNU Radio Development10. Git and GNU Radio11. How to use Octave or Matlab with GNU Radio12. The GNU Radio Scheduler13. Using Custom Buffers for Hardware Accelerated Blocks14. Remote Control and Automation of Flowgraphs with XMLRPC

Closing Thoughts

- Incredibly easy to get started
- Open-source general-purpose DSP framework
- Great for rapid prototyping and bridging theory and practice
- Supported by strong open-source community
- Extendable at various levels of abstraction
 - Blocks
 - Python
 - C++

Questions

