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

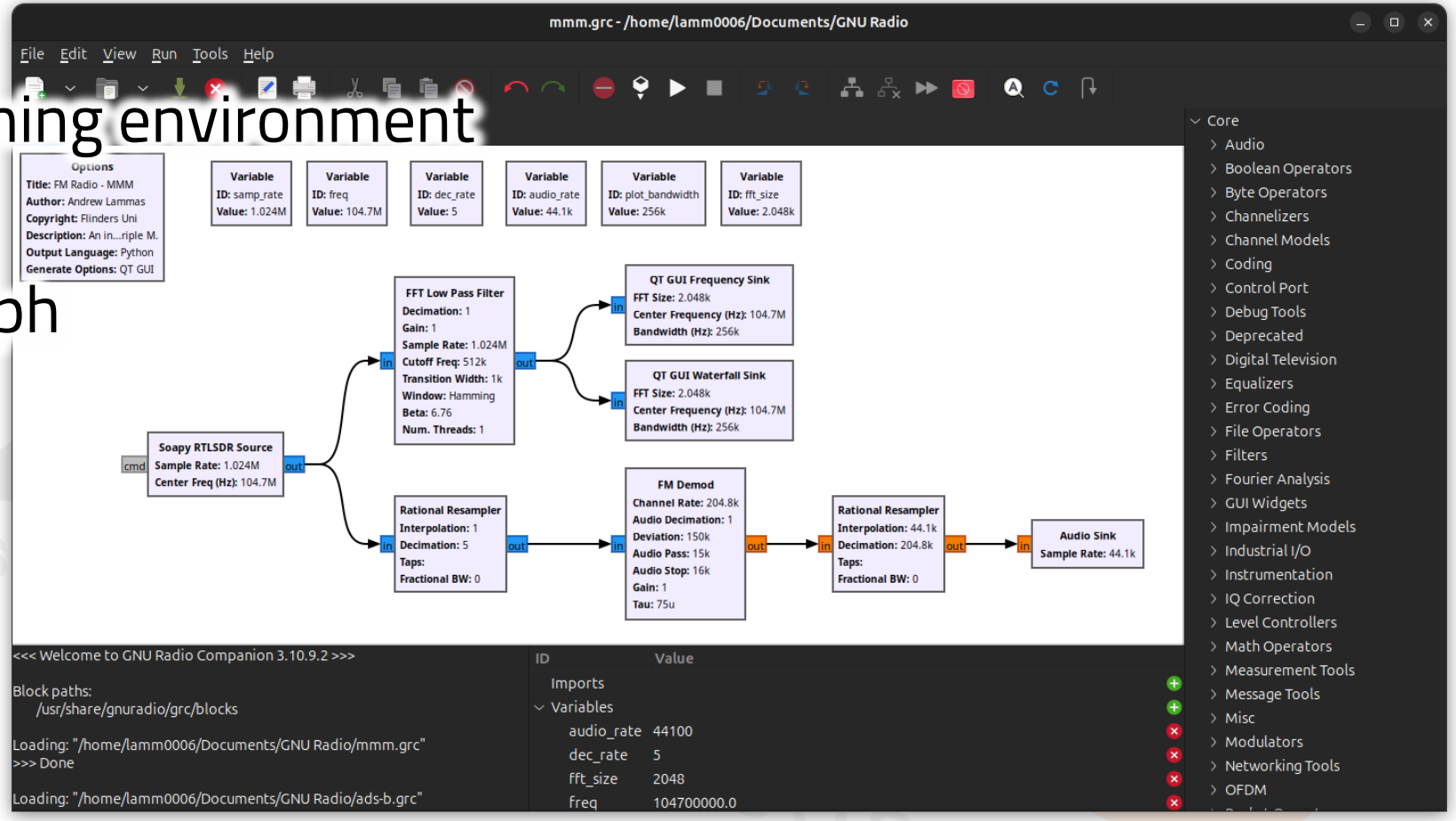
- Software-Defined Radio (SDR) Development
- Wireless Communications
- Spectrum Monitoring and Analysis
- Education and Training
- Satellite and Space Applications
- Defence and Security
- Amateur Radio and Hobbyist Projects

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

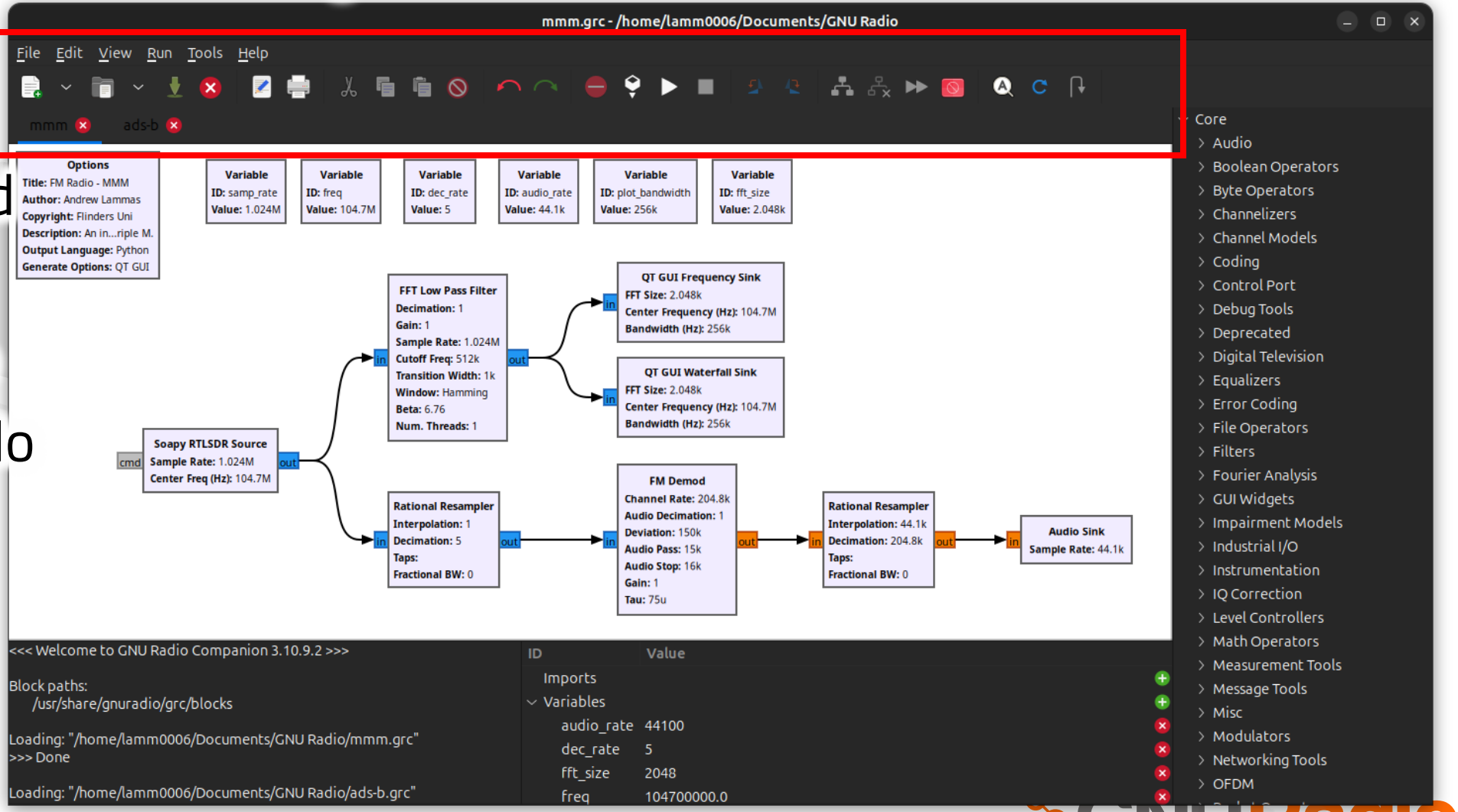
- Graphical programming environment
- Block diagram
- Dataflow - Flowgraph



GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

GNU Radio Companion

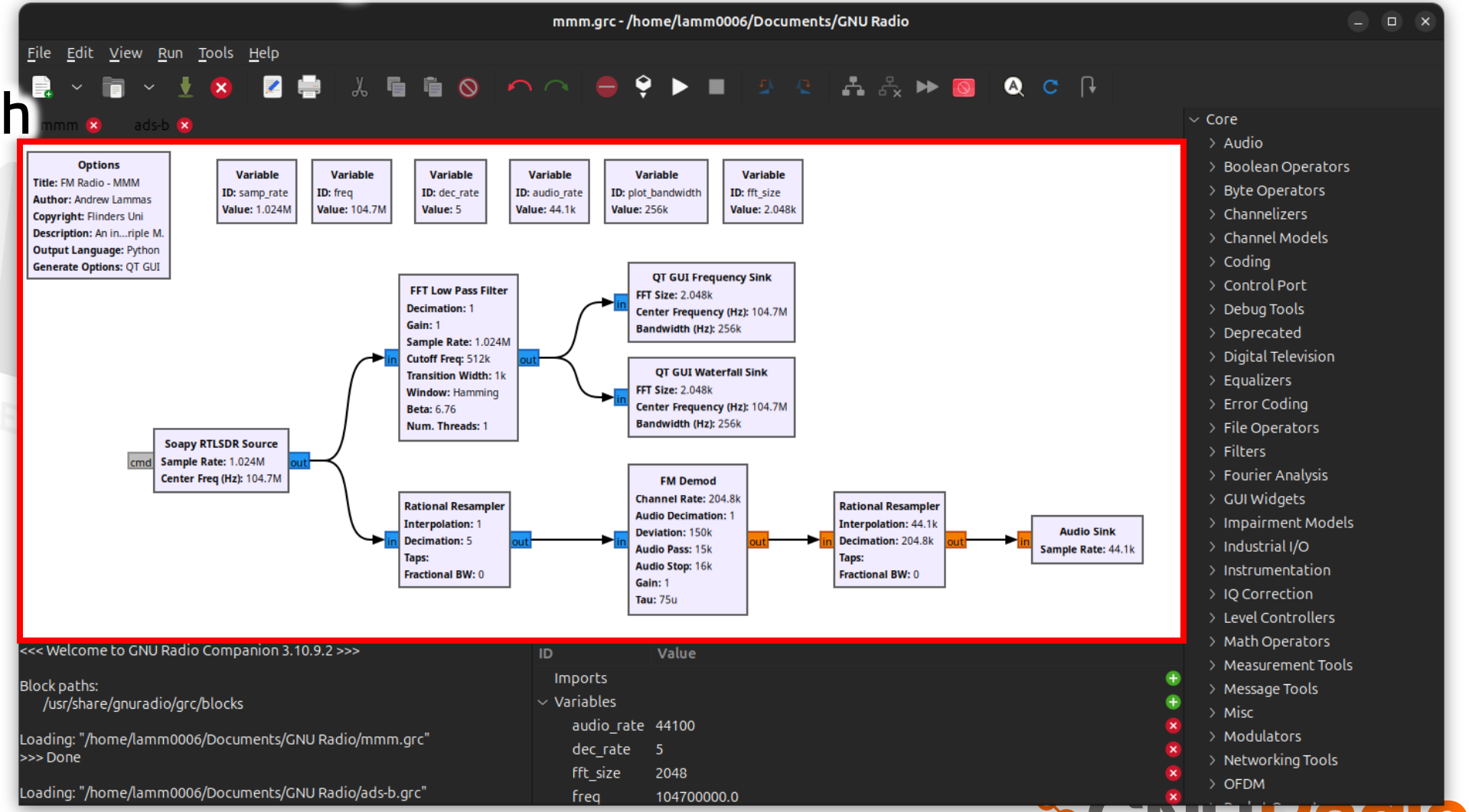
- Controls
- Save/Load
- Print
- Play/Stop
- Redo/Undo



GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

GNU Radio Companion

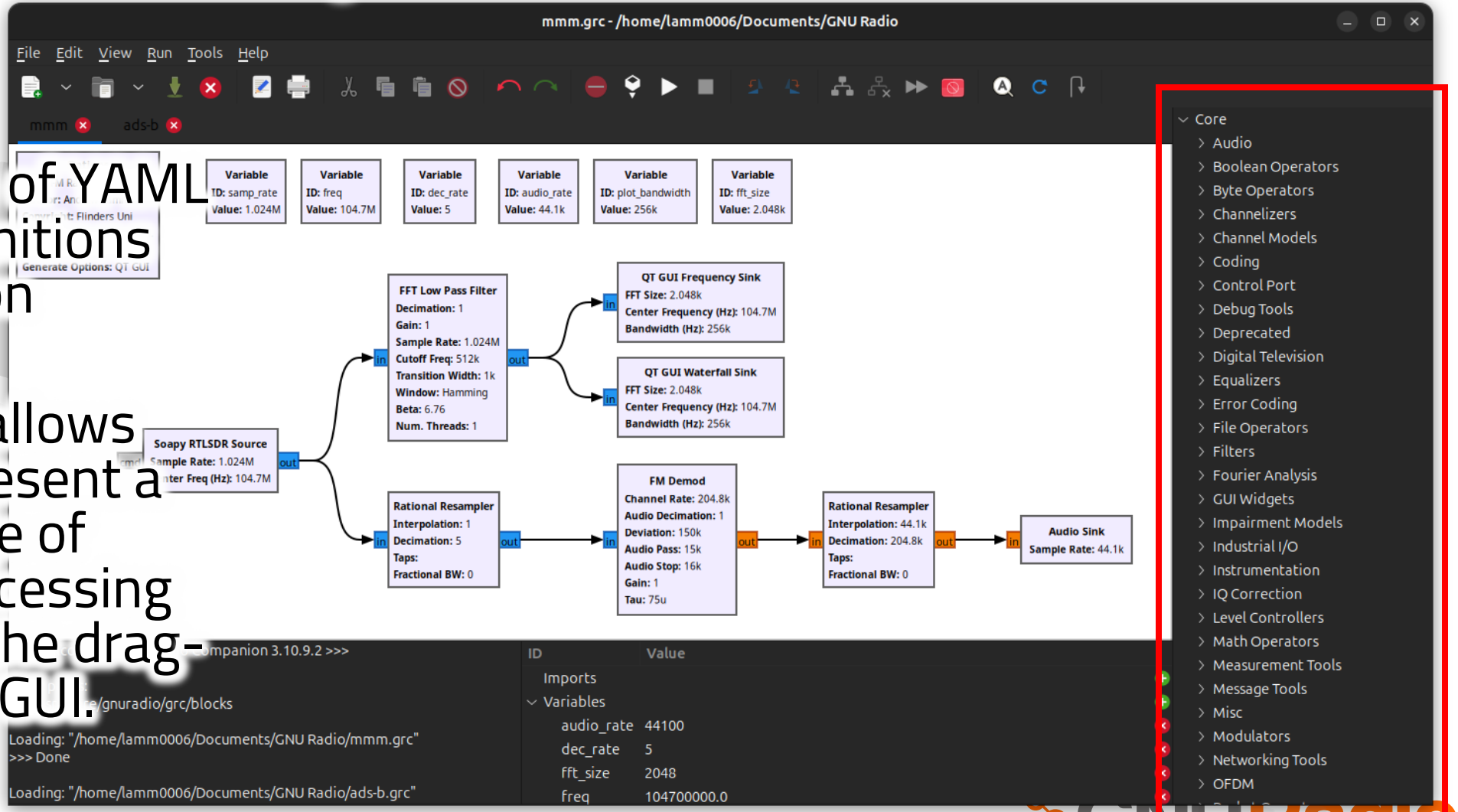
- Flowgraph



GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

GNU Radio Companion

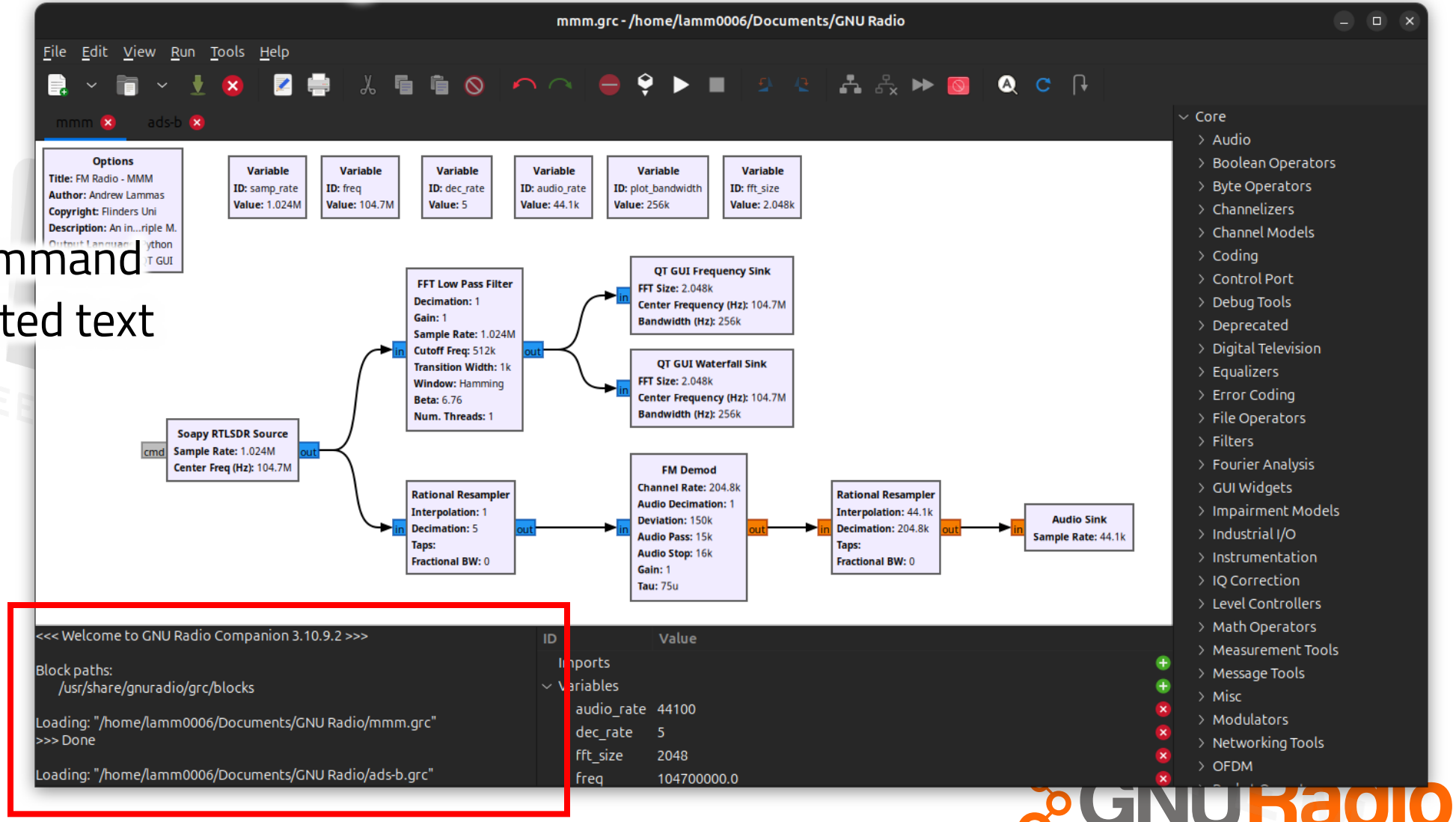
- Library
- Collection of YAML block definitions and Python scripts.
- It's what allows GRC to present a wide range of signal processing blocks in the drag-and-drop GUI.



GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

GNU Radio Companion

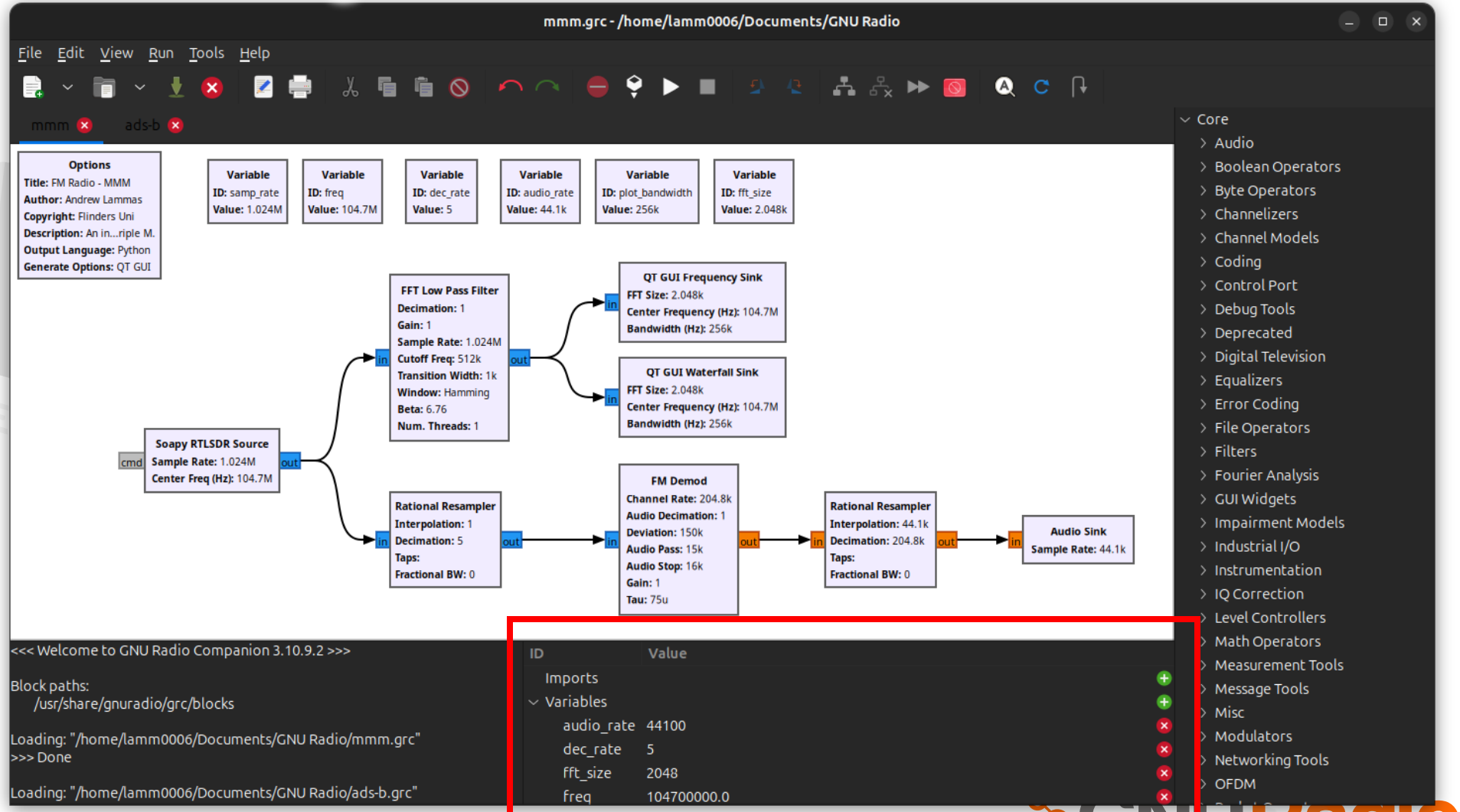
- Terminal
- Shows
  - run command
  - outputted text
  - errors



GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

GNU Radio Companion

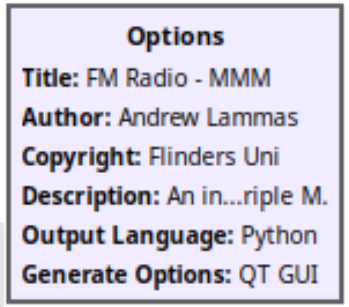
- Variables
- Name
- Value

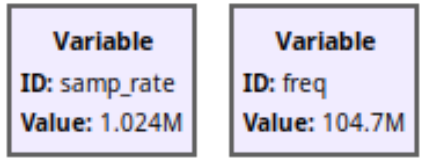


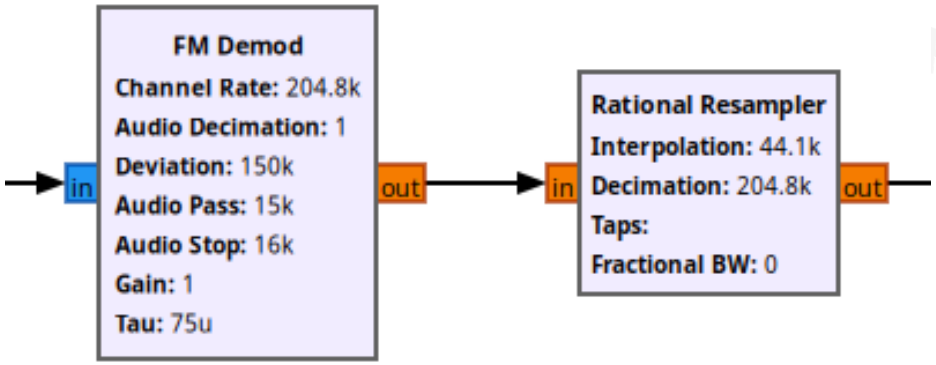
GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

GNU Radio Companion Blocks

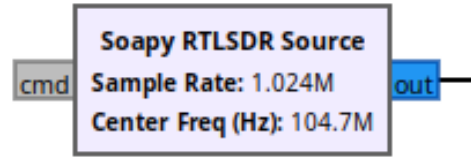
- Options
- Variables
- Processing

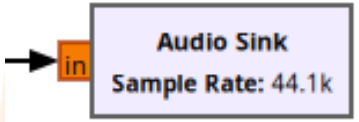






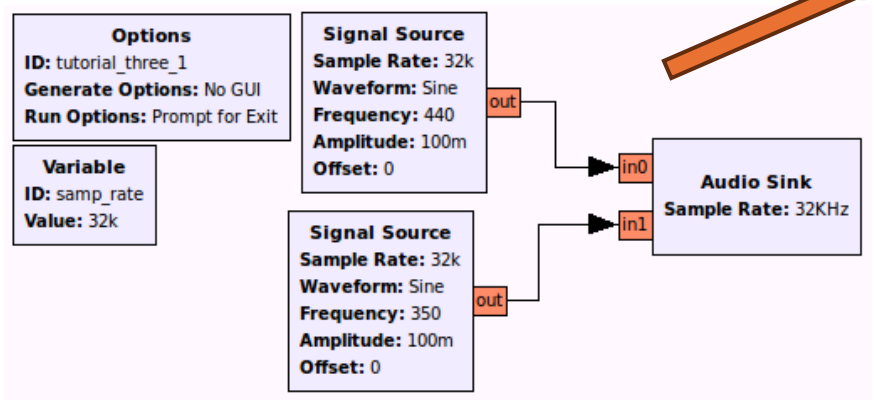
- Sources
  - Log files
  - Receivers
- Sinks
  - Transmitters
  - Log files
  - Audio
  - Visualisations





What Happens When You Press Play?

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




```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# SPDX-License-Identifier: GPL-3.0
#
# GNU Radio Python Flow Graph
# Title: tutorial_three_1
# GNU Radio version: 3.8.0.0

from distutils.version import StrictVersion

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print("Warning: failed to XInitThreads()")

from gnuradio import analog
from gnuradio import audio
from gnuradio import gr
from gnuradio.filter import firdes
import sys
import signal
from PyQt5 import Qt
from argparse import ArgumentParser
from gnuradio.eng_arg import eng_float, intx
from gnuradio import eng_notation
```

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

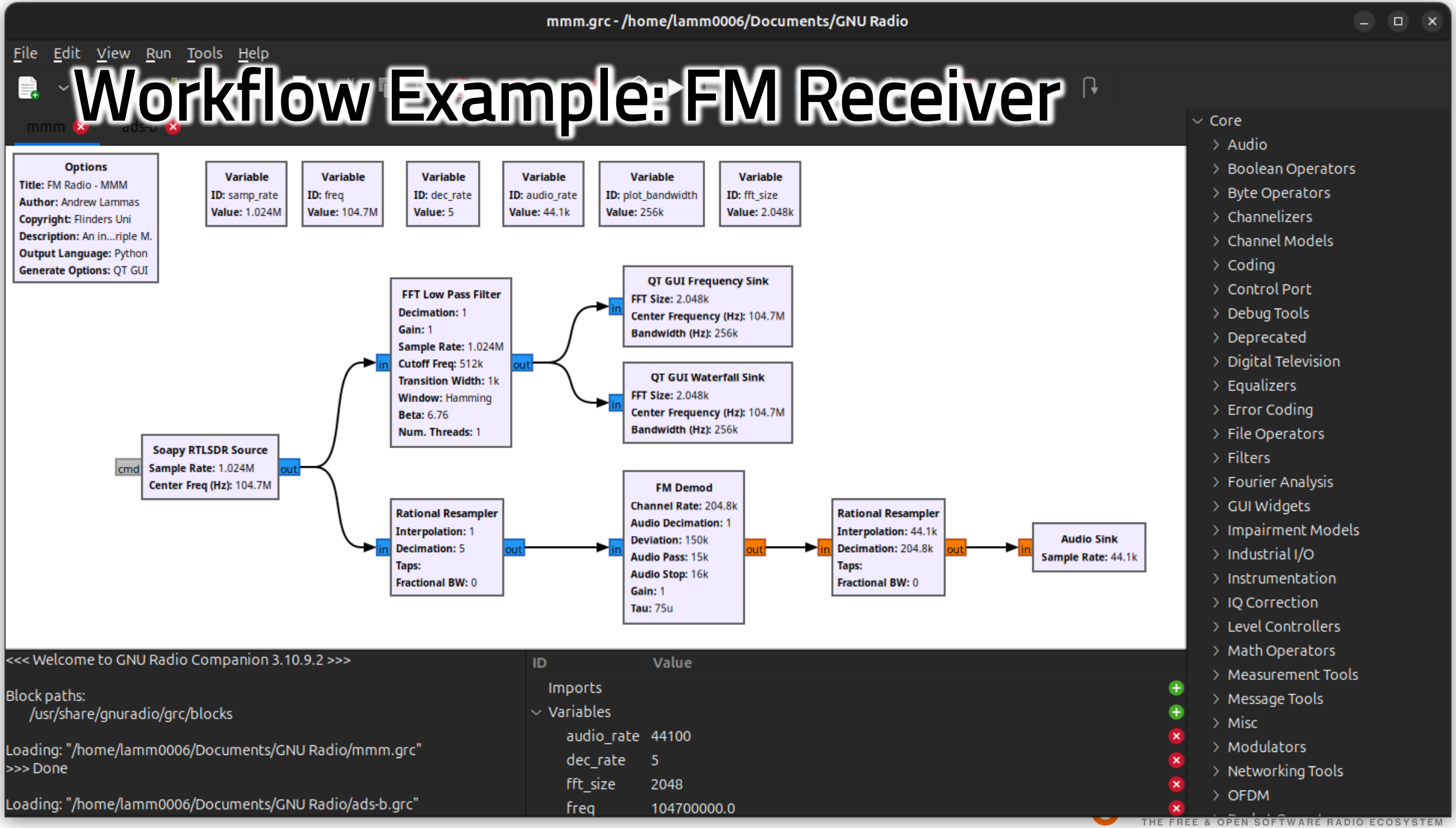
Extending GNU Radio

- 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

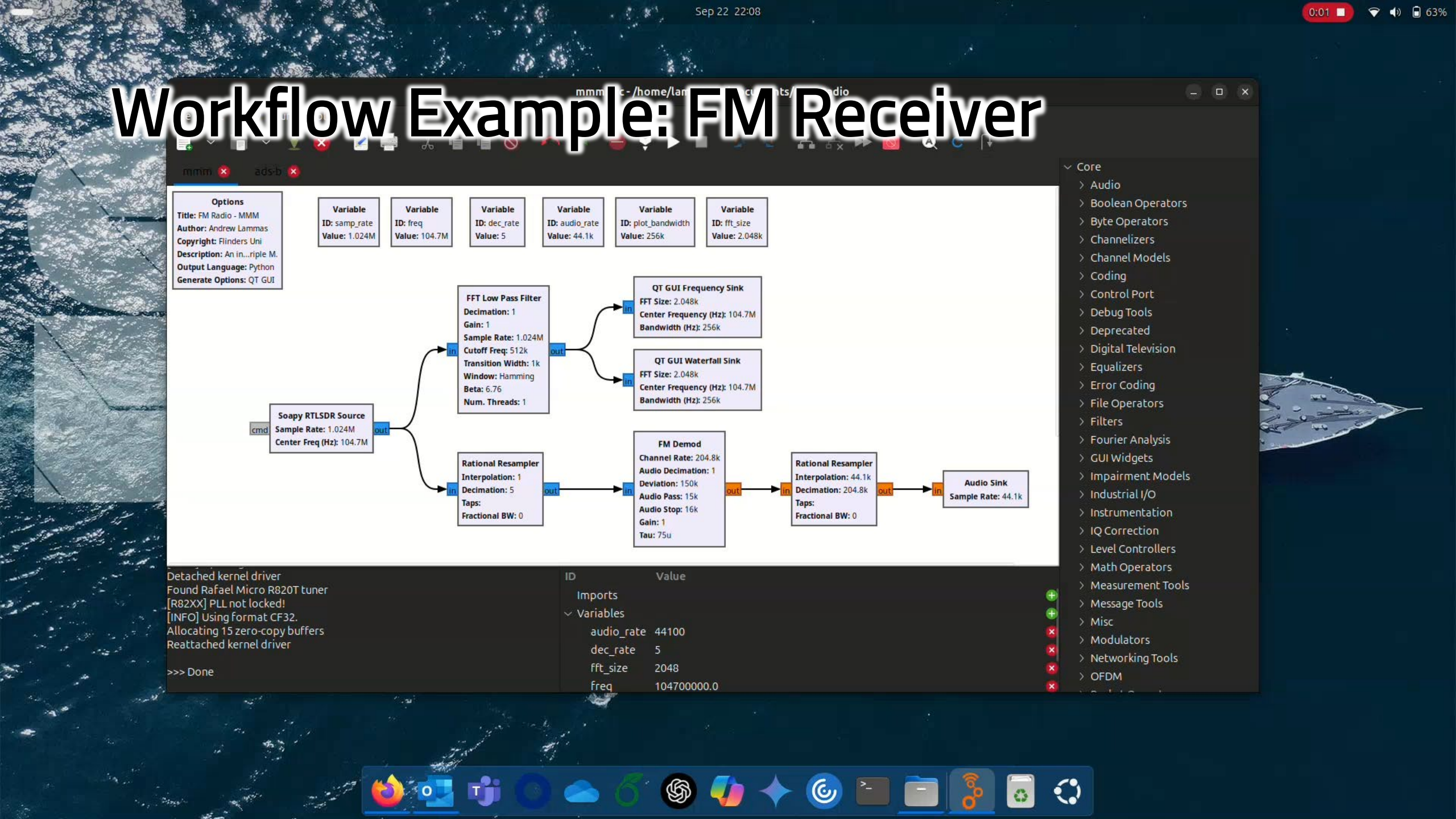
Strengths

- 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



Workflow Example: FM Receiver

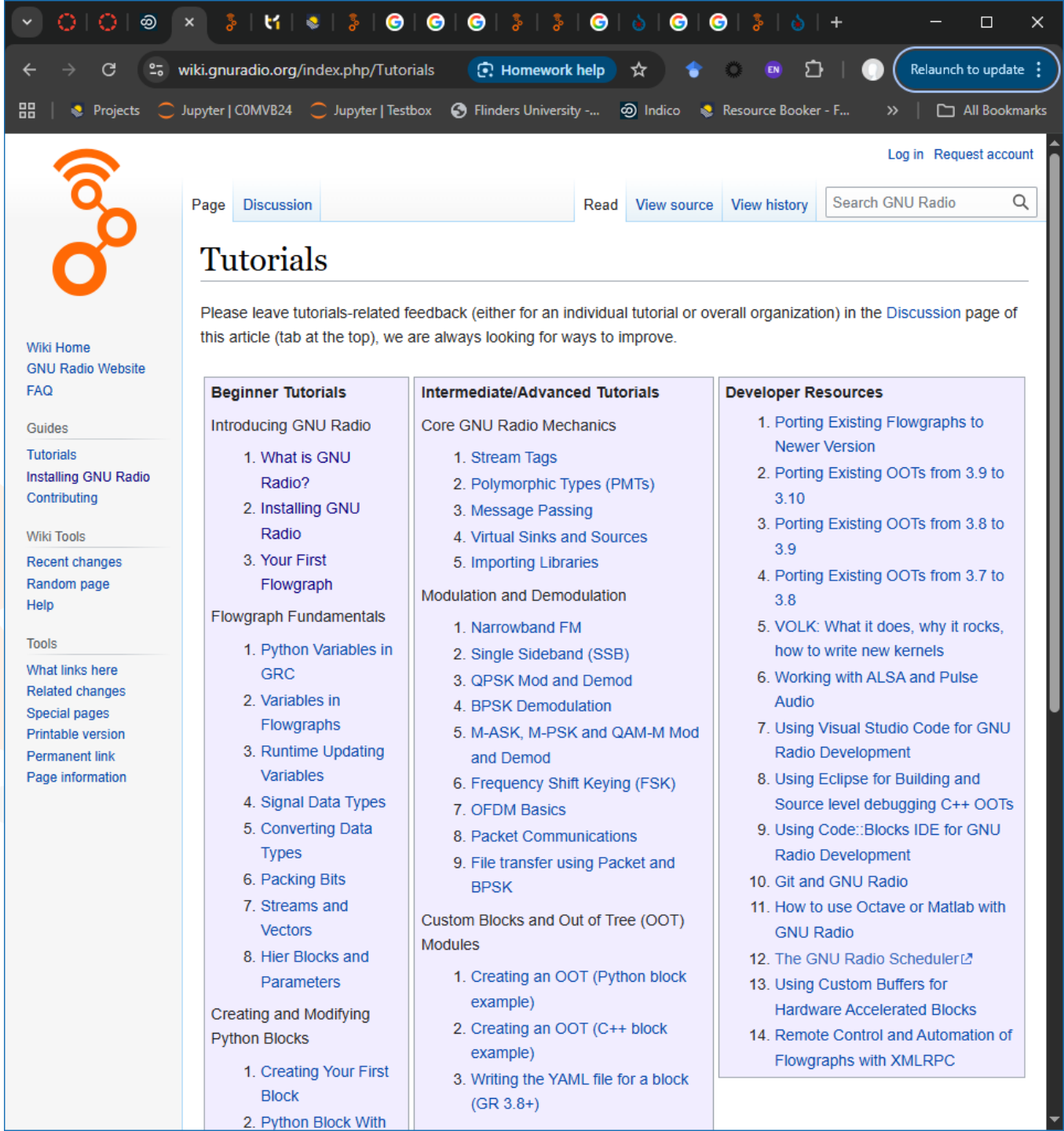


Getting Started

- 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



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