



GNURadio

THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

Welcome!

1st Australian GNU Radio Days

Dr Derek Kozel - dkozel@gnuradio.org

GNU Radio

A framework and set of libraries to build and run digital signal processing applications, primarily software defined radio ones

Started in 2001

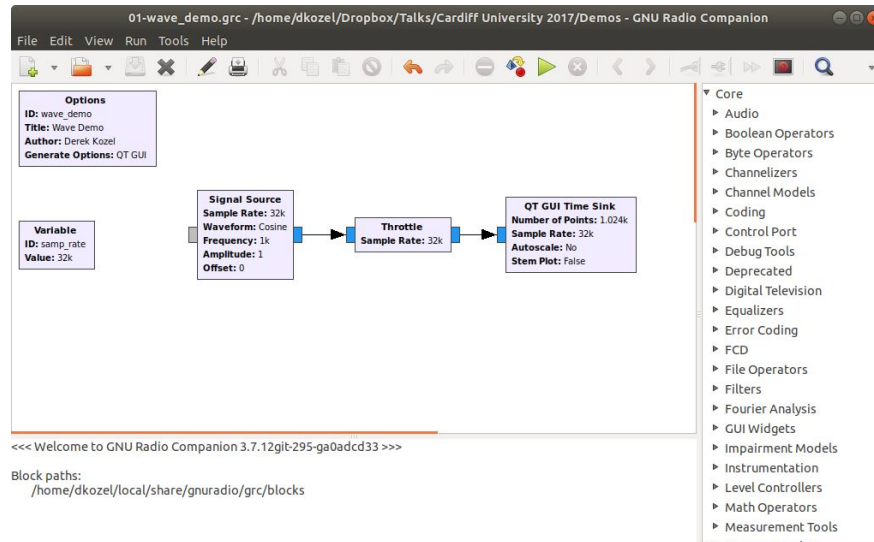
Libre and open source

Written in C++ and Python primarily

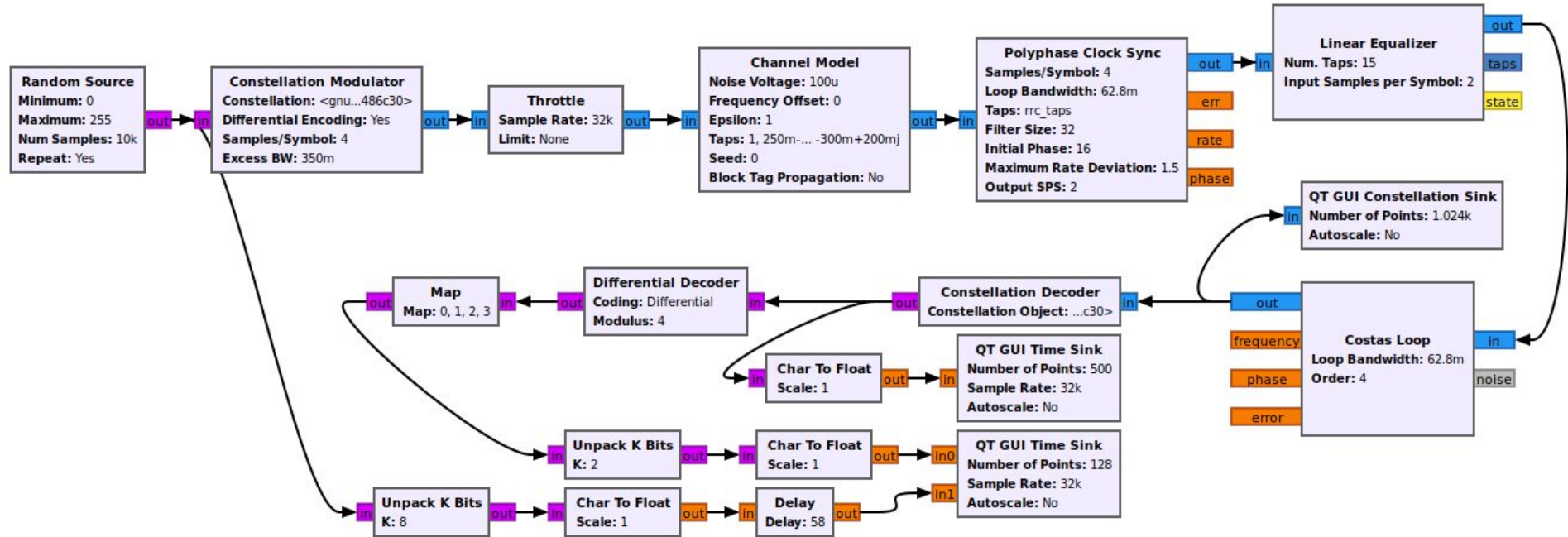
Available on Linux, Windows, and Mac

Used by a very wide variety of users

Commercial, hobbyist, government



QPSK Modulator and Demodulator



mpsk_stage6.grc

Channel Receiver

Noise Voltage

0.3500

Frequency Offset

0.01400

Timing Offset

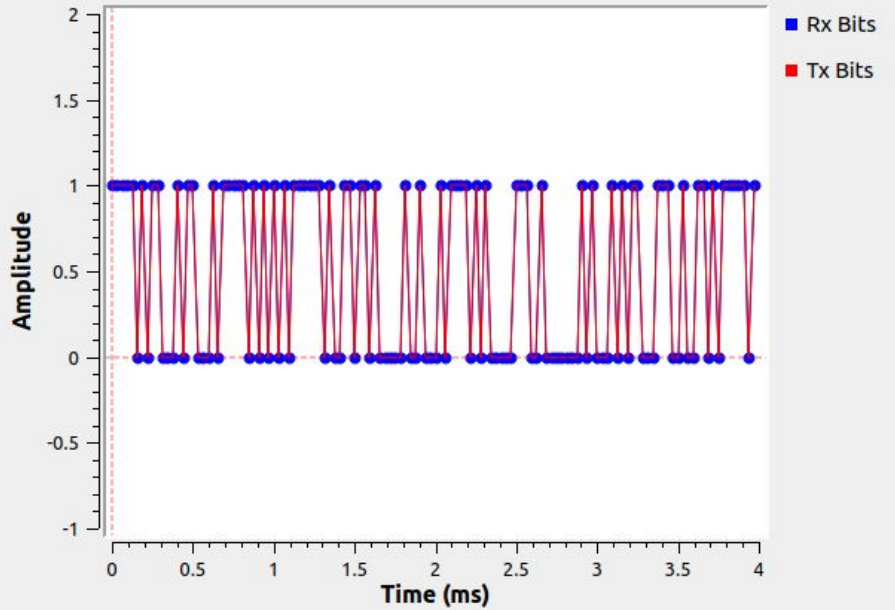
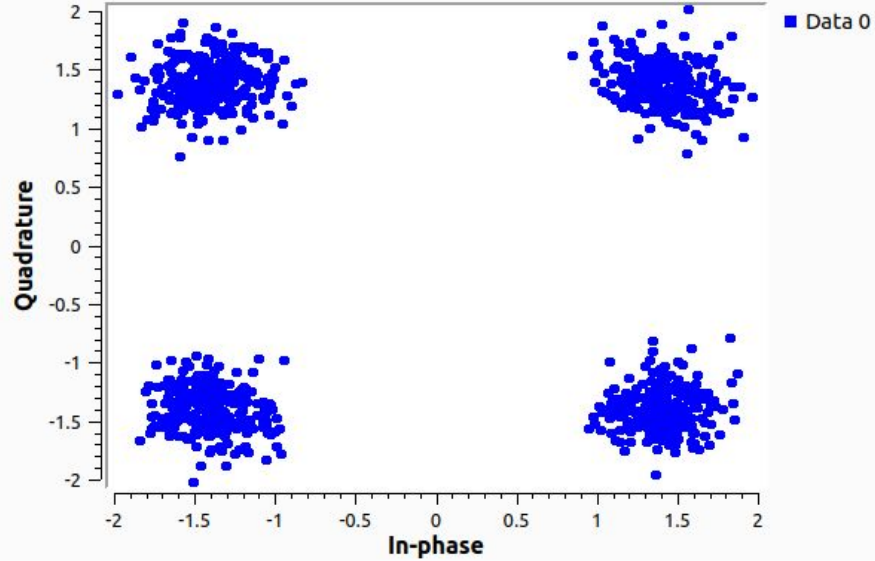
1.000000

Delay

58

Constellation

Symbols

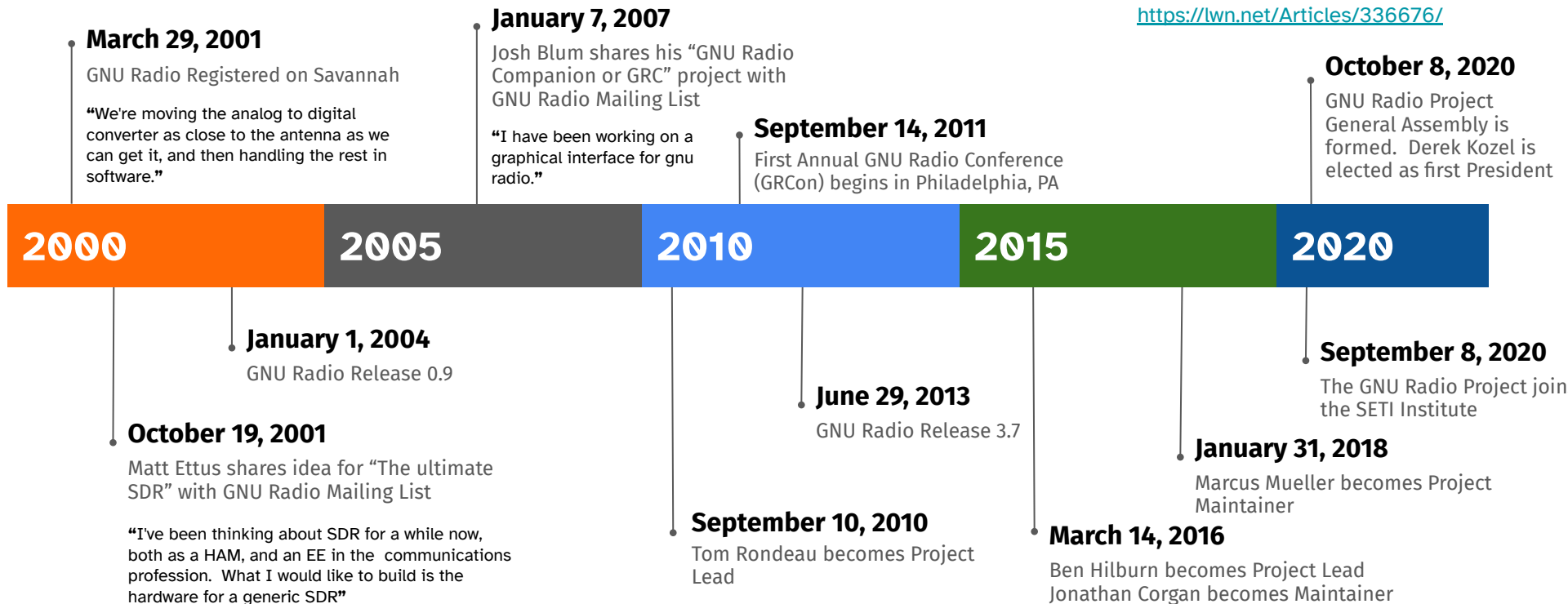


Brief History of GNU Radio

“The GNU Radio project began in 2001 and initial development was funded by John Gilmore and implemented by Eric Blossom. Later, additional work was completed as part of a United States National Science Foundation grant. The aim of the project was to make software radio technology more accessible with lower cost hardware and off-the-shelf computers.”

- Jonathan Corgan

<https://lwn.net/Articles/336676/>



[Discuss-gnuradio] The ultimate SDR

From: Ettus, Matt

Subject: [Discuss-gnuradio] The ultimate SDR

Date: Fri, 19 Oct 2001 16:24:30 -0700

I've been thinking about SDR for a while now, both as a HAM, and an EE in the communications profession. What I would like to build is the hardware for a generic SDR:

- at least 1 (up to 20) Msamples/sec, 4 to 16 bits per sample (obviously fewer bits per with higher rates)

- Tuning in 2 ranges, either from 0 to 1 GHz, or 800 to 2500 MHz

- Transmit and Receive full duplex, on different frequencies

- Connected to PC by a dedicated 100Mbit ethernet line, with raw Ethernet packets (i.e. no TCP, UDP, IP, etc.)

I have a basic design for the RF, analog, and DSP downconversion. Its not that hard with Analog Devices products...

The hard part is the Ethernet interface. Does anyone have suggestions for the ethernet interface? I haven't worked with any Ethernet MACs, nor have I embedded one on an FPGA. I'd rather avoid having an OS on the device, but a [fast] microcontroller might do the job.

The other option is Firewire, but support under linux seems sketchy, and I have no experience with it.

Any ideas? Anyone interested in helping out?

Matt
N2MJI





What is the GNU Radio Project?

The codebase of the core, VOLK, and SigMF

The dozens of supporting projects and infrastructure

The active contributors

The ecosystem of modules, applications, and interfaces

This conference, EU GNU Radio Days, other events

You - everyone that is interested in making open source software radio even better



Where is GNU Radio Used

Wireless Communications

RADAR

4G/5G/6G

Radio Astronomy

Spectrum Monitoring

IOT and Sensors

Space Comms

Particle Accelerators

Education

Amateur Radio

Physics Research

Security Research

Public Safety

Citizen Science

Transportation

Recreation (CtF)

Weather

Medical

RFML

...



Jordi Alamo
@JordiAlamo

Let's go to learn and practice [#sdr](#) [#hamradio](#) [#gnu](#)
[#radio](#) [#urcat](#) [#urc](#)

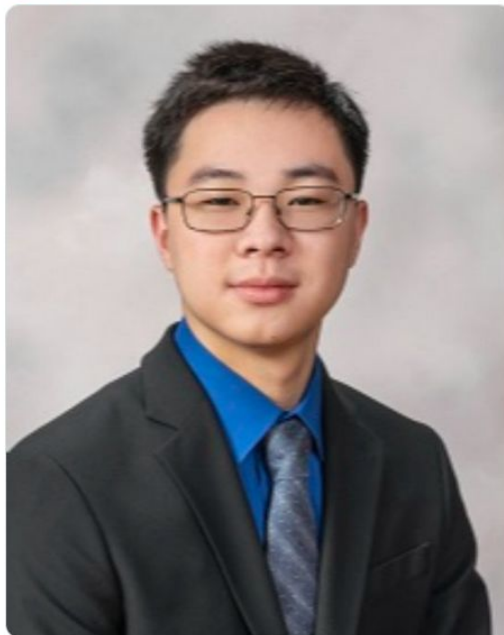


DVSFairs
@DVSFairs

Congrats

Victor Cai, Parkland High School, Allentown, PA
"Designing a Narrowband Radar Using GNU Radio and
Software Defined Radio for Tomography and Indoor
Sensing"

Finalist in [#RegeneronSTS](#), oldest & most prestigious
science & math competition for high school seniors.
[#DVSFairs](#)



Luke Berndt
@LukeBerndt

Want to learn how to send stuff to the cloud from
[@gnuradio](#)? We just put together tutorial on mapping
airplanes using GNU Radio Companion [@Azure](#) &
[@MSPowerBI](#)

[github.com/microsoft/azur...](https://github.com/microsoft/azure-sensor-data)

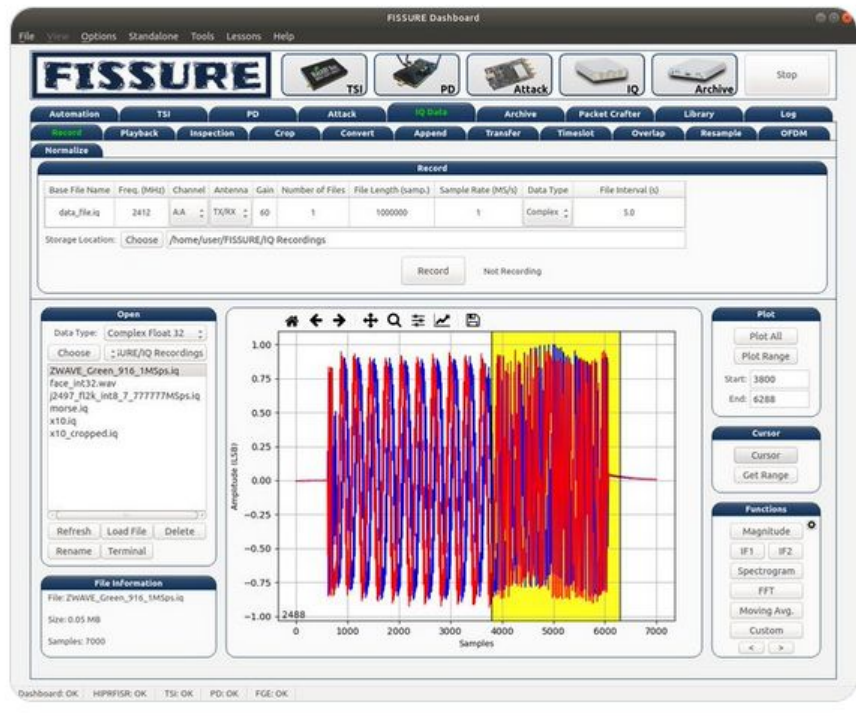
If you give it a try, let me know how it goes!



5:22 PM · Mar 4, 2022 · Tweetbot for Mac

FISSURE - Frequency Independent SDR-based Signal Understanding and Reverse Engineering bit.ly/3BYcz8h

#FISSURE #Gnuradio #WiFi



Hacking the Radio Spectrum with GNU Radio

Talk by Dave Rowntree

Saturday from 6:30 PM - 7:00 PM in [Stage B](#)

ETH zürich

ETH Library

Radio DaCe: A Data-Centric Framework for Software-Defined Radio

Master Thesis

Author(s):
Ilies, Andra-Maria

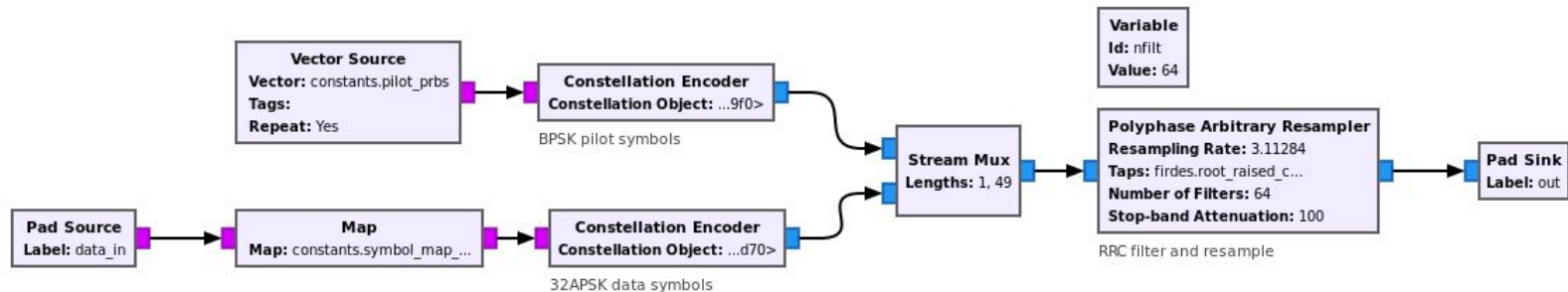
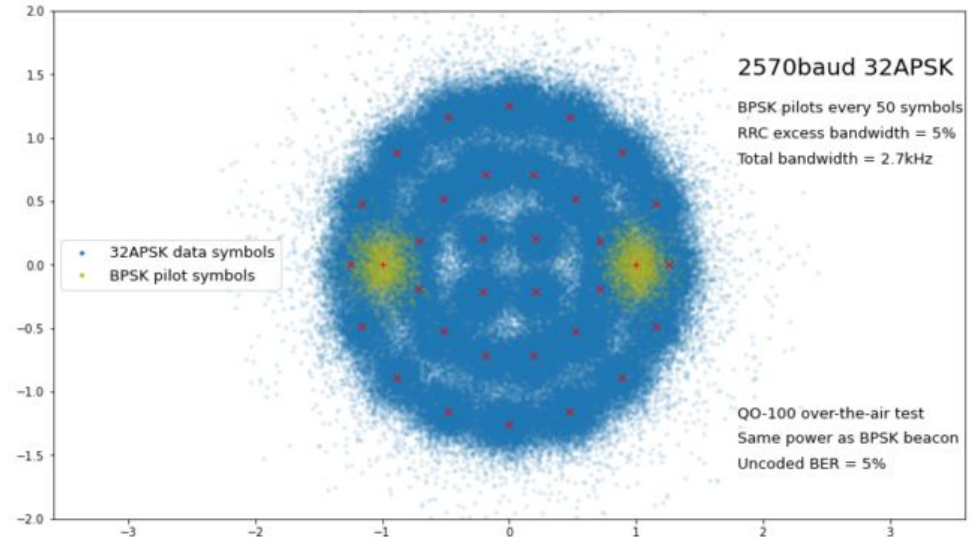
Publication date:
2022

High Speed modem on QO100

Developed by Daniel Estévez EA4GPZ

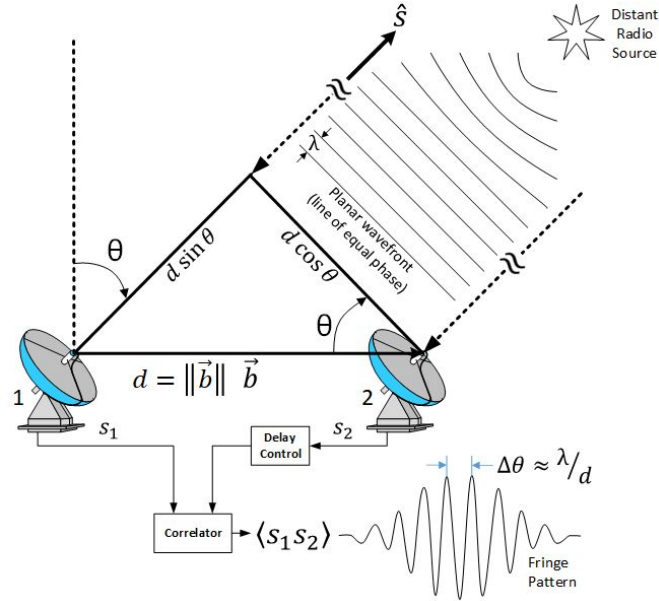
32 APSK, similar to DVB-S2(X)

Expected datarate of ~11 kbps

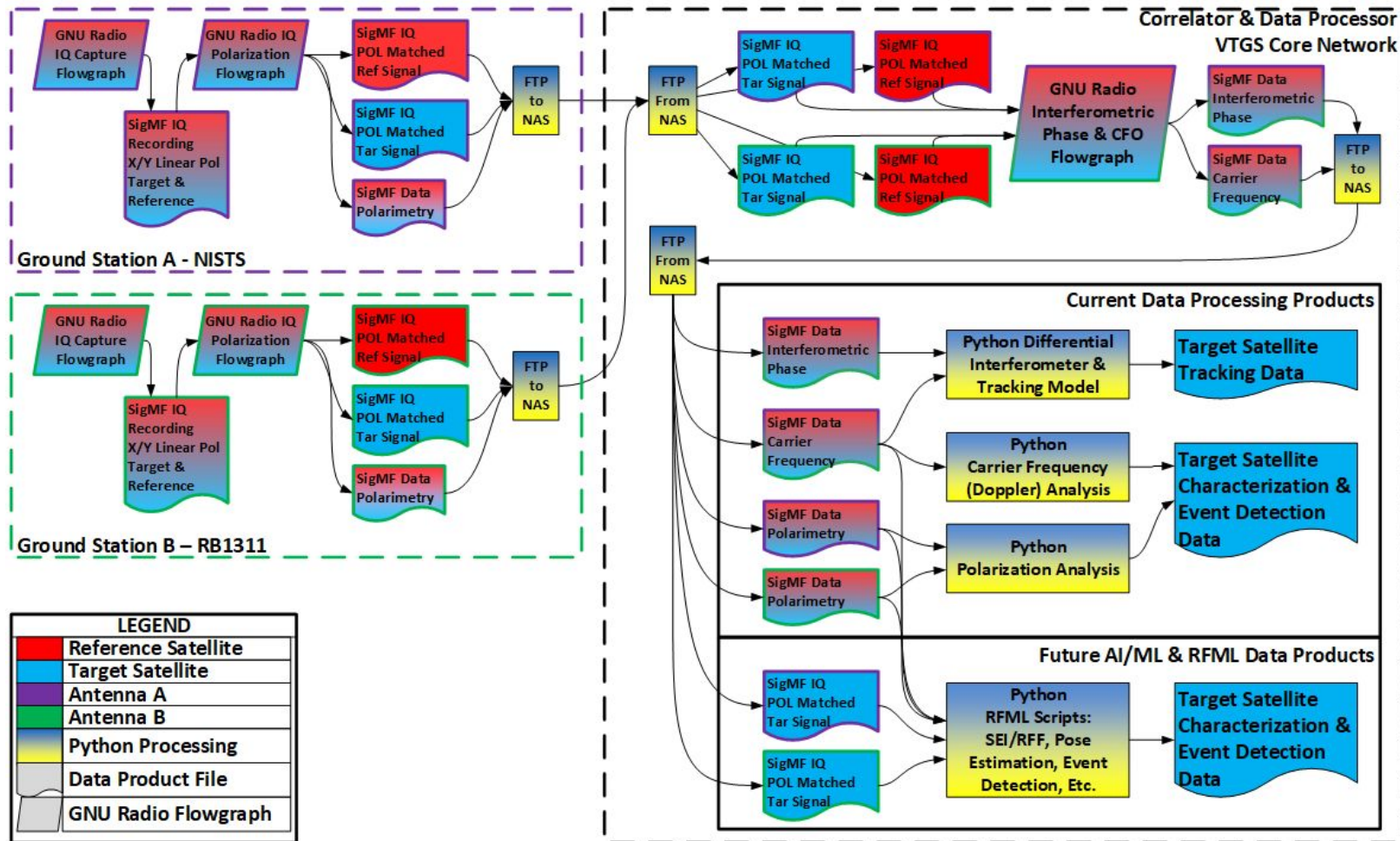




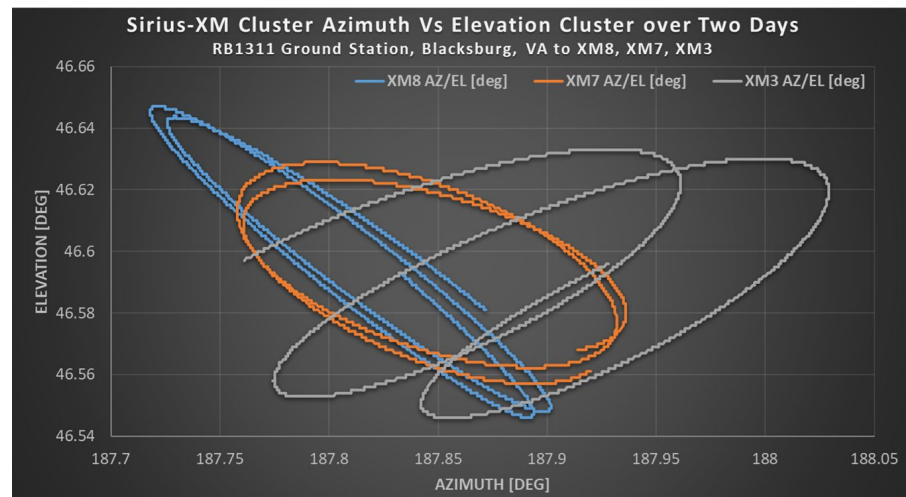
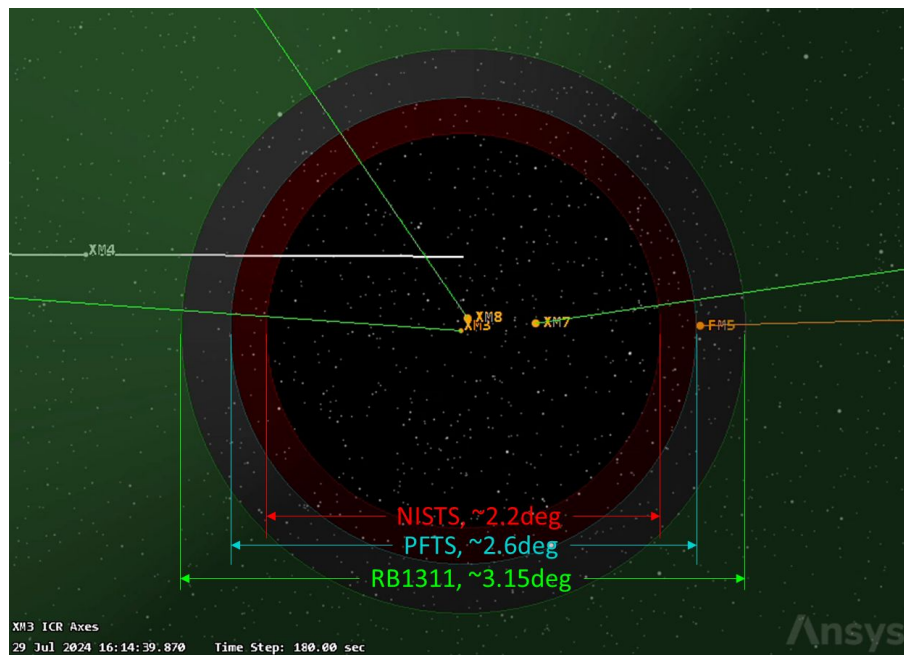
Precision Satellite Tracking



Passive Radio Frequency Techniques & Demonstration for Space Domain Awareness
Zach Leffke, Kevin Schroeder, Matthew Phelps, Justin Fletcher, 2024



Precision Satellite Tracking



How is GNU Radio Governed?

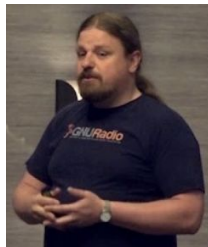


GR BOARD

MARC LICHTMAN - VP



MARCUS MUELLER - VP



JOSH MORMAN - PRESIDENT



DEREK KOZEL - PI

General Assembly

Martin Braun - Community/Ecosystem

Andrej Rode - Infrastructure

Bastian Bloessl - Architecture

Johannes Demel - VOLK

Jeff Long - Maintenance

Seth Hitefield - GRC

Josh Morman - President

Marc Lichtman - Vice President

Derek Kozel - SETI PI

Marcus Müller - Chief Architect / Lead Maintainer

Nate Temple - Community

Philip Balister - Embedded

Samantha Palazzolo - GRCon

Jacob Gilbert - SigMF

Jean-Michel Friedt - Academic Engagement

Ben McCall - Documentation

John Sallay - GR 4.0

Cyrille Morin



Teams

GNU Radio Conference - grcon@gnuradio.org

Architecture - architecture@gnuradio.org

GRC - grc@gnuradio.org

Documentation - docs@gnuradio.org

SigMF - sigmf@gnuradio.org

VOLK - volk@gnuradio.org

Infrastructure - admin@gnuradio.org

Education and Academic Outreach - education@gnuradio.org

Remember - GNU Radio is a completely
volunteer run organizations



GNU Radio Maintenance

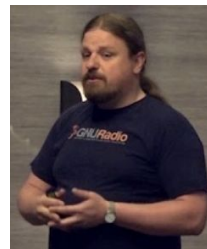
Jeff Long has officially stepped down as maintainer

Marcus Mueller officially has taken on the lead maintainer role

Looking for additional maintainers!

- Be a recognized part of the maintenance team
- Own a module or group of modules (e.g. gr-qt, gr-soapy, gr-zeromq)
- Be responsible for ensuring relevant PRs are reviewed and merged
- Help backport merges to maint-3.10

Email: info@gnuradio.org for more information



Packaging



Installing GNU Radio has gone from headache to trivial over the past several years

Thanks to the packagers in our community



Debian(/Ubuntu) - Maitland Bottoms

RadioConda - Ryan Volz

...

radioconda

```
josh@josh-Alienware-m15-R3:~$ sudo apt install gnuradio
[sudo] password for josh:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  gnuradio-dev libboost-filesystem1.83-dev libgnuradio-dev
  libgnuradio-audio3.10.9t64 libgnuradio-bluetooth3.10.9t64
  libgnuradio-channels3.10.9t64 libgnuradio-complex3.10.9t64
  libgnuradio-dtv3.10.9t64 libgnuradio-fec3.10.9t64
  libgnuradio-filter3.10.9t64 libgnuradio-gpio3.10.9t64
  libgnuradio-network3.10.9t64 libgnuradio-pmt3.10.9t64
  libgnuradio-runtime3.10.9t64 libgnuradio-soapy3.10.9t64
  libgnuradio-trellis3.10.9t64 libgnuradio-uhd3.10.9t64
  libgnuradio-video-sdl3.10.9t64 libgnuradio-wavelet3.10.9t64
  python3-networkx python3-pyqt5.9.0
Suggested packages:
  gqrx-sdr gr-fosphor gr-osmosdr rtl-sdr
  python3-pydot python3-pygraphviz
The following NEW packages will be installed:
  gnuradio gnuradio-dev libboost-filesystem1.83-dev
  libgnuradio-analog3.10.9t64 libgnuradio-audio3.10.9t64
```



Parrot	3.10.3.1
PCLinuxOS	3.10.10.0
pkgsrc current	3.10.12.0
PLD Linux	3.10.11.0
PureOS amber	3.7.13.4
PureOS byzantium	3.8.2.0
PureOS landing	3.10.12.0
Raspbian Oldstable	3.8.2.0
Raspbian Stable	3.10.5.1
Raspbian Testing	3.10.12.0
Rosa 2021.1	3.10.9.2
Rosa 13	3.10.12.0
SlackBuilds	3.10.12.0
SlitTaz Cooking	3.7.13.4
Solus	3.10.12.0
Spack	3.8.2.0
Trisquel 10.0	3.8.1.0~rc1
Trisquel 11.0	3.10.1.1
Ubuntu 14.04	3.7.2.1
Ubuntu 16.04	3.7.9.1
Ubuntu 18.04	3.7.11
Ubuntu 20.04	3.8.1.0~rc1
Ubuntu 22.04	3.10.1.1
Ubuntu 24.04	3.10.9.2
Ubuntu 24.10	3.10.11.0
Ubuntu 25.04	3.10.11.0
Ubuntu 25.10	3.10.12.0
Void Linux x86_64	3.10.11.0
Wikidata	3.10.12.0

Getting Involved

Join Chat - <https://chat.gnuradio.org>

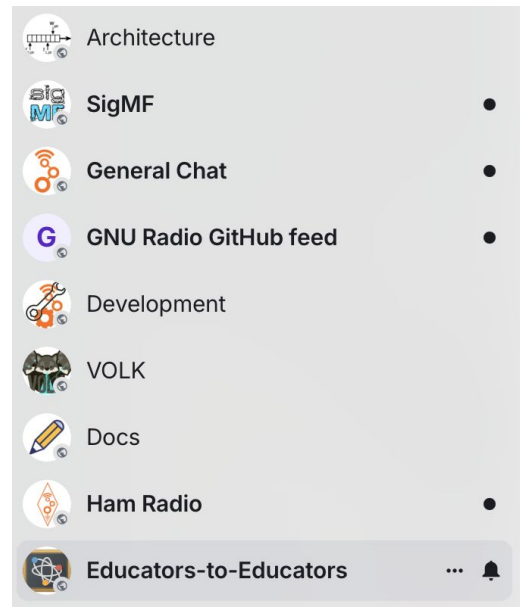
Join the Mailing List - discuss-gnuradio

File issues - <https://github.com/gnuradio>

Review issues, add details, recreate ones

Improve the tutorials - <https://tutorials.gnuradio.org>

Give a talk about GNU Radio - ~1200 publications in 2024 mentioning GNU Radio



Getting Involved

<input type="checkbox"/>	<input type="radio"/> 33 Open <input checked="" type="checkbox"/> 48 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	<input checked="" type="radio"/> Generate/Run fails silently if the GRC flowgraph has not yet been saved Bug good first issue GRC #6185 opened 6 days ago by dkozel						1
<input type="checkbox"/>	<input checked="" type="radio"/> Duplicate "All files" filter entry in save screenshot dialog Bug good first issue GRC #6010 opened on Jul 14 by haakov						9
<input type="checkbox"/>	<input checked="" type="radio"/> Wavfile Sink: Header docstring still only mentions PCM, but we do all of libsnd Bug Documentation good first issue #5971 opened on Jun 26 by marcusmueller						1
<input type="checkbox"/>	<input checked="" type="radio"/> Improve error text when block without a handler receives a message enhancement good first issue logging Runtime #5960 opened on Jun 20 by dkozel						3
<input type="checkbox"/>	<input checked="" type="radio"/> analog/fastnoise_source constructor argument <code>seed</code> type inconsistent with docstring good first issue analog Bug Documentation #5959 opened on Jun 17 by bgottula						1
<input type="checkbox"/>	<input checked="" type="radio"/> Adding node sets <code>state: true</code> disabling and enabling it sets <code>state: enabled</code> good first issue GRC #5779 opened on Apr 22 by nils-werner						2
<input type="checkbox"/>	<input checked="" type="radio"/> Misinformation in fft filter header file Documentation good first issue #5741 opened on Apr 4 by srmnw						1
<input type="checkbox"/>	<input checked="" type="radio"/> Add option to display grid in GRC canvas Feature Request good first issue GRC help wanted UX #5672 opened on Mar 22 by dkozel						



Latest GNU Radio Project Releases



GNU Radio 3.10.12

- GRC-QT Parity
- UHD/RFXNoC
- Other tooling updates

VOLK 3.2.0

- New/improved kernels
- Modernized CI
- Improved tests

SigMF v1.2.5

- Spec fixes and updates

SigMF

Standardized Data Format

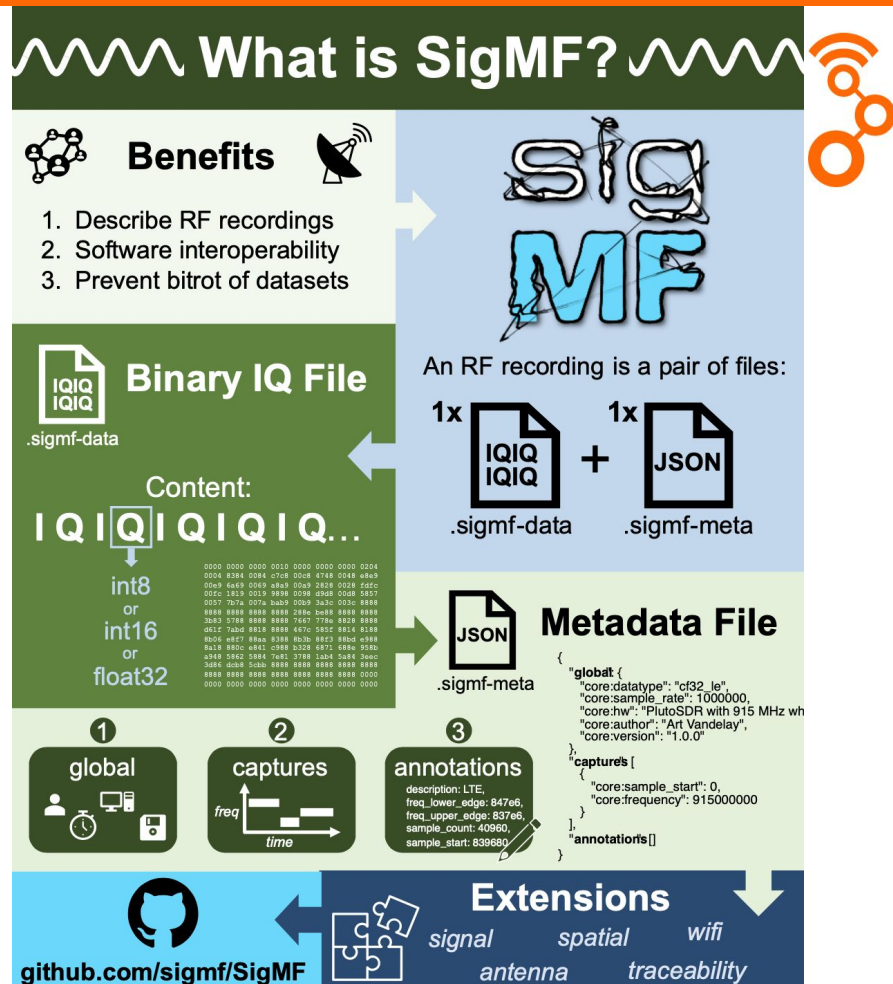
- Common way to describe IQ Data captures with metadata
- Portable format - widely adopted

Training and Evaluation Pipeline

- Unified input format for ML training, validation, testing
- Simplified integration with frameworks like PyTorch

Applications

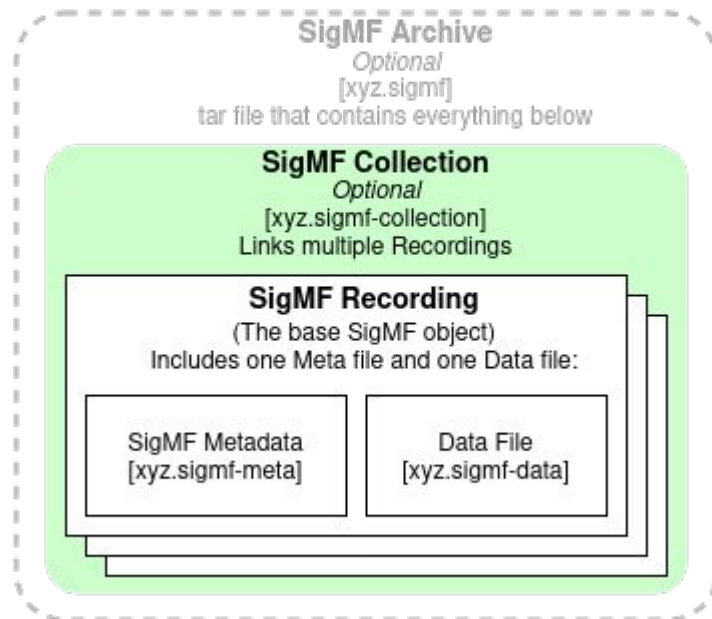
- RFML, Spectrum Sensing, signal identification, ...



SigMF Meta file



```
{
  "global": {
    "core:datatype": "cf32_le",
    "core:sample_rate": 1000000,
    "core:hw": "PlutoSDR with 915 MHz dipole",
    "core:author": "Art Vandelay",
    "core:version": "1.2.0"
  },
  "captures": [
    {
      "core:sample_start": 0,
      "core:frequency": 915000000
    }
  ],
  "annotations": [
    {
      "core:comment": "NID1 = 0",
      "core:freq_lower_edge": 1875986518.88,
      "core:freq_upper_edge": 1877891518.88,
      "core:label": "PSS",
      "core:sample_count": 548,
      "core:sample_start": 147201
    }
  ]
}
```



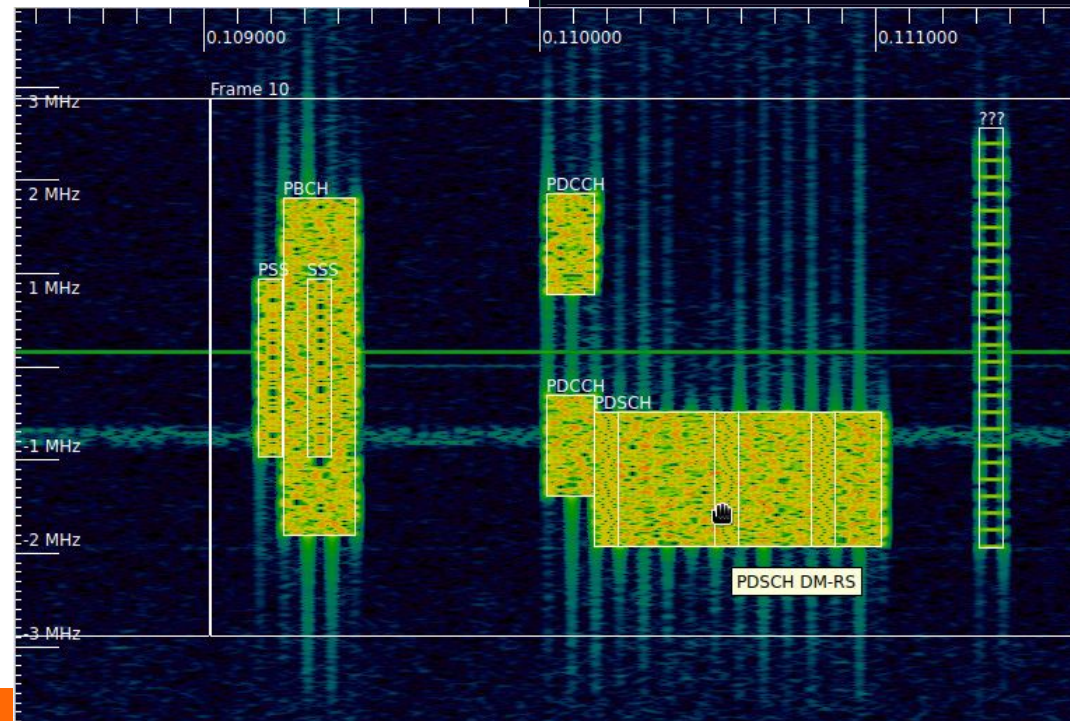
Annotations

Show 10 entries

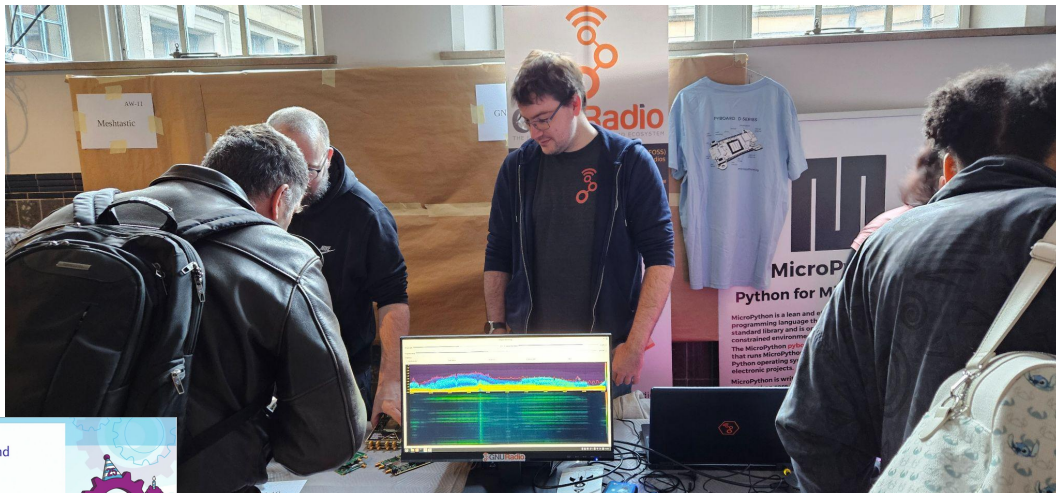
Search 72 records...

Annotation	Frequency Range	BW	Label	Time Range	Comment	Duration	Actions
0	1.874074 GHz - 1.879834 GHz	5.76MHz	Frame 0	2023-07-20T11:39:48.689023568Z - 2023-07-20T11:39:48.699023568Z		10ms	→ ✕
1	1.875986519 GHz - 1.877891519 GHz	1.905MHz	PSS	2023-07-20T11:39:48.689166797Z - 2023-07-20T11:39:48.689238151Z	NID1 = 0	71.354167us	→ ✕
2	1.875146519 GHz - 1.878761519 GHz	3.615MHz	PBCH	2023-07-20T11:39:48.689238151Z - 2023-07-20T11:39:48.689452214Z	NcellID = 1lssb = 0	214.0625us	→ ✕
3	1.875986519 GHz - 1.877891519 GHz	1.905MHz	SSS	2023-07-20T11:39:48.689309505Z - 2023-07-20T11:39:48.689380859Z	NID1 = 0NID2 = 1NcellID = 1	71.354167us	→ ✕
4	1.874074 GHz - 1.879834 GHz	5.76MHz	Frame 1	2023-07-20T11:39:48.699023568Z - 2023-07-20T11:39:48.709023568Z		10ms	→ ✕
				2023-07-20T11:39:48.699166797Z - 2023-07-20T11:39:48.699238151Z	NID1 = 0	71.354167us	→ ✕
				2023-07-20T11:39:48.699238151Z - 2023-07-20T11:39:48.699452214Z	NcellID = 1lssb = 0	214.0625us	→ ✕
				2023-07-20T11:39:48.699309505Z - 2023-07-20T11:39:48.699380859Z	NID1 = 0NID2 = 1NcellID = 1	71.354167us	→ ✕
				2023-07-20T11:39:48.709023568Z - 2023-07-20T11:39:48.719023568Z		10ms	→ ✕
				2023-07-20T11:39:48.709166797Z - 2023-07-20T11:39:48.709238151Z	NID1 = 0	71.354167us	→ ✕

Prev Next



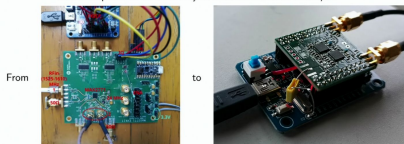
FOSDEM 2025



Efficient USB communication under GNU/Linux for a wideband
(MAX2771-based) L-band (GNSS) SDR receiver

J.-M. Friedt

FEMTO-ST Time & Frequency, Besançon, France
associate professor at University of Franche-Comté in Besançon



February 1, 2025

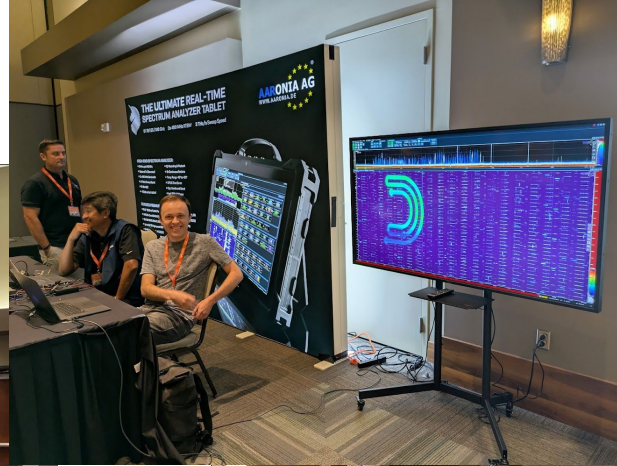
FOSDEM 25



EU GR Days 2025



US GNU Radio Conference 2025



SETI Hackathon

Sponsored by ARDC

Allen Telescope Array



SETI ARISE

agiseti.com

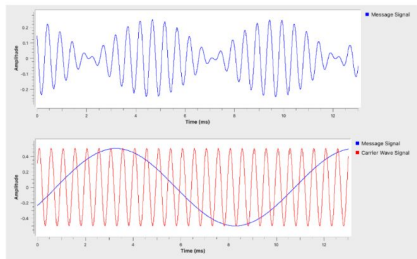


News

SETI Institute Launches ARISE Lab, Bringing SETI and Radio Astronomy to Community Colleges

Possible Types of Extraterrestrial Intelligence Radio Transmissions

In this lab, students dive into the field of SETI (Search for Extraterrestrial Intelligence) by exploring the types of radio transmissions that may be detected from extraterrestrial sources. This module covers essential concepts in signal processing, including the Nyquist sampling theorem, filtering techniques, and modulation. Students will examine various possible SETI signal types, such as beacons, leakage signals, narrowband vs. wideband signals, as well as continuous wave and pulsed transmissions. The hands-on component involves designing and implementing an AM transmitter and receiver, allowing students to apply their understanding of signal characteristics and modulation. By the end of the lab, students will gain practical experience in signal processing while considering the broader implications of detecting signals from extraterrestrial intelligence in SETI research.



Lecture Notes

Pre-Lab Reading

Teaching Resources

Communication Systems Engineering with GNU Radio: A Hands-on



Approach 1st Edition

by [Jean-Michel Friedt](#) (Author), [Herve Boeglen](#) (Author)

[See all formats and editions](#)

✓ **Pre-order Price Guarantee.** [Terms](#)▼

An approachable guide to an invaluable radiofrequency communication toolkit

Software-defined radio (SDR), which emerged in the 1990s, has become a core development method in certain high-profile fields, including military and space communications. High cost and problems with hardware availability, however, prevented this technology from being widely disseminated. The advent of low-cost hardware beginning in the 2010s, however, has made GNU Radio—the leading open-source software toolkit for developing SDR systems—an increasingly viable and even critical tool for a new generation of radiofrequency communication engineers.

Communication Systems Engineering with GNU Radio provides an accessible overview of this toolkit and its applications. Beginning with the fundamentals of using GNU radio for digital signal processing, the volume then moves to the practicalities of decoding data and the advantages of accessing raw data normally unavailable in hardware-defined radiofrequency receivers. The result is a potentially crucial tool for engineers looking to adopt this cost-effective and flexible standard for transmitting and processing radiofrequency signals.

Readers will also find:

- A careful balance of radio communications theory with GNU Radio practicalities
- Practical implementation examples employing well-developed open-source GNU Radio platforms
- Extensive accompanying documentation and explanation

Communication Systems Engineering with GNU Radio is ideal for graduate and undergraduate students in communications systems courses, as well as professionals working in SDR.

Available now!

<https://a.co/d/cXoltKj>

COMMUNICATION SYSTEMS ENGINEERING WITH GNU RADIO

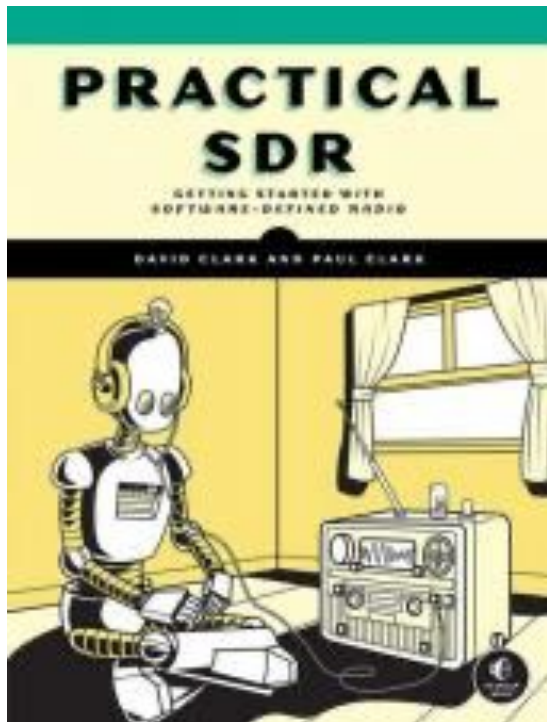
A HANDS-ON APPROACH

JEAN-MICHEL FRIEDT • HERVÉ BOEGLÉN



Practical SDR - No Starch Press

<https://a.co/d/9QlvQMv>



Introduction

Part I: Building a Basic Receiver

Chapter 1: What Is a Radio?

Chapter 2: Computers and Signals

Chapter 3: Getting Started with GNU Radio

Chapter 4: Creating an AM Receiver

Part II: Inside the Receiver

Chapter 5: Signal Processing Fundamentals

Chapter 6: How an AM Receiver Works

Chapter 7: Building an FM Radio

Part III: Working with SDR Hardware

Chapter 8: The Physics of Radio Signals

Chapter 9: GNU Radio Flowgraphs with SDR Hardware

Chapter 10: Modulation

Chapter 11: SDR Hardware Under the Hood

Chapter 12: Peripheral Hardware

Chapter 13: Transmitting

Matrix Chat room for Educators



chat.gnuradio.org

education@gnuradio.org



Educators-to-Educators

#edu:gnuradio.org



Public room



Not encrypted

Discussing how to use GNU Radio in
education



Project Vision

STRENGTHEN

Ongoing improvement of the core framework

SPREAD

Grow and expand the reach of the project

SUSTAIN

Keep the ecosystem healthy and active

GNU Radio Funding



→ **GNU Radio Conference**

→ **Grants**

→ **Donations**

→ ...

→ **Sustainability**

Google Summer of Code

This year we had 4 sponsored projects!

Thanks to Google, the students, and the mentors



Contributor	Mentor
<Aaditya/>	Michael Petry
5G CELL SCANNER	
This project develops an open-source 5G cell scanner utilizing GNU Radio and SDR hardware to passively detect and decode 5G signals, extracting key...	
View project details	
Contributor	Mentor
Daniel Paul	Marcus Müller
GNU Radio FT8/WSPR OOT Module	
This project aims to develop a comprehensive Out-of-Tree (OOT) module for GNU Radio that enables amateur radio enthusiasts to incorporate FT8 and...	
View project details	
Contributor	Mentor
KrishGupta	Josh M
Expanding the GNU Radio 4.0 Block Set	
I propose to migrate and enhance critical signal-processing blocks from GNU Radio 3.x to the next-generation GNU Radio 4.0 architecture. Leveraging...	
View project details	
Contributor	Mentor
StudHamza	noc0lour
FM Broadcast Radio application	
In this project, I aim to develop a fully fledged FM broadcast receiver application with integrated RDS and spectrum scanning. The application should...	
View project details	



Fund your projects/ideas through grants

Small Grants through GNU Radio - \$5-\$15k
Nov. 15 Deadline - info@gnuradio.org

Grant Ideas

Similar to the [GSoC Ideas](#) page, this is a list of things we could potentially ask for grant money to accomplish, or even just volunteers who want something specific to dive into:

Contents [\[hide\]](#)

- 1 [Documentation Related](#)
 - 1.1 [Doxygen Cleanup](#)
 - 1.2 [More Tutorials!](#)
 - 1.3 [Filling out Block Docs](#)
 - 1.4 [Coming up with a system for exporting and versioning wiki](#)
- 2 [Training Materials Related \(Not including tutorials in our docs\)](#)
- 3 [Packaging](#)
 - 3.1 [Windows & OS X Packaging](#)
 - 3.2 [CI-generated Packages](#)
 - 3.3 [Maintainer Assistance](#)
- 4 [GNU Radio 4.0 and Beyond](#)



AMATEUR RADIO DIGITAL COMMUNICATIONS

The NumFOCUS Small Development Grants program is a community collaboration which addresses project needs while also engaging dedicated volunteers. See the process below:



How to Support GNU Radio

Get Involved

Advertise that you use GNU Radio

Write a letter of support

Publish a whitepaper

Join a team

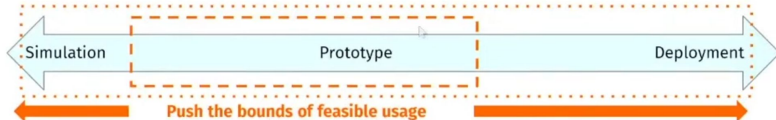


Technical vision remains unchanged

... but is starting to become reality

From GRCon22

GNU Radio in the development workflow



Through the 4.0 architecture, expand feasible usage from simulation through to deployment

Technical Vision for GNU Radio 4.0

Modular CPU Runtime

- Scheduler as plugin
- Application-specific schedulers

Heterogeneous Architectures

- Seamless integration of accelerators (e.g., FPGAs, GPUs, DSPs, SoCs)

Distributed DSP

- Setup and manage flowgraphs that span multiple nodes

Straightforward implementation of (distributed) SDR systems that make efficient use of the platform and its accelerators

GR 4.0 Timeline



2019: Benchmarking of GR scheduler, identification of optimization opportunities. Attempts to implement in current codebase

2020: SDR 4.0 - enhance GNU Radio capabilities on to heterogeneous compute platforms

2020: newsched project - fresh start on architectural concepts

2022: GRCon22 - newsched modularity and usability demonstration

2022: GSI/FAIR commits resources and takes on deeper changes to scheduler and API

2024: EU GR Days - @GSI/FAIR hands on workshops

2024: EU GR Days - Demonstration of packet modem in GR 4.0

Permissive Licensing



Our goal is for free and open software defined radio to get into the hands of as many people as possible.

GNU Radio is boxed out of many ecosystems because of (often misunderstood) GPL constraints

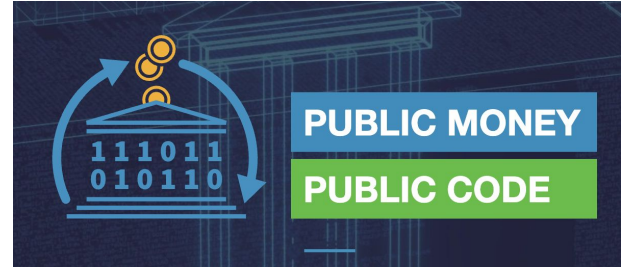
GR4 will have a permissive MIT licensed core and minimal blockset

Allows derived works to be licensed *at the discretion of the author*

- We envision a healthy GPLv3 ecosystem of OOT modules

Gain access to new applications / markets / ecosystems

Note: no relicensing of GR3 - ported code will remain GPLv3



The future of the GNU Radio framework



Imagine ...

GNU Radio that is suitable for mission critical and enterprise grade applications

GNU Radio flowgraphs that can be designed as prototypes and be deployed as products with minimal additional effort

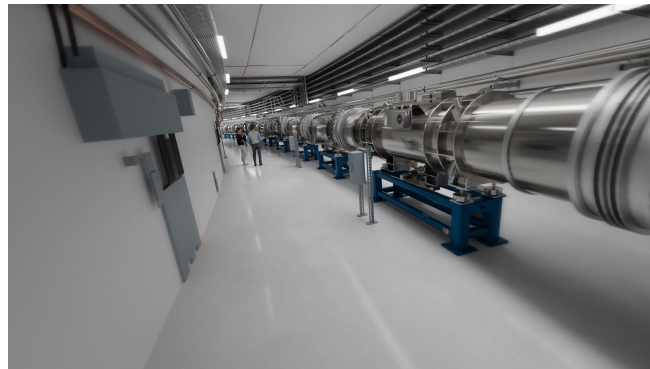
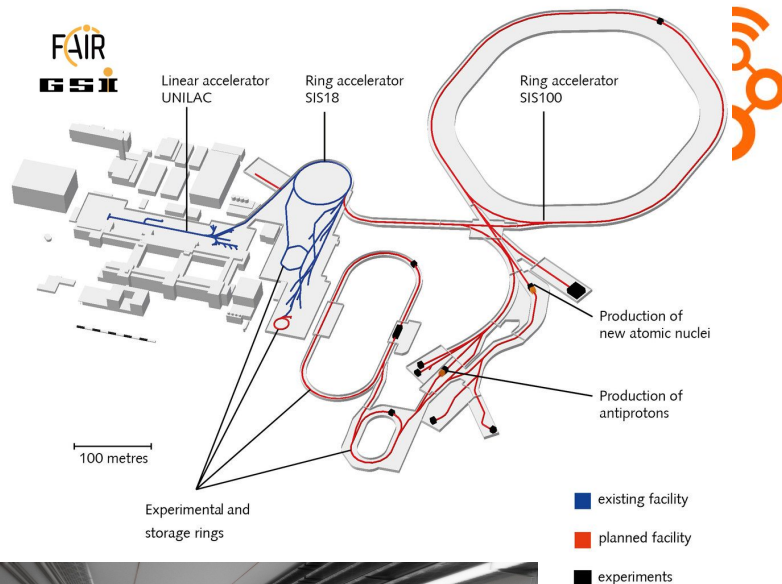
GNU Radio development workflows and data pipelines that better enable AI tools and AI/ML applications

Mission Critical

GR4 was developed as the measurement and sample processing solution for the particle accelerators at FAIR

Designed to run on **critical infrastructure**, and thus *meet higher standards for **safety**, **cybersecurity**, and **product liability**.*

- Redesigned from the ground up
- Modern C++ and Best Practices
- Lean, clean codebase
- High throughput and low latency demands





Finland



France



Germany



India



Poland



Romania



Russia



Slovenia



Sweden



UK





From prototype to deployed product

Block based development is a powerful concept - rapid application development based on modularity, reuse - but how well can a general purpose framework allow this to become a “product”

How these blocks get executed is highly dependent on the runtime framework

- Optimal scheduling across available compute resources
 - Custom scheduler to meet application needs
- Heterogeneous compute
 - Custom buffers to suit specific compute resources
- Maximize CPU performance
 - Lock Free buffers, constexpr optimization, std::simd standardization, Block Merging

GR4 Killer Features

Lean on compiler for efficient optimizations

Type strictness and constraints

Code is single source of truth

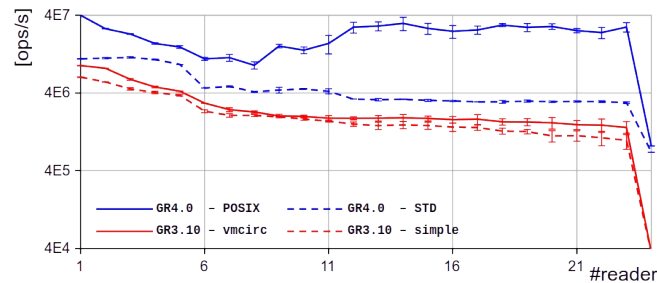
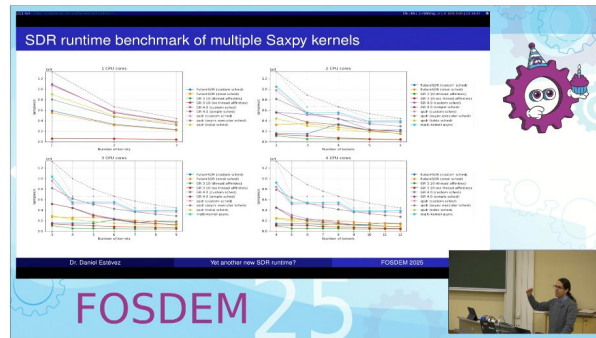
SIMD and Merge API

Direct testing of blocks without flowgraph

Lock free circular buffers

Modular Scheduling

Ease of Development



benchmark:		cache misses	mean	stddev	max	ops/s
merged src->sink		1.3k / 3k = 46%	626 ns	110 ns	952 ns	16.4G
merged src->copy->sink		391 / 971 = 40%	957 ns	106 ns	1 us	10.7G
merged src(N=1024)->b1(Ns128)->b2(N=1024)->b3(N=32...128)->sink		398 / 960 = 41%	957 ns	103 ns	1 us	10.7G
merged src->mult(2.0)->div(2.0)->add(-1)->sink		401 / 1k = 40%	3 us	108 ns	4 us	3.0G
merged src->(mult(2.0)->div(2.0)->add(-1))^10->sink		470 / 1k = 42%	41 us	189 ns	42 us	248M
runtime src->sink		9k / 174k = 5%	42 us	98 us	336 us	241M
runtime src(N=1024)->b1(Ns128)->b2(N=1024)->b3(N=32...128)->sink		20k / 648k = 3%	125 us	328 us	1 ms	81.7M
runtime src->mult(2.0)->div(2.0)->add(-1)->sink - process_one(..)		24k / 663k = 4%	105 us	259 us	882 us	97.5M
runtime src->mult(2.0)->div(2.0)->add(-1)->sink - process_bulk(..)		24k / 664k = 4%	152 us	358 us	1 ms	67.3M
runtime src->(mult(2.0)->div(2.0)->add(-1))^10->sink		56k / 686k = 8%	127 us	28 us	198 us	80.6M



GR4 in practice

This is a completely functional GR4 block

```
struct Square : Block<Square> {  
    using Description = Doc<"@brief Squares the input value">;  
  
    PortIn<float> in;  
    PortOut<float> out;  
  
    GR_MAKE_REFLECTABLE(Square, in, out);  
  
    [[nodiscard]] constexpr float processOne(float input) const noexcept { return input*input; }  
};
```



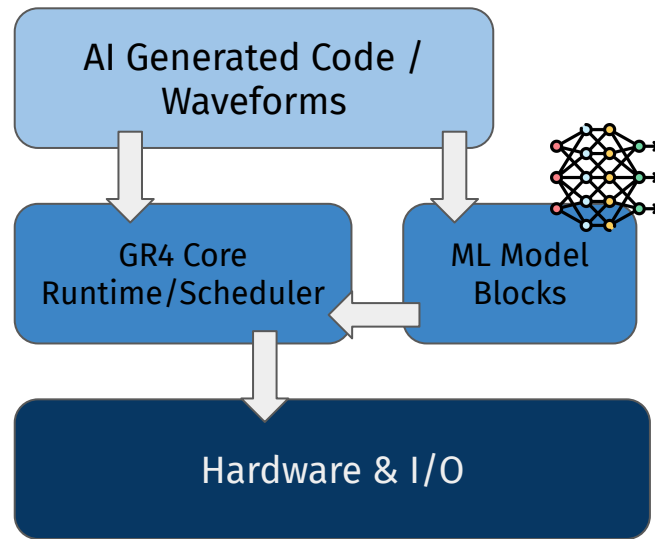
GR4 as a low barrier entrypoint for AI enabled SDR

AI-enabled SDR starts with an easy to use, high-performance framework.

Without a stable, performant core, developers waste time reinventing DSP chains, schedulers, and I/O handling.

GR4 provides a production-grade signal processing engine so AI-generated code and AI-driven waveforms can focus on innovation, not reimplement.

A shared foundation means AI models and outputs are immediately deployable and interoperable with other SDR components.





AI Ready Data Model

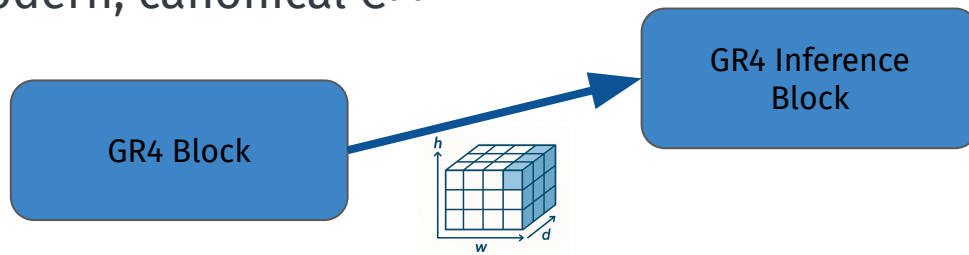
PMT Tensors: superset of vectors with shape, dtype, rank.

Fits naturally with PyTorch, TensorFlow, ONNX workflows.

Rich metadata (time/frequency/antenna) travels with data → training & inference ready.

PMT typed ports: simplify sending structured data between blocks

More sane PMT API - modern, canonical C++





Data Movement and Compute Placement

Custom buffers and zero-copy GPU/accelerator paths.

- Custom Buffer architecture in place
- SYCL as one prototyped option for compute abstraction

Modular schedulers allow a vast ecosystem of custom and future domain-specific schedulers

- Async & batch execution to match AI inference engine behavior.

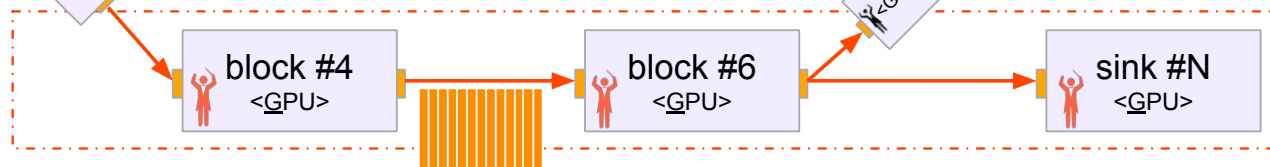
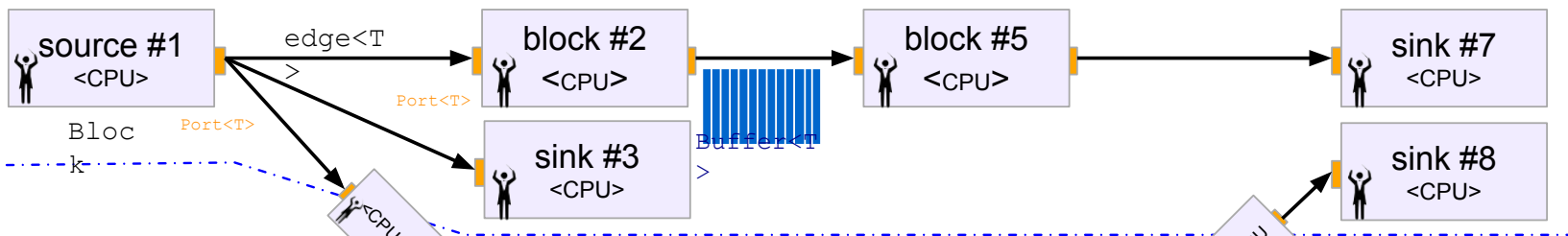


Graph-Based Signal-Flow Description

GR3.x→4: multiple compute domains & inverted scheduler paradigm Block → Graph

flow-graph (global scheduler)

sub-flow-graph: e.g. CPU scheduling domain



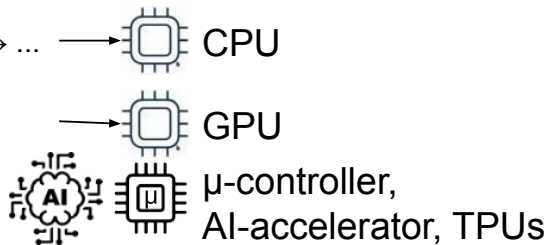
sub-flow-graph: e.g. GPU scheduling domain

flow-graph → schedule



sink#3:work() → block#2 → block#5 → ... → CPU

scheduler#2 → block#4:work() → block#6 → ...





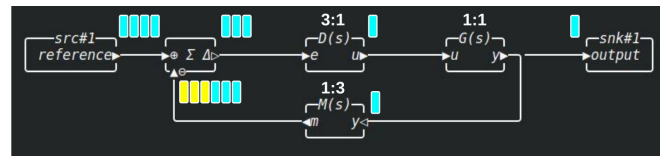
Latest GR4 Features

New/properly integrated features

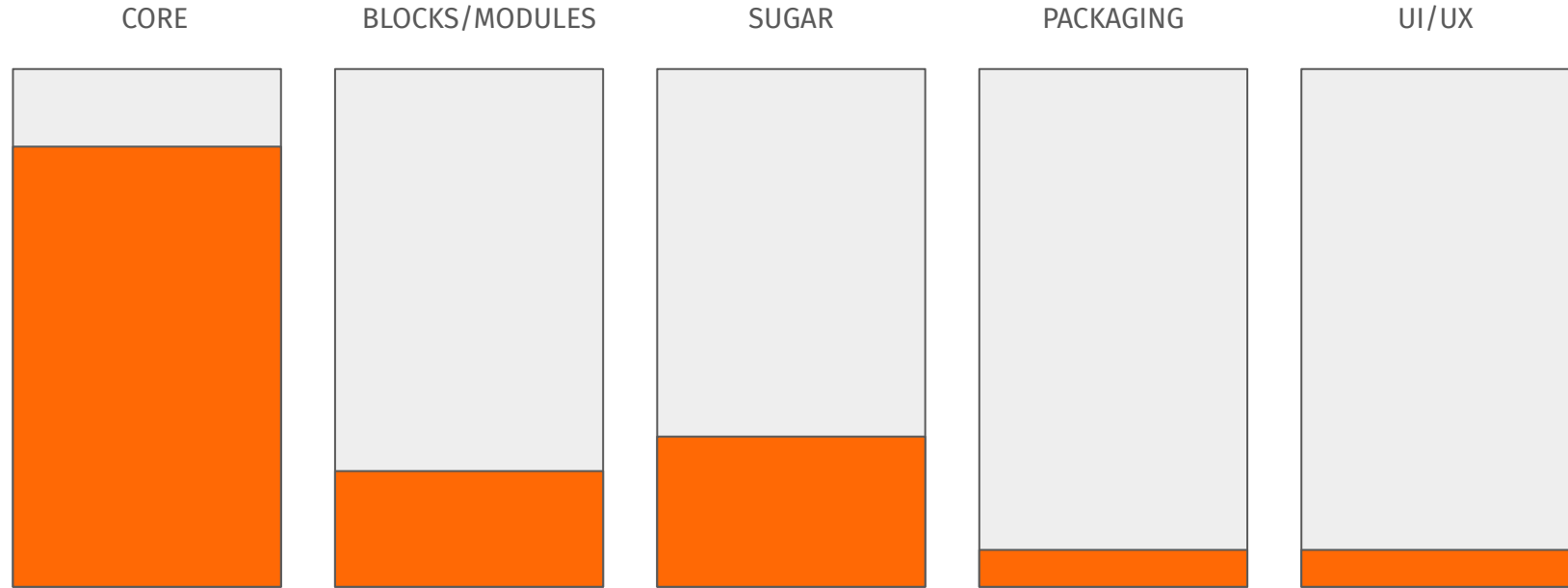
- custom user-defined scheduler
- low-latency schedulers ($< \sim 10$ us worst-case latencies, chain of 5 blocks)
- message passing API to read/write/modify graphs from a non-C++ context (important for GUI developments)
- fully compile-time static/runtime polymorphic (plugin) block API
- unmanaged + managed (WIP) polymorphic sub-graphs (plugin-API)
- ... finalising the Buffer, Port, Block, Graph, Scheduler API, ...
- ... reduced the overhead for tag-processing (~ 35 M tags/s possible)
- ... feedback-loop integration (WIP)
- new `<simd>-FFT` (MIT-licensed, as-fast/faster than FFTW)

Tentative proof-of-concepts:

- tentative Windows support (Chris Gorman et al.)
- SYCL integration (GPUs, FPGAs, ...)
- ONNX integration (vendor-neutral ML integration) -> needs user-driven examples
- `Tensor<T>` support (John Sallay et al.)



Goals for GR4 - to be broadly adoptable





Low Hanging Fruit

What are some things that the community could jump on *right now*

Blocks / Modules

- Porting things from GR3
- SigMF
- ONNX, other AI model frameworks

REST/HTTP Integration

Example flowgraphs - AM/FM/Digital modulation/demodulation

Performance Benchmarks



How to Engage

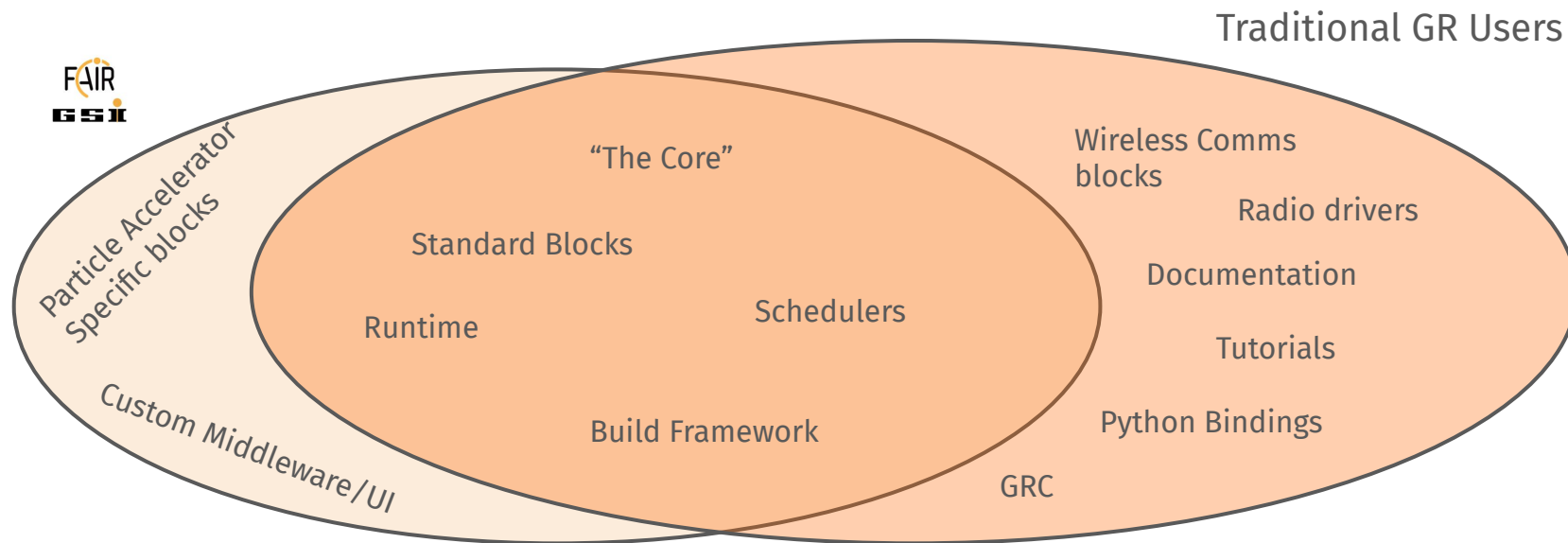
Architecture Working Group -

- This is where we discuss what is going on with GR4
- 4th Thursday of every month (mostly) - 12PM ET
- <https://groups.io/g/gnuradio-scheduler>
 - To get meeting invites
- #Architecture Channel on chat.gnuradio.org
- Topics:
 - We would like to have more community driven topics - show what you are doing with GR4, issues you might be running into, how you would like to use it
 - Apart from that general updates and feature planning

Community Maintainership



Goal is to transition ownership / maintenance to the GNU Radio community



Let's Make this a Reality!



Reach a level of community maintainership

Expand the blockset (join the GR4 block tutorial tomorrow, and hackfest Friday)

Create new and exciting applications that demonstrate the functionalities and performance advantages





Contact

- **`dkozel@gnuradio.org`**
- **Matrix: `dkozel:gnuradio.org`**
- **Bluesky: `@derekkozel.com`**