Maintainer's Update

How soon is now?

September 11, 2025[1]

[1] Version 2025-09-11 04:25:18Z

GRCon25

Outline

1. Introductory words
2. What's up with GNU Radio 4?
3. What's up with GNU Radio 3?

$(whoami)

- Marcus Müller

mmueller@gnuradio.org

- GNU Radio board member
- Maintainer 2018-2021
- With GNU Radio since around 2009

engineering@baseband.digital

- Freelancing GNU Radio consultant
- Workshops, Rent-a-DSP-Engineer, will talk about signal processsing for hours
- Contract SDR development

What's up with GNU Radio 4?

- In short: **We (I) believe it's the future!**
- At some point

  "I did it with GNU Radio"

  should mean

  "I did it with GNU Radio 4"!
- Project priority:

  Get GNU Radio 4 to be the platform on which the community runs
- Constraint: **Don't lose the community on the way.**
- Thus: Achieve *functional attractiveness* comparable to GNU Radio 3

GRCon25

If you believe in GNU Radio 4, what happens to GNU Radio 3?

We continue to maintain GNU Radio 3, for now.

- Porting things from "old" to "new" is hard when "old" refuses to build
- Adoption of GNU Radio 4 isn't going to be instantaneous
  - Take a look around and estimate developer hours + hardware €/$ bound by GR3
- The strength of GNU Radio is its community
  - Out-of-tree ecosystem: needs proven ways to port C++, Python code to GNU Radio 4
  - Installation base: needs to be packaged, avoid source builds wherever possible
  - Low hurdle to entry: The graphical design frontend is central
- Make new GNU Radio 3 OOTs be "more like GR 4", to make it easier for OOT developers to support both

Getting GNU Radio 4 to be *functionally attractive*

User Experience

- Strength of GNU Radio 4: Speed
- Not that relevant for someone receiving an IoT waveform, if your RPi can already do it with GNU Radio 3

Developer Experience

- Strength of GNU Radio 4: Less antique code base
- Lots of architecturally avoided mistakes
  - Certainly a host of new fun to discover!
- Much slimmer minimum out-of-tree module

GR4 User Experience: Catch up to GNU Radio 3

Block library

- "core functionality" (math ops, converters, filters…) very advanced
- Comms toolbox needs extension
  - Chance to consistently reimplement gr-fec, write working AGCs, use consistent normalization …
- visualization toolkit: GSI has good blockset, but architecturally not necessarily good fit for "home use"

API

Stabilizing

- Easier to write blocks that *use* GNU Radio 4, instead of *extending* it
- Still C++ only for interaction
- Declarative (YAML) way of definining flow graphs
- Need to invest into getting Python bindings

Design Tool

Missing![2]

- Håkon Vågsether will lead development
- Web technology
  - Not just a code port of GRC-Qt to GR4

[2] GSI has a design toolbox for their needs, and you can try it!

Design Tool: Why *on earth* Web Technologies?

- Frequently requested to be able to design GR flowgraphs in the browser
  - We're granting money to someone to finish GR3's GRC-in-the-browser
- > 18 years of experience with GRC show:
  - we're good C++ and Python developers, but
  - we're so-and-so or worse GUI developers, mostly, and
  - writing GUI applications as code is inefficient, and hard to maintain (code tangle), and
  - the "home advantage" of using the same language for DSP and GUI isn't very large
- We'd like to tap not only the *technology*, but the GUI developer *talent* pool
- Needs RPC (probably: REST) endpoints on the GNU Radio runtime process

GR4 Developer Experience

- Great progress on compilation time & memory consumption
- Clear directory structure, responsive team
- Some things look like “template magic", but in fact, fewer pitfalls
- Single-source of thruth: no more separate bindings, yaml description

What is new in GNU Radio 3?

Jeff Long voluntired[3] from his maintainer role:
It's astonishingly hard to achieve Jeff's consistency

Thank you, Jeff!

[3] https://www.gnuradio.org/news/2025-02-20-saying-thank-you-to-jeff/

What's happened in GNU Radio 3 since GRCon24?

Organisationally: Let's share the load!

Subsystem maintainers:

- not every gr-xyz has to be reviewed by the maintainer directly
- well-working model for **gr-uhd**! Thank you, mbrown!
- **GRC** is led by Håkon
- Henning Paul stepped up and coordinates, reviews **gr-iio**

We need more of these! (Talk to me!)

On 3.10: GNU Radio Companion, mostly!

- GRC-Qt works pretty well now!
  - With newer GNU Radio: try `gnuradio-companion --qt`
- GRC workflows

- Abstraction of the "we produce Python XOR C++ XOR ..."
- Driven by heterogeneous platform needs
- Allows to generate non-GNU Radio-code

What's *ongoing* in GNU Radio 3?

CMake & Installer Work

- Paid-for CMake work: modernize CMake, be less "special"
- National Instruments: Make UHD + GNU Radio installers
  - Please be **very** nice to M. Koop, he gets to deal with OS differences *and* CMake

Breaking changes: Needs a minor release ⟶ 3.11

What's to be done *soonish* in GNU Radio 3?

Qt5 → Qt6 migration

- To be reworked atop the installer work
- "Extinction Level Event" avoided (debian trixie still ships Qt5)
- Breaking changes: Possibly in 3.11, if not, we'll need a 3.12 to fulfill our versioning

What's to be worked on in GNU Radio 3 *at some point*?

Bugs, Bugs, Bugs

- Buffer infrastructure: 1 (one) mutex that locks all:
  - Reading buffer positions, writing buffer positions, Reading tags, writing tags, reading message buffers...
  - Not only a performance bottlenecks: can't call Python from buffer context
- Qt threading horribleness
  - Especially (but not excl.) in GRC-generated Python, we call Qt functions in non-GUI threads
    - sporadic data corruption
- Visualization horribleness
  - Waterfall sink with uneven sampling
- AGC horribleness, ...

What's would you *like* to see in GNU Radio 3?

- Superseeding `get_tags_in_range(senseless_copy_into_this, start, end)` by `apply_to_tags_in_range(start, end, function)`
- OOTs: Learn from GR4, prepare people for GR4 migration
  - drop impl/dimpl (`myblock.h`, `myblock_impl.h`, `myblock_impl.cc` ⟶ `myblock.h`, `myblock.cc`) for most use cases
  - make blocks self-describing
    - allow GRC to get block description from block class
    - replace separate pybinding with convenience functions
  - prefer message passing over setters
- Latency-bounded processing

There's no bad questions, just inadequate answers

I can provide the answers, but you'll have to ask!