CyberEther: Heterogeneous GUI

GNU Radio Conference 2022

Presented by Luigi Cruz (@luigifcruz) on September 26th

Summary

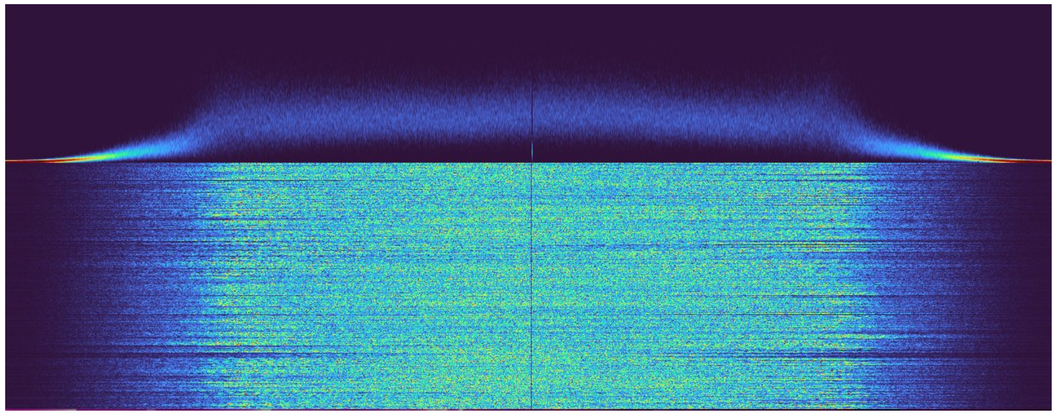
- How CyberEther came to be.
- What CyberEther is at the moment.
- What makes CyberEther different and how it works.
- Future and GNU Radio integration.

Section One: **Early Development**

Early Development

Luigi Cruz
@luigifcruz

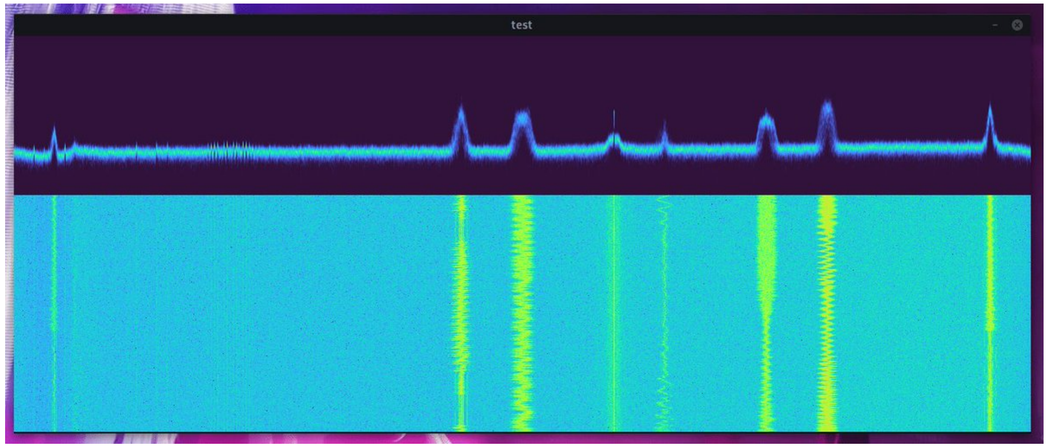
New project! Playing an FM station inside a web-browser with Airspy R2 streaming complex I/Q at 2.5 msps and LiquidDSP demodulating it in real-time. Powered by my libusb translation layer to WebUSB and clever JS threading. No changes are required to the libairspy. #WebAssembly



8:35 PM - Jan 25, 2021 - Twitter Web App

Luigi Cruz
@luigifcruz

This is the spectrum viewer I did for the browser-based SDR console. I'll release an upgraded version that also works on an OS. It'll focus on simplicity, portability, and customizability. Yes, I'm using this as an excuse to learn multiple graphical technologies.

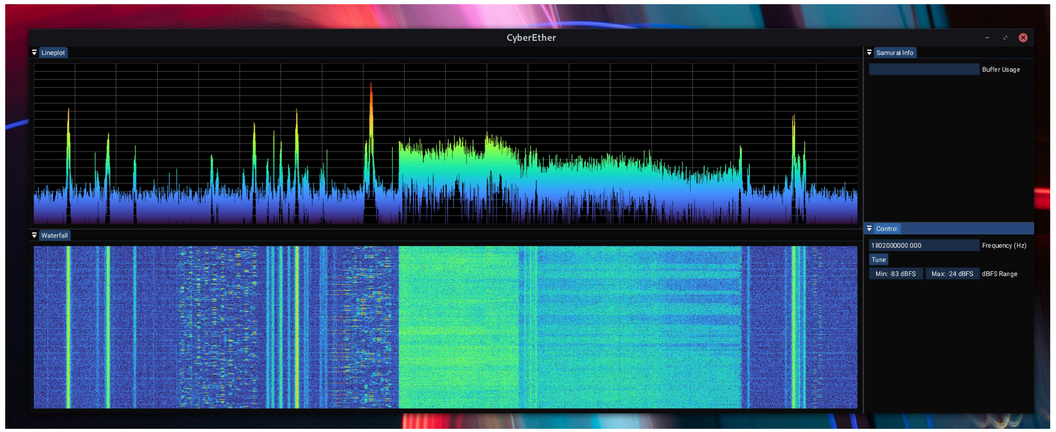


3:32 PM - Mar 15, 2021 - Twitter Web App

Early Development

Luigi Cruz
@luigifcruz

The FFT is now CUDA based.



10:20 AM - Jun 8, 2021 - Twitter Web App

devnulling
@devnulling

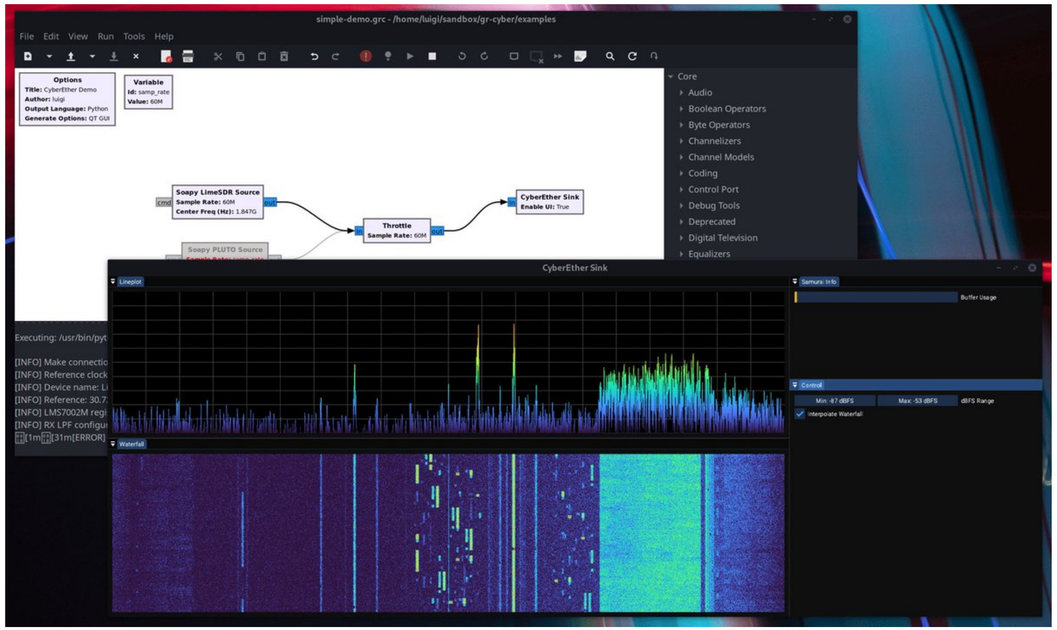
Any change to make this FFT/waterfall into a GR OOT block?
:)

4:13 PM - Jun 8, 2021 - Twitter Web App

Early Development

Luigi Cruz
@luigifcruz

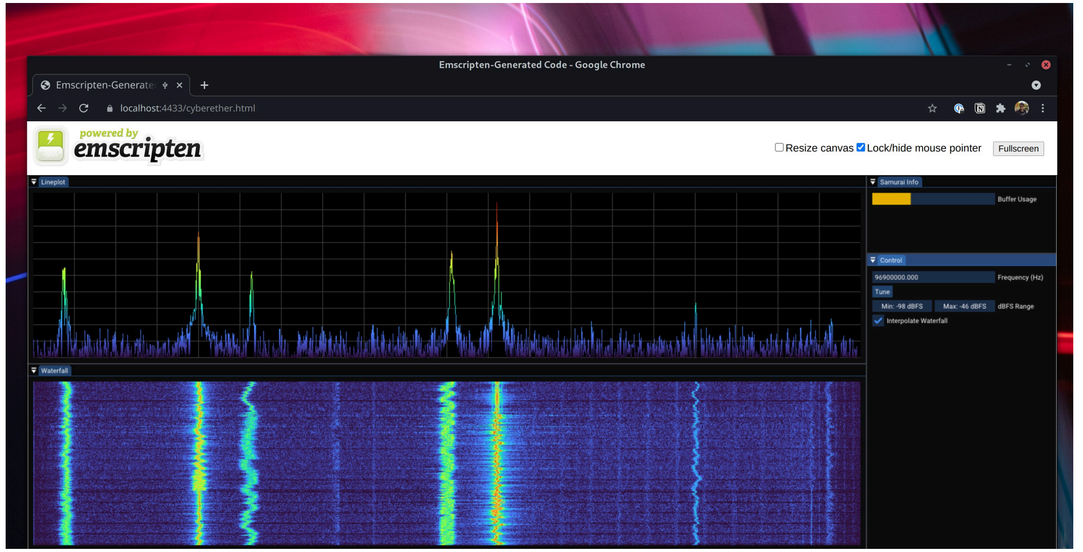
oh hi. cuda spectrogram on gnuradio.



1:47 PM - Jun 17, 2021 - Twitter Web App

Luigi Cruz
@luigifcruz

This is nowhere near real-time yet but it's running inside Chrome, from USB to UI. I think I created a monster.



7:48 PM - Jun 15, 2021 - Twitter Web App

Section Two: CyberEther

CyberEther

Portable and heterogeneously-accelerated GUI for radio signals.

- Built from the ground up to display signals generated by SDRs.
- Currently offers Lineplot, Waterfall, and Spectrogram visualizations.
- Runs as close to the metal as possible using heterogeneous APIs.
- Minimal dependencies with a modular design.
- Low code duplication by abstracting graphical and compute APIs.
- Easy to implement on third-party projects.

CyberEther

Lineplot

- A simple and lightweight way to visualize frequency domain signals.
- Low memory requirements. No data retention.

CyberEther

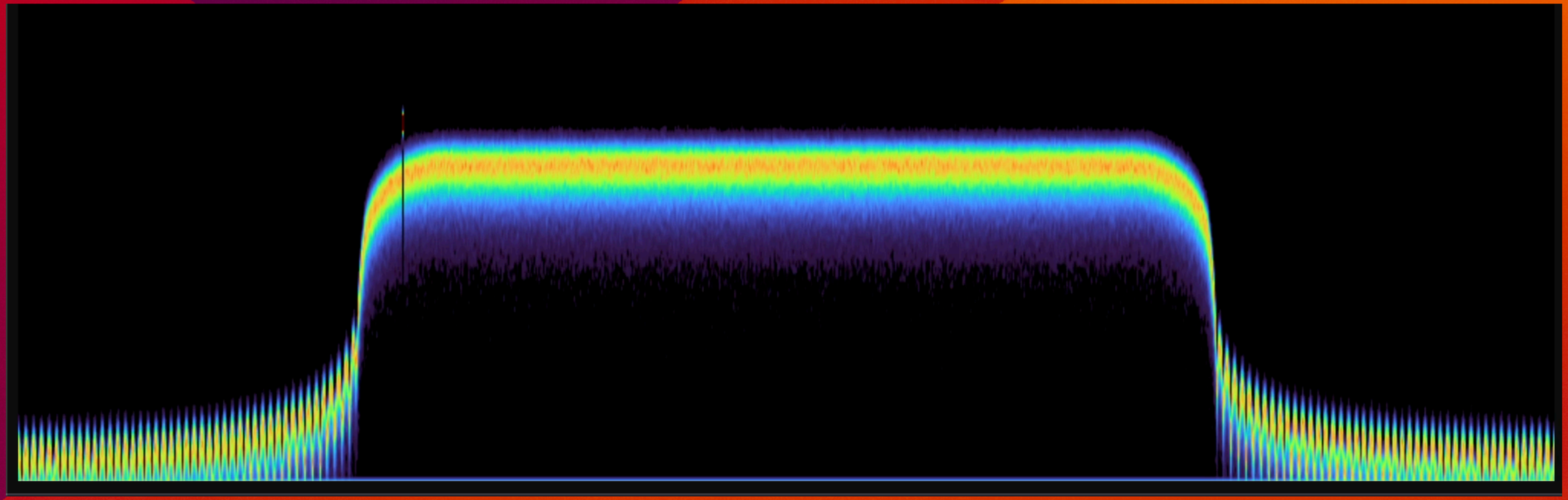
Waterfall

- Standard way to visualize how frequency domain signals change over time.
- Larger memory requirements than Lineplot. Data is retained.
- Output can be fed to Neural Network for inference.

CyberEther

Spectrogram

- A **wholesome** way to visualize how frequency domain signals change over time.
- Lower memory requirements than Waterfall but much more compute-intensive.



DEMO

It hopefully works!

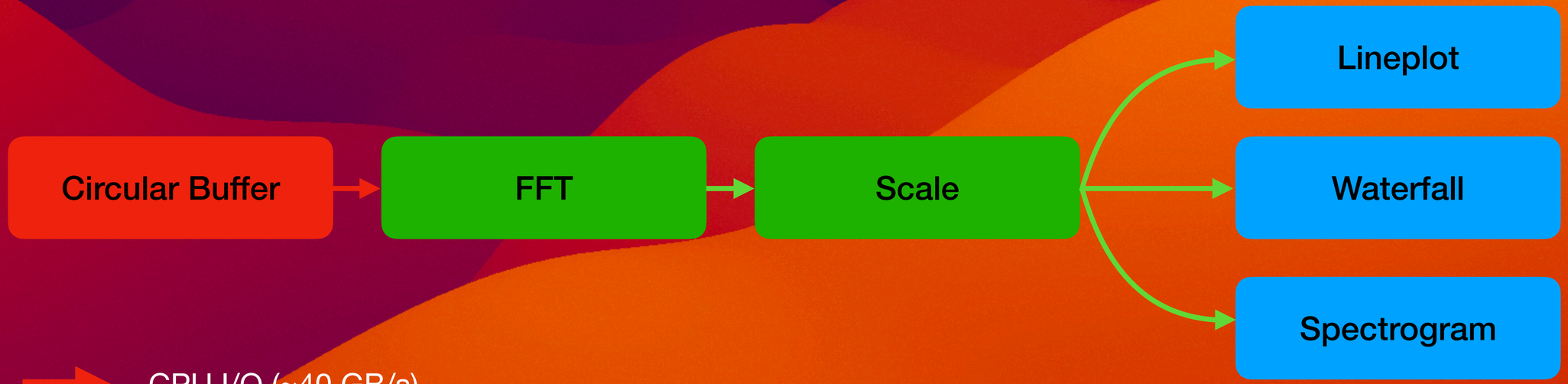
CyberEther

Specifications for NERDS

- Written in modern **C++20**.
- Compiled with **Meson** build system.
- Utilizes **FFTW3** for the CPU-based FFT.
- The standard way to contribute is a GitHub **pull-request**.

CyberEther

Example Flowchart



Section Three: **Why & How?**

CyberEther

Built from the ground-up to display signals generated by SDRs.

- Less code to optimize.
- Built to work in real-time.
- No bloat trying to support everything.
- More friendly ways to integrate with an existing project.
- User-friendly visualizations targeted at signal products.

CyberEther

Runs as close to the metal as possible using heterogeneous APIs.

- Accelerated **graphics** with low-level frameworks (Metal, Vulkan, WebGPU).
- Accelerated **compute** with heterogeneous APIs (CUDA, Metal, Vulkan).
- Above covers operations on Apple, NVIDIA, AMD, Raspberry Pi, and more!
- Fallback to CPU-based compute when no hardware acceleration is available.
- Dependencies aren’t mandatory, it will compile with the available ones.

CyberEther

Low code duplication by abstracting graphical and compute APIs.

- Graphical modules are written on top of a abstraction layer.
- This abstraction layer for graphics is called **`Jetstream::Render`**.
- The graphical backend can be selected during runtime.
- Compute modules are also abstracted using **`Jetstream::Module`**.
- Module backend can be selected in runtime.
- Mix-match of backends are not ideal but supported.

CyberEther

Easy to implement on third-party projects.

- Front-end agnostic.
- All visualizations are rendered on a headless frame buffer.
- Rendered frame can be attached to any window (Qt, Cocoa, MTKView, etc).
- CyberEther uses ImGui only for windowing, no actual render is handled by it.
- Rendered frame can even be shared with another process using DMA-BUF.
- TL;DR: It’s easy to use on iOS, Android, macOS, or even a browser.

Section Four: **Future & GNU Radio**

CyberEther

Current status of **graphical** modules.

Module	Metal	CUDA	CPU+Render
Lineplot	✅ (Full)	❌ (Porting)	✅ (Slow but full)
Waterfall	✅ (Full)	❌ (Porting)	✅ (Slow but full)
Spectogram	❌ (Next)	❌ (Porting)	✅ (Slow but full)

CyberEther

Current status of **compute** modules.

Module	CPU	CUDA	Metal	Vulkan	Description
Amplitude	✅	❌ (Porting)	❌ (Next)	❌ (Future)	Complex data to power.
FFT	✅	❌ (Porting)	❌ (Next)	❌ (Future)	Channelization.
Multiply	✅	❌ (Porting)	❌ (Next)	❌ (Future)	Vector multiplication.
Scale	✅	❌ (Porting)	❌ (Next)	❌ (Future)	Scaling vector by factor.
Windowing	✅	❌ (Porting)	❌ (Next)	❌ (Future)	Apply window to vector.

CyberEther

Future development

- Implement CyberEther inside QtWindow.
- Add more compute (Metal) and graphical (Vulkan) backends.
- Add font support.
- Improve handling of multiple compute backends at the same time.

DEMO

It hopefully works (twice)!

Thanks for listening!

https://github.com/luigifcruz/CyberEther

Questions?

Contact me!

https://luigi.ltd/contact/