# GNU Radio Studio: Use Cases and Future Plans

Designing, running and controlling GNU Radio 4 flowgraphs

Håkon Vågsether and Josh Morman

# gr-studio

Browser and desktop frontend for GNU Radio 4 (React, TypeScript, Electron)

### 1. Design

Pick blocks from the catalog, connect ports, and set parameters and variables.

### 2. Arrange

Arrange plots, controls and audio output into an application view.

### 3. Run

Start a session and change parameters and variables while it runs.

Graphs are saved as .gr4s files. Application views can open in the app, in a new tab or in a separate window.

Interacts with ***gr4-control-plane***

*https://github.com/gnuradio/gnuradio4-studio*
*https://github.com/gnuradio/gnuradio4-control-plane*

# gr4-control-plane

A small C++23 service that manages GNU Radio 4 sessions over HTTP

- Create, start, stop, restart and delete sessions
- Read and change block settings on a running graph
- Block catalog built from the GR4 plugins that are loaded
- Session state kept in memory
- REST API, plus *gr4cp-cli* as a thin client

# gr4-control-plane - HTTP API

```
POST    /sessions
GET     /sessions
GET     /sessions/{id}
DELETE  /sessions/{id}
POST    /sessions/{id}/start
POST    /sessions/{id}/stop
POST    /sessions/{id}/restart
GET     /sessions/{id}/blocks/{name}/settings
POST    /sessions/{id}/blocks/{name}/settings
GET     /blocks
GET     /blocks/{id}
GET     /schedulers
```

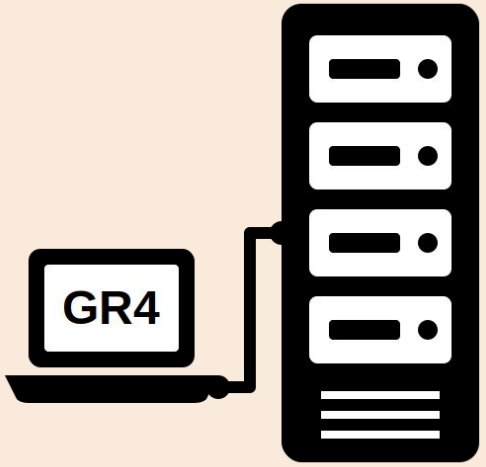
# gr4-studio Use Cases

## 1. Fully local

Control plane and studio running locally

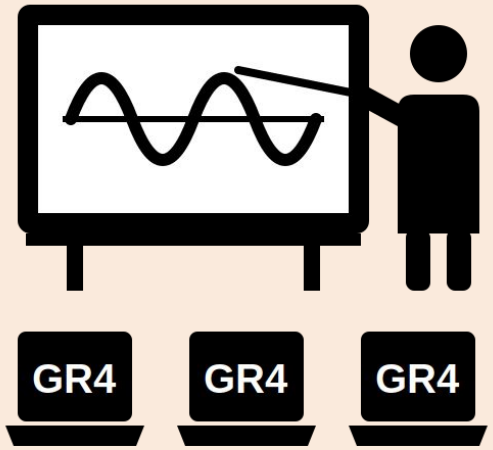

## 2. Remote control

Control plane running remotely, studio running locally or remotely




## 3. Fully in-browser

No installation needed



# Running Studio

## Browser

```
docker run --rm -p 8080:8080 \
  ghcr.io/gnuradio/gnuradio4-studio:latest
```

Then open *http://localhost:8080*

## Desktop

```
gr4-studio
```

Starts a local *gr4cp_server* on a free port, in a bundled Electron runtime

## Remote

```
gr4-studio --remote \
  http://host:8080
```

Uses a control plane running on another machine

### Build everything from source and run

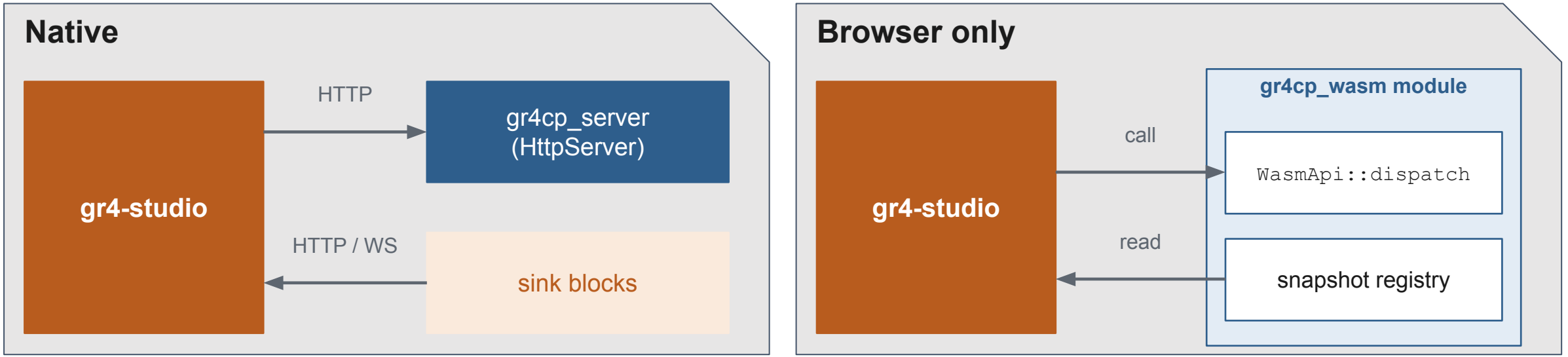
```
git clone https://github.com/gnuradio/gnuradio4.git && cd gnuradio4
cmake --preset full && cmake --build --preset full
source build/full/activate.sh && gr4-studio
```

# Studio Sink Blocks

- Ordinary GR4 blocks, built and loaded as plugins
- Note: Studio blocks must be built and installed where GR4 control plane lives

| | | |
|---|---|---|
| Series | 2D series | Dataset |
| Power spectrum | Waterfall | Scalar |
| Status | Audio | Image |

# WebAssembly




- The control plane builds with Emscripten; *WasmApi::dispatch* replaces the HTTP server and keeps the same request and response format
- Sinks publish into an in-process registry instead of opening sockets
- Studio uses it when built with *VITE_CONTROL_PLANE_TRANSPORT=wasm*
- **Try it yourself at *https://gr4wasmdemo.vercel.app/***

# Demo

# Current Status and Future Plans

- Studio is under active development
- The control-plane API is kept small and is still changing
- GNU Radio 4 itself is in RC3, working towards its first stable release
- GR4 Studio development continuing under ARDC grant
- Future
  - Add OOT modules easily
  - Sharing flowgraphs with URLs?
  - Cleaner UI
  - General UX improvements

# Links

| | |
|---|---|
| `github.com/gnuradio/gnuradio4` | Workspace that builds everything |
| `github.com/gnuradio/gnuradio4-studio` | Studio and Studio blocks |
| `github.com/gnuradio/gnuradio4-control-plane` | Control plane |

Discussion happens in the GR4 Technical Users room on Matrix, and there are open GR4 developer calls twice a month.

# Questions?