
Elk
Audio OS

# Running DSP algorithms on hardware using Faust and Elk Audio OS


Functional Audio Stream

## CCRMA Workshop, Feb. 8th 2020


Stefano Zambon stefano@elk.audio

# Useful Links

| | |
|---|---|
| Faust Editor | https://faust.grame.fr/tools/editor/ |
| Faust Manual | https://faust.grame.fr/doc/manual/ |
| Elk Docs | https://elk-audio.github.io/elk-docs |
| Elk Github | https://github.com/elk-audio |

# Disclaimer

I'm a Faust noob!

The workshop is about how to deploy your Faust code on Elk Audio OS

# Outline

1. Board connection and sound check
2. Elk Overview
3. Simple Faust examples & Elk configuration
4. Python Glue application
5. Online compiler
6. Experiments and Q&A

# Requirements

1. Terminal with SSH
2. Headphones with 3.5mm plug

# Connect to your board over WiFi

Connect your PC

WiFi: Elk_Workshop_2.4GHz

PW: elk_at_ccrma

On your board, there is a sticker with a name elkpi-X.

```
$ ssh mind@elkpi-X.local
```

Pwd: elk

# If hostname doesn't work for you

## Use IP address:

| | |
|---|---|
| elkpi-1 | 192.168.0.129 |
| elkpi-2 | 192.168.0.153 |
| elkpi-3 | 192.168.0.179 |
| elkpi-4 | 192.168.0.141 |
| elkpi-5 | 192.168.0.122 |
| elkpi-6 | 192.168.0.120 |
| elkpi-7 | 192.168.0.132 |
| elkpi-8 | 192.168.0.142 |

| | |
|---|---|
| elkpi-9 | 192.168.0.146 |
| elkpi-10 | 192.168.0.143 |
| elkpi-11 | 192.168.0.175 |
| elkpi-12 | 192.168.0.169 |
| elkpi-13 | 192.168.0.133 |
| elkpi-14 | 192.168.0.103 |
| elkpi-15 | 192.168.0.126 |
| | |

# Hostname identification error

If SSH fails with this error:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Type on your local shell (macOS / Linux):

```
$ ssh-keygen -R elkpi-X.local
```

# Sharing a Board (Linux / OSX)

1st person types:

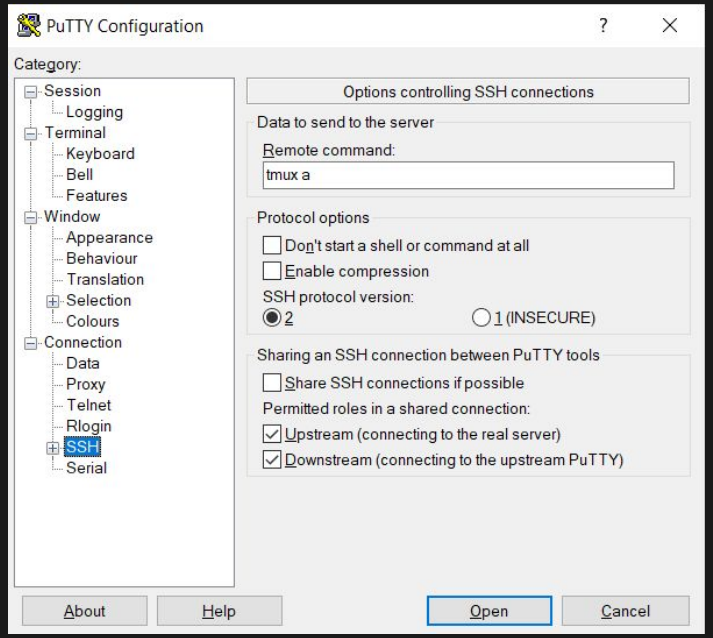
```
$ cd /udata/ccrma
$ tmux
```

2nd person runs SSH with:

```
$ ssh mind@elkpi-X.local -t tmux a
```

# Sharing a Board (Windows + PuTTY)

2nd person has to change the custom command to “`tmux a`” in the SSH tab before opening [her|his] SSH connection:



# Tmux crash course

| | | |
|---|---|---|
| Ctrl+B | (release) c | create a new terminal window |
| Ctrl+B | (release) num | switch active window |

If you don’t want to use tmux, you’ll have to launch blocking processes in background with &

# File transfer / remote editing

OSX: connect to samba folder with shortcut cmd-K from Finder.
Ubuntu: “Files” window, “Other Locations”, and in “Connect to Server” field, enter: `smb://elkpi-X.local/`
Win 10: right-click this PC. “Add a network location”. … in field “//elkpi-X.local”.

User: mind Pwd: elk

If Samba doesn’t work:

```
$ scp yourfile mind@elkpi-X.local:/udata/
```

# Sound Check!

```
$ sushi -r -c configs/sushi/sound_check.json
```

You should hear a synth playing a pattern.

Ctrl+C to stop.


Audio OS

# Technology

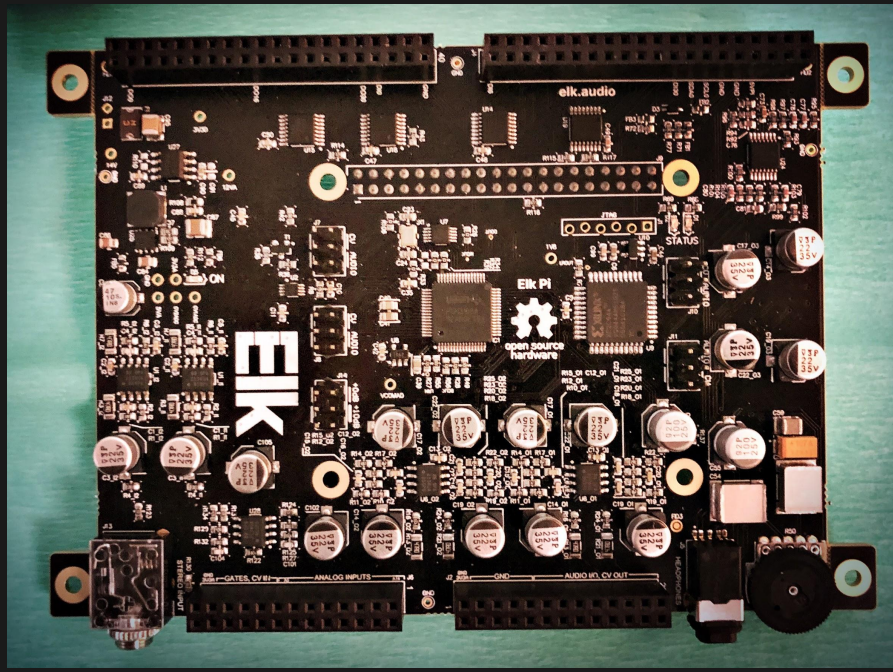
Custom Linux distribution
Xenomai real-time Kernel
ARM & x86

Connectivity: WiFi, BLE
Plugin support: VST2, VST3, LV2
CV & Gate I/O
Ableton Link

# Hardware

# Elk Development Kit

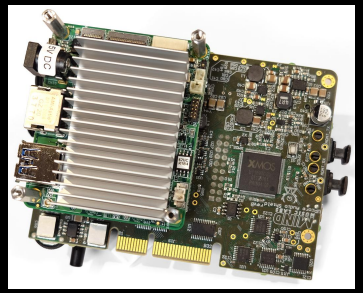
Elk Pi Hat
Open-Source Hardware
Optional control-board
SDK
6 analog In / 8 analog out
CV in/out & gate/triggers
16 sensor analog ins
32 in/out GPIOs



# Supported CPUs

See elk.audio for complete specs

## Intel Atom



Intel Atom X5-Z8350 @ 1.92 GHz
Quad Core

Intel Atom E39XX

(any Intel SOC trivial to support)

## Raspberry Pi 3


Elk Pi Hat for Raspberry Pi 3
Broadcom BCM2837
4 x ARM Cortex A53 @ 1.2 GHz
Open Source Dev Kit
6 Audio I/O

RPi 4 support 2020
4 x ARM Cortex A72 @ 1.5 GHz

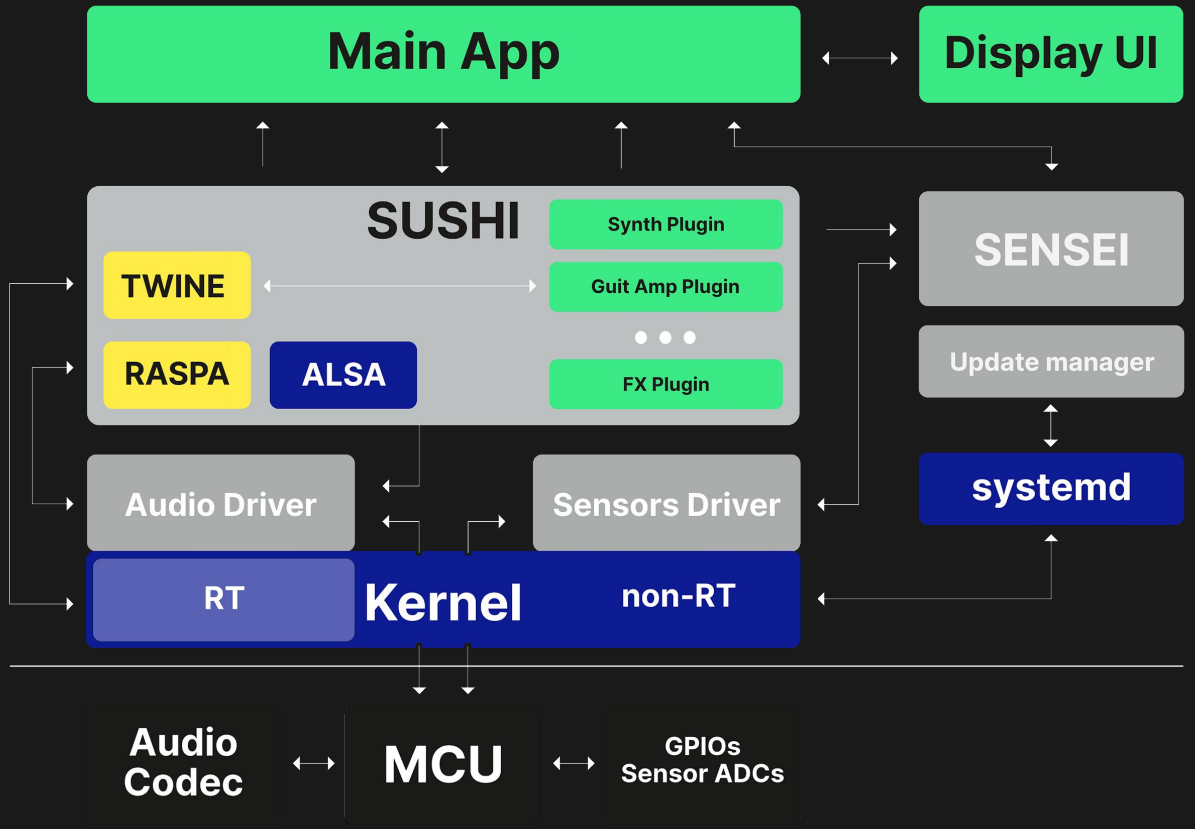
## iMX 7/8


NXP i.MX8M Mini SoC
4 x ARM Cortex A53 @ 1.8-2 GHz
1 x ARM M4

STM32MP1 in alpha
i.MX8M Nano support 2020

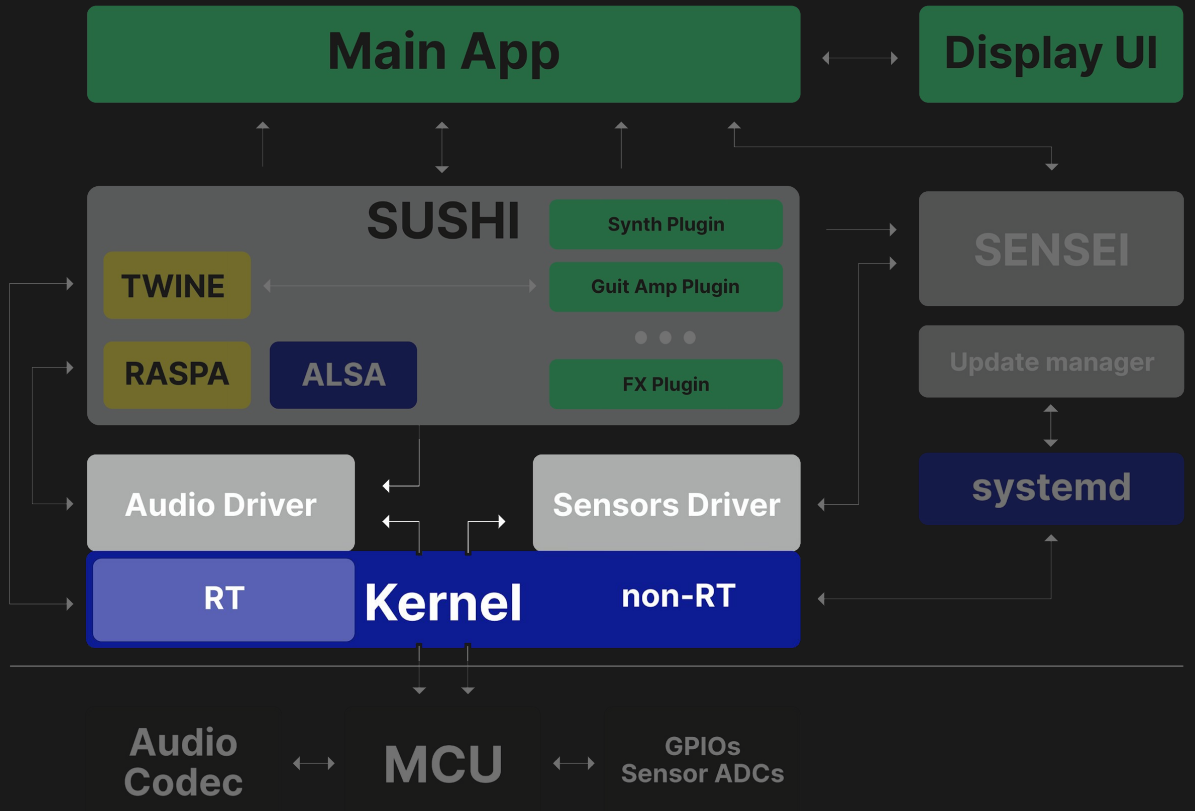
# Architecture

# Architecture



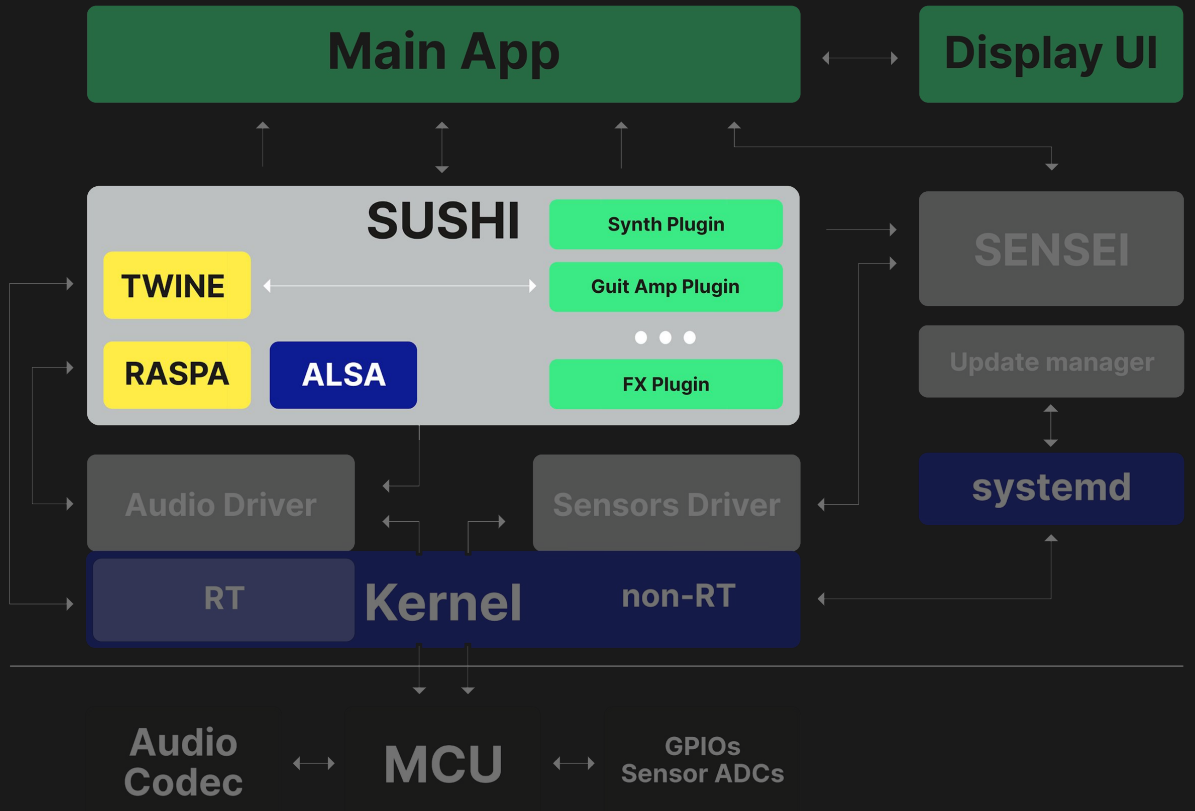
# Architecture

## Dual Kernel



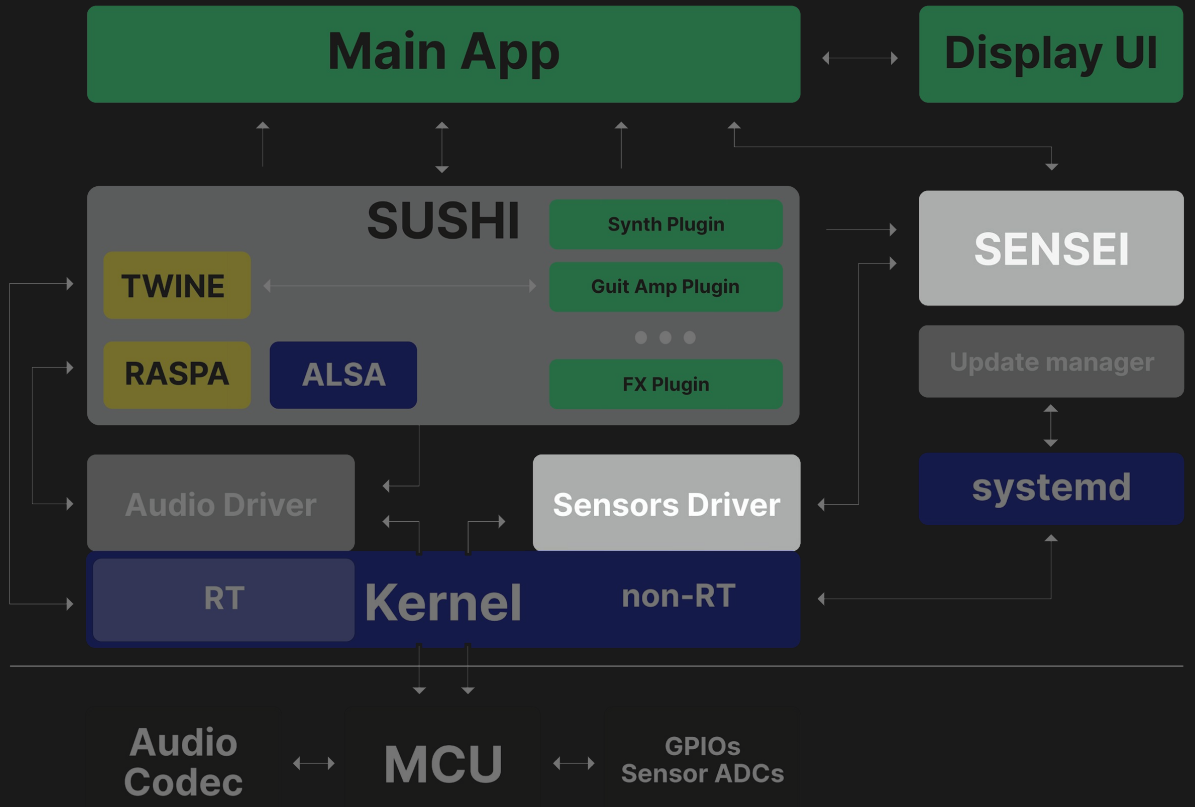
# Architecture

Dual Kernel
Plugin Host



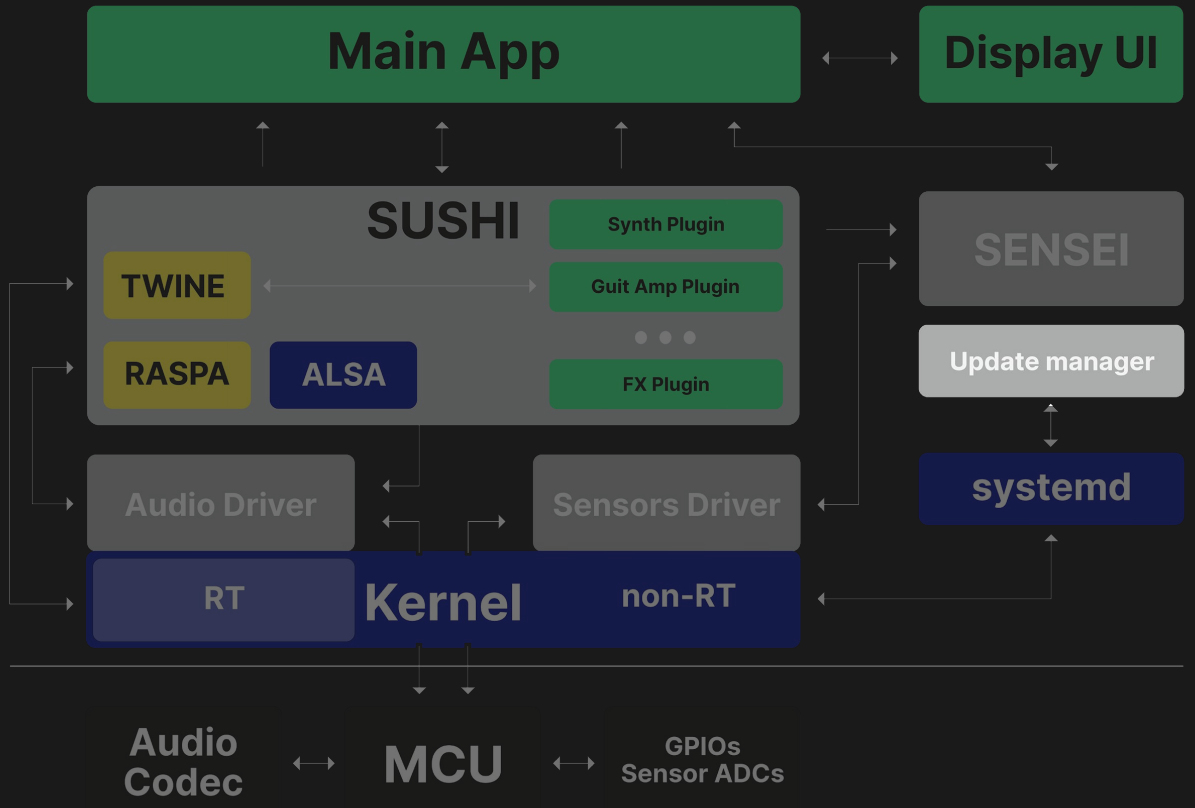
# Architecture

Dual Kernel
Plugin Host
Sensor Daemon



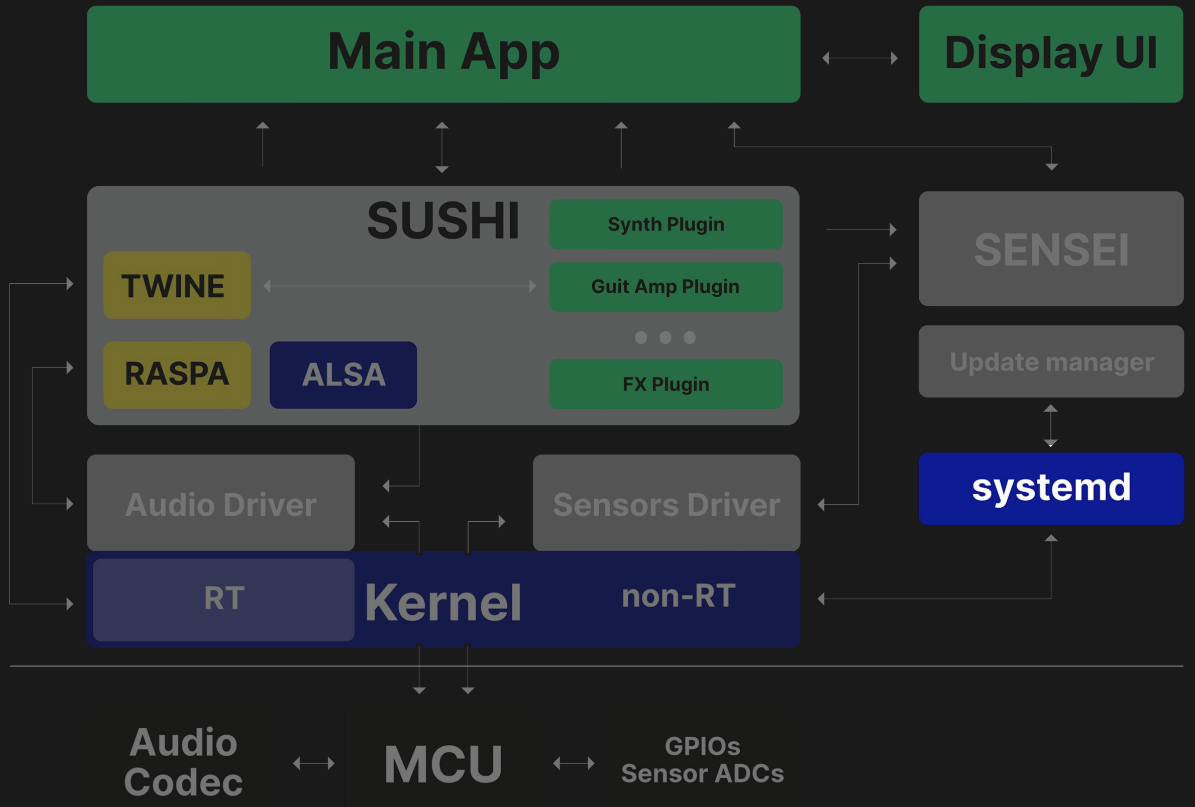
# Architecture

Dual Kernel
Plugin Host
Sensor Daemon
Software Update



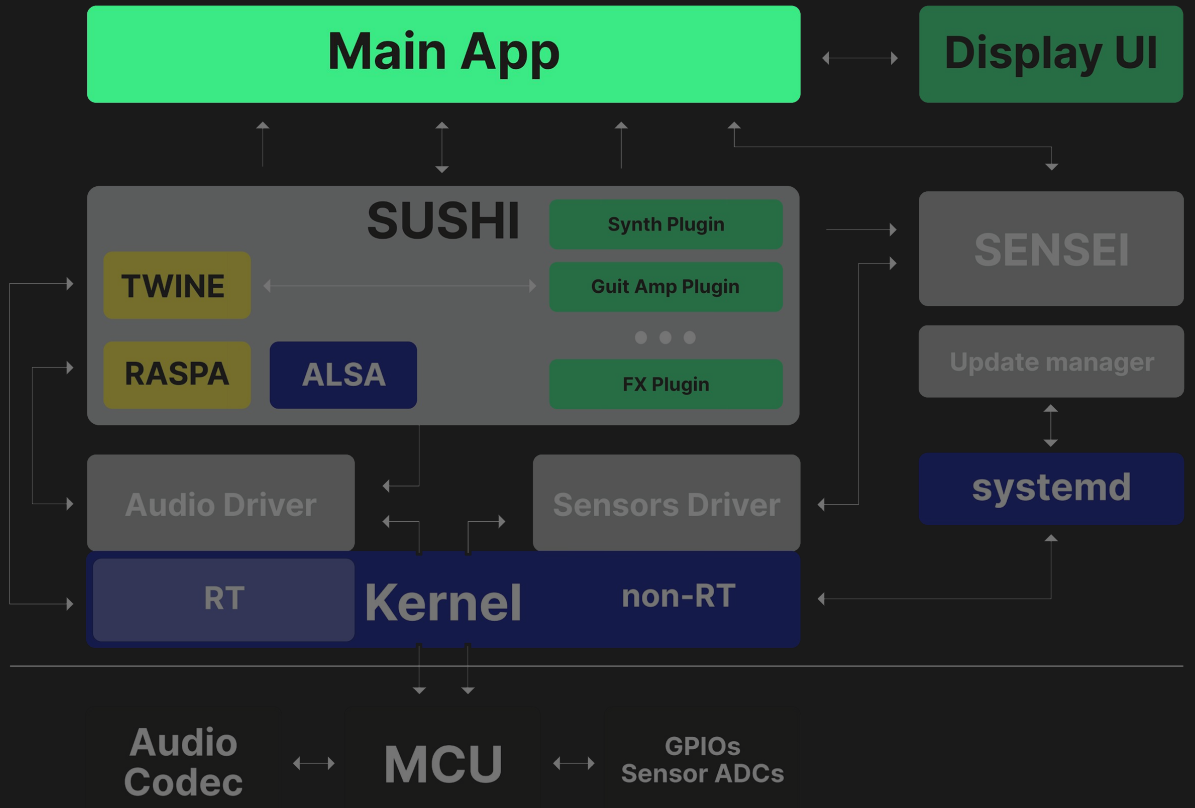
# Architecture

Dual Kernel
Plugin Host
Sensor Daemon
Software Update
Systemd



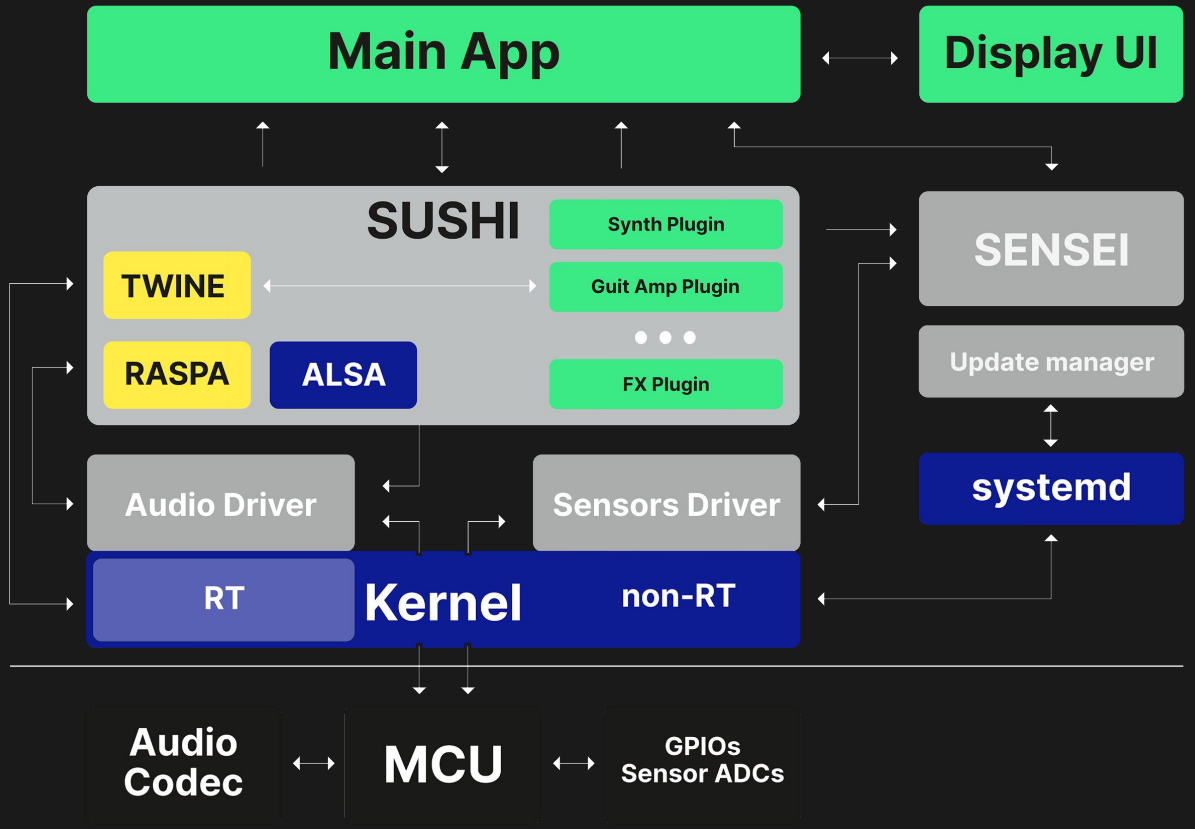
# Architecture

- Dual Kernel
- Plugin Host
- Sensor Daemon
- Software Update
- Systemd
- Main App



# Architecture

- Dual Kernel
- Plugin Host
- Sensor Daemon
- Software Update
- Systemd
- Main App



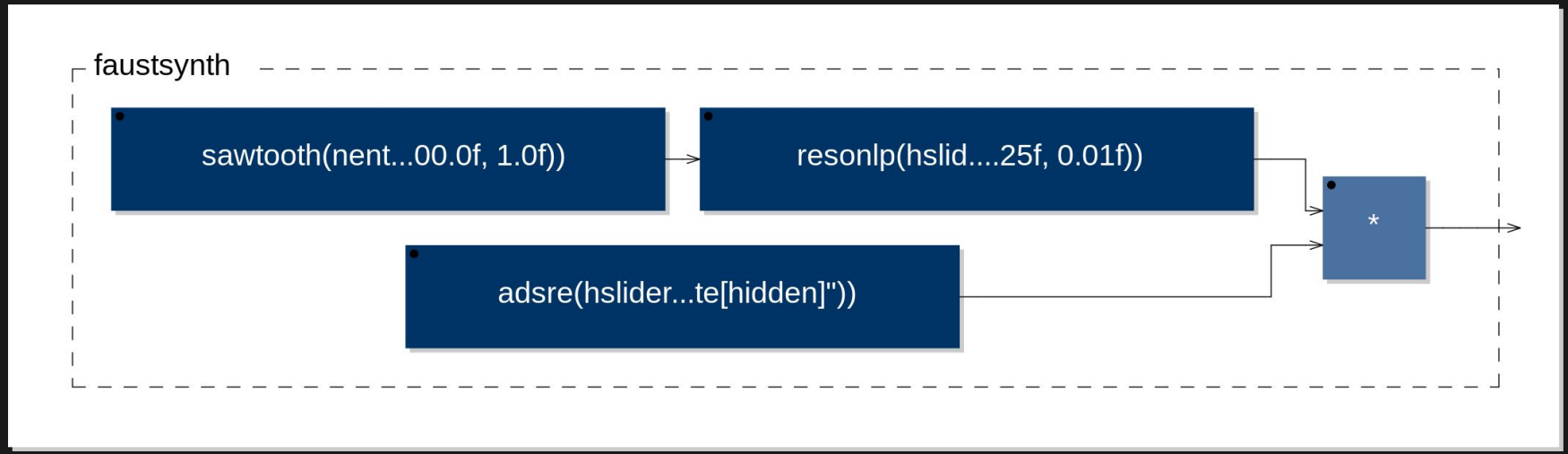
# Running Faust plugins

# Directory layout

```
.
├── apps                    # Python glue apps
├── configs                 # Elk config files
│   ├── sensei
│   └── sushi
├── faust-code              # Faust example sources
└── plugins                 # Pre-built binaries for Elk Pi
```

# Simple Faust synthesizer

`faust-code/faustsynth.dsp`



# SUSHI configuration

configs/sushi/faustsynth.json:

```
# ...
"plugins" : [
    {
        "path" : "/udata/ccrma/plugins/faustsynth.so",
        "name" : "synth",
        "type"   : "vst2x"
    }
]
# ...
```

# Start SUSHI

```
$ sushi -r -c configs/sushi/faustsynth.json
```

If you have a USB MIDI controller:

(open a new tmux tab with Ctrl+B c)

```
$ aconnect -l
$ aconnect source_num dest_num
```

e.g. `$ aconnect 16 128`

# Control parameters over OSC

Get info on the plugins & parameters in a config:

```
$ sushi --dump-plugins -c configs/sushi/faustsynth.json
```

```
$ oscsend localhost /parameter/synth/[name] f [value (0..1)]
```

example:

```
$ oscsend localhost /parameter/synth/Cutoff f 0.1
```

or you can use any OSC app on your laptop / mobile .

# SENSEI configuration

configs/sensei/uihat.json

It’s already configured for the top UI shield, so:

**Don’t worry about it!**

(unless you came with breadboard & components...)

# Start SENSEI

(open a new tmux tab with Ctrl+B c)

```
$ sensei -f configs/sensei/uihat.json
```

You can leave it running for the rest of the workshop.

# Python glue app

Helper module for UI hat: apps/elk_ui.py

Minimal Glue app: apps/main_app_minimal

`ElkUIController` : access controls via callbacks

set_led(...), set_display_lines(...)

You can use elkpy to control SUSHI

# Run the glue app

(open a new tmux tab with Ctrl+B c)

```
$ cd apps
$ ./main_app_minimal
```

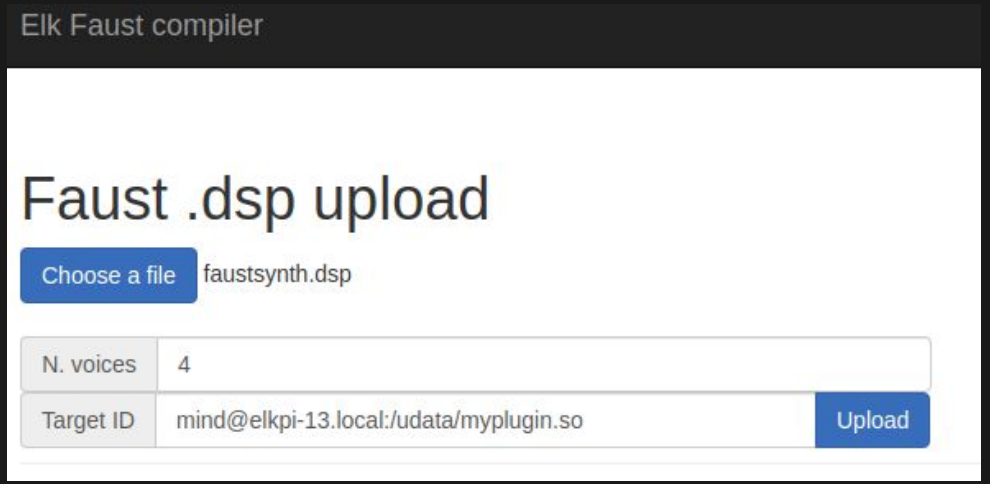
Buttons: play notes
Knobs: change parameters

Ctrl+C to stop

# Online Faust compiler
# (don’t try it right now!)

Open in a web browser: http://elkbuild.local:5000/

or: http://192.168.0.128:5000/



# Analyzing plugin RT performance

```
$ sushi --timing-statistics -r -c path/to/config.json
```

Rough analysis with Xenomai scheduler statistics:

```
$ watch cat /proc/xenomai/sched/stat
```

Detailed analysis using SUSHI's timings queried over gRPC:

```
$ ./apps/benchmark-synth -p [sushi_plugin_name]
```

# More complex example
# (be careful when using headphones!)

Two tracks: guitar PM synth + FX chain

Connect audio out 3-4 to audio in 3-4 on the shield

```
$ sushi -m2 -r -c configs/sushi/guitar_fx.json
```

(-m2 is for multicore processing)

```
$ cd apps
$ ./main_app_full
```

# Experiments Ideas

- Try your own Faust code  (test it with the online editor first)
- Add more Faust plugins in one or more tracks
- Modify the Python app to make your own instrument


Thank you!