
Elk
Audio OS

Elk Workshop NAMM 2020

Stefano Zambon stefano@elk.audio

Docs	https://elk-audio.github.io/elk-docs
Code	https://github.com/elk-audio
Community Plugins	https://elk-audio.github.io/elk-community
Website	https://elk.audio
Forum	https://forum.elk.audio

Outline

1. Connection and board test
2. Elk Overview
3. Simple examples
4. Analyzing plugin issues
5. 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

PW: elk_at_adc

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

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

Pwd: elk

Running and controlling plugin: file transfer

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”.

For all: User: mind PW: elk

If Samba doesn’t work:

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

Make some noise with SUSHI

```
$ cd /udata/elk_namm2020_workshop
$ sushi -r -c configs/sushi_config_arp.json
```

You should hear a Cello-like synth in a repeating pattern

CTRL+C to stop


Audio OS

Technology

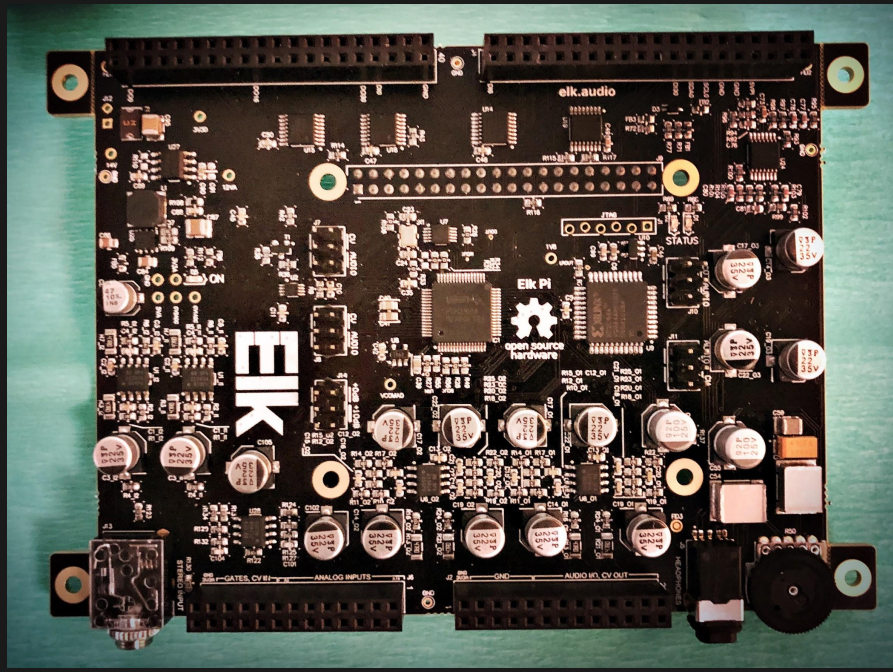
Custom Linux Distribution
Xenomai realtime Kernel
ARM & x86

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

Hardware

Elk Development Kit

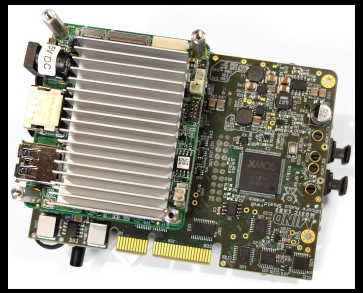
Elk Pi Hat
Open-Source Hardware
Optional control-board
SDK
6 analog In / 8 analog out
CV in/out & gate/triggers
16 sensor analog ins
32 in/out GPIOs



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

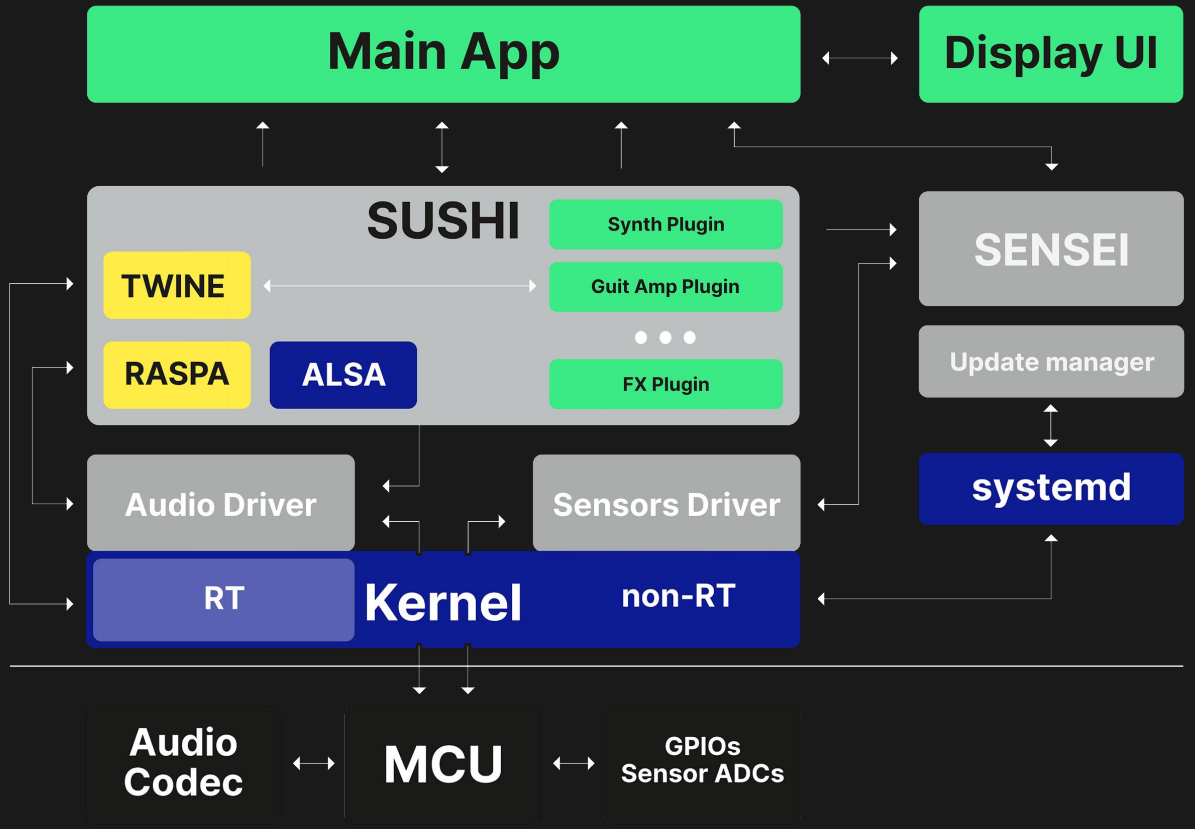
iMX 7/8

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

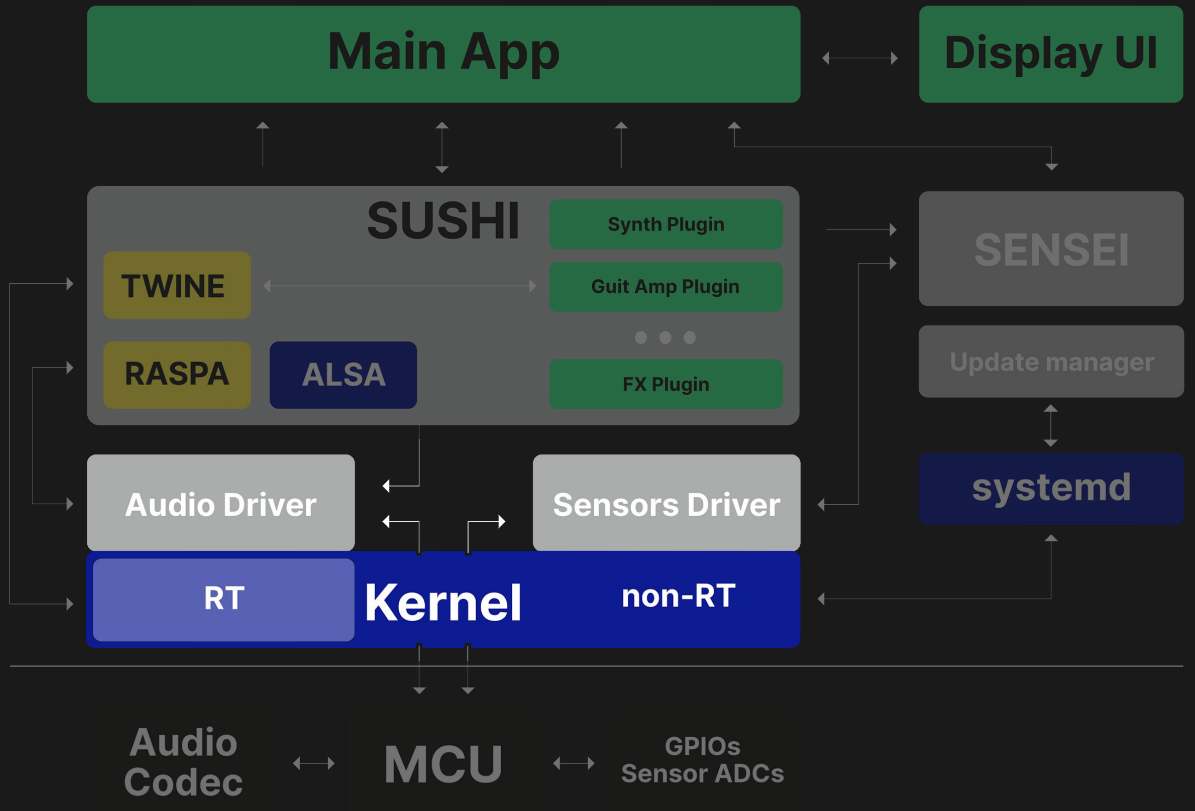
Architecture

Architecture



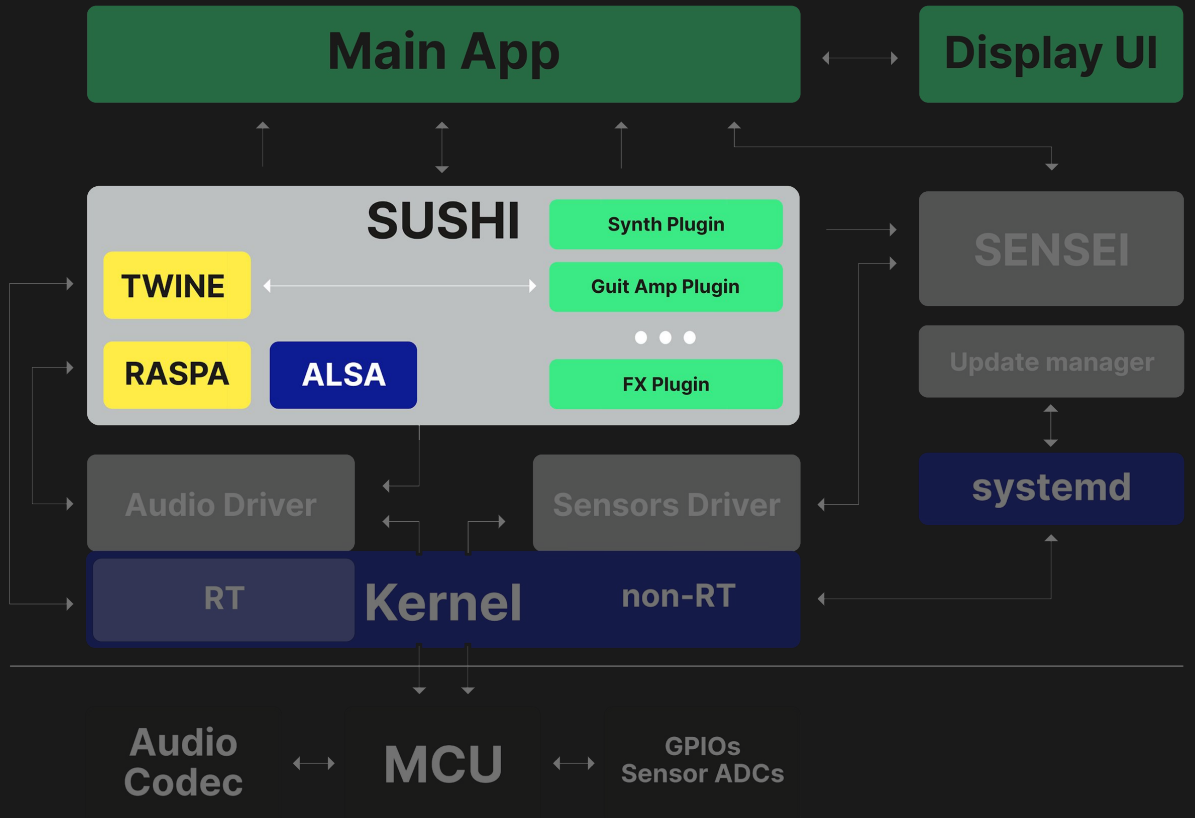
Architecture

Dual Kernel



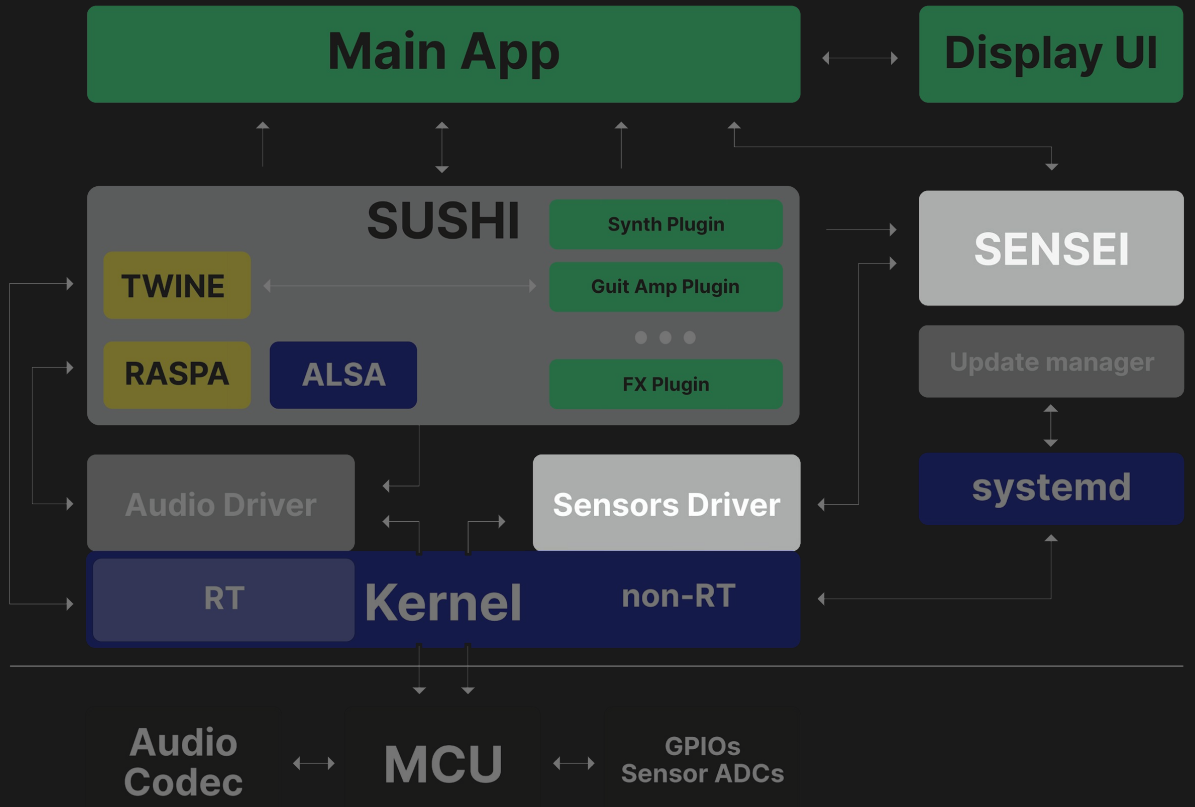
Architecture

Dual Kernel
Plugin Host



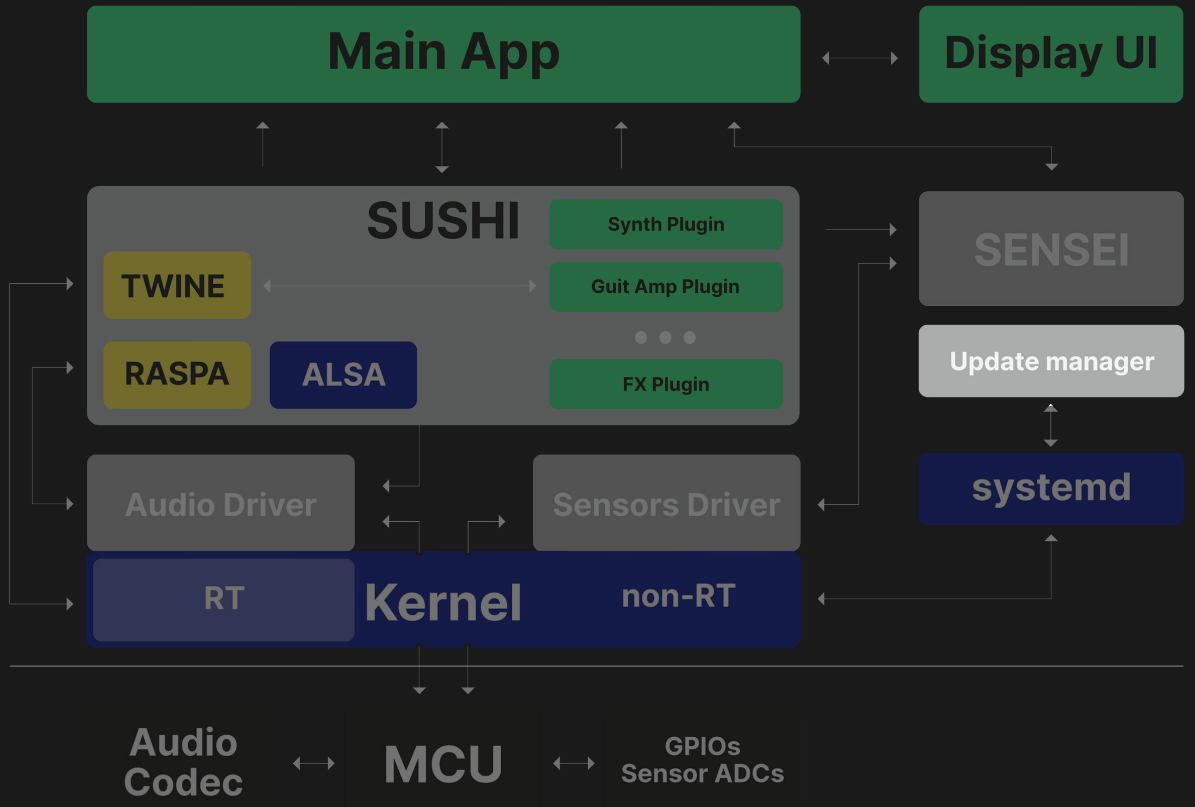
Architecture

Dual Kernel
Plugin Host
Sensor Daemon



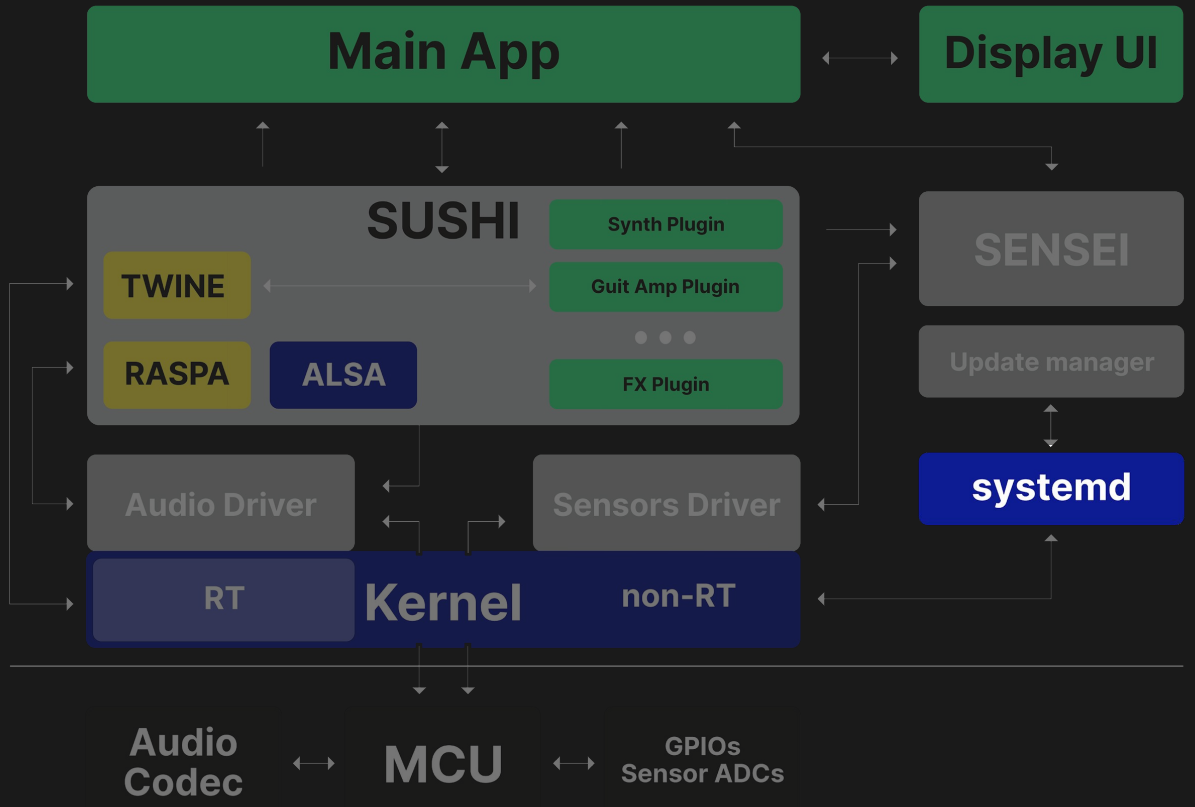
Architecture

Dual Kernel
Plugin Host
Sensor Daemon
Software Update



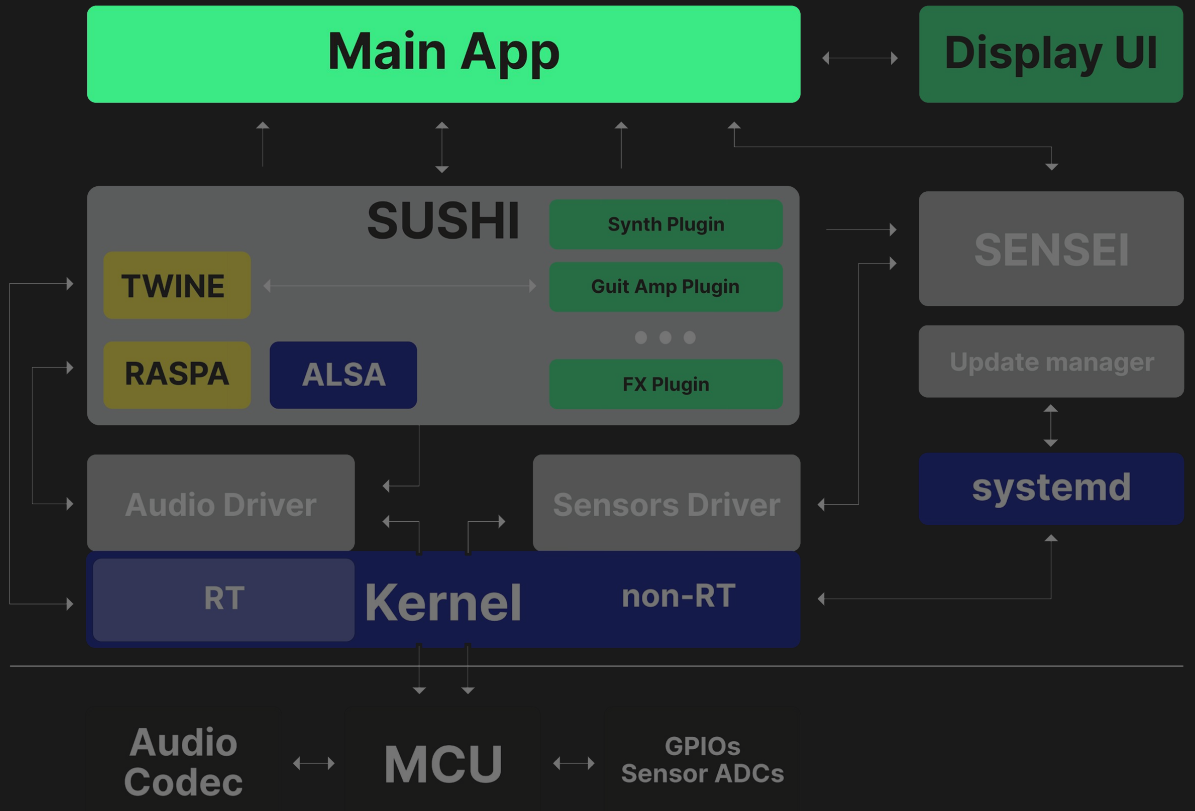
Architecture

Dual Kernel
Plugin Host
Sensor Daemon
Software Update
Systemd



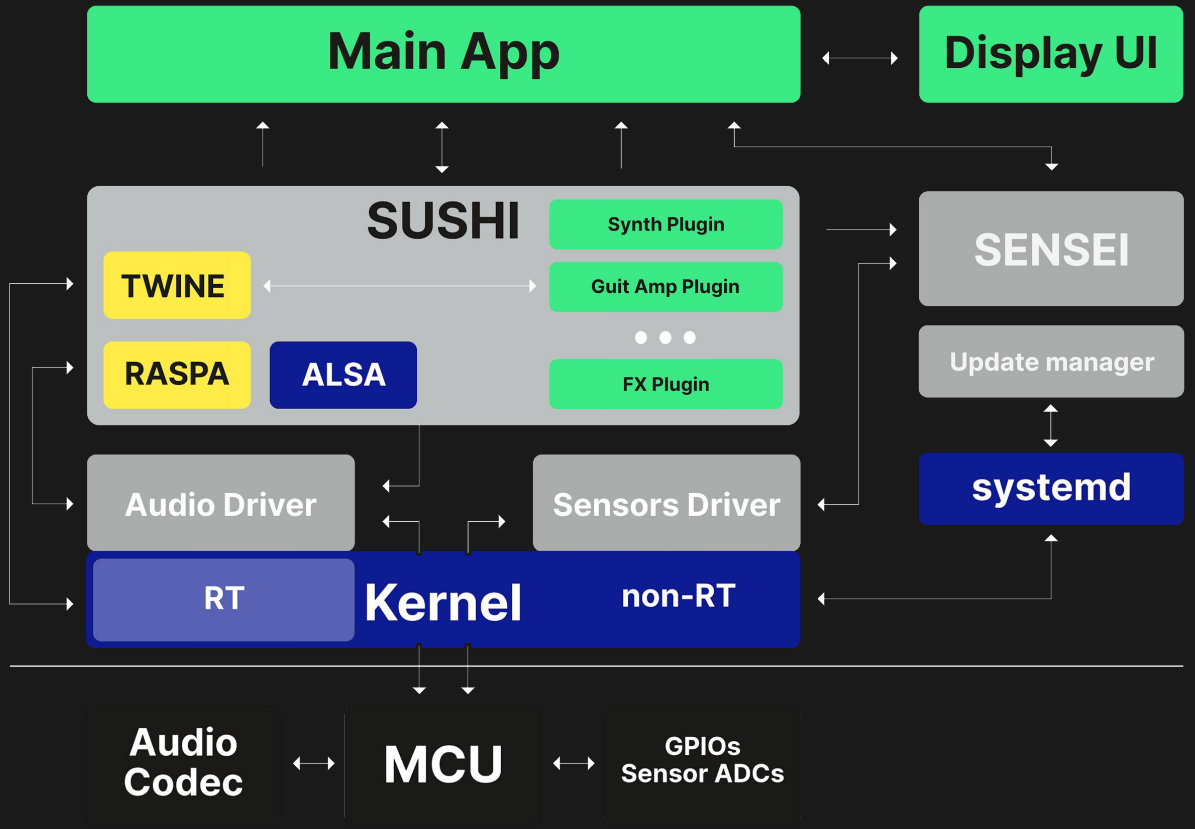
Architecture

Dual Kernel
Plugin Host
Sensor Daemon
Software Update
Systemd
Main App



Architecture

Dual Kernel
Plugin Host
Sensor Daemon
Software Update
Systemd
Main App



Working with Elk

Plugin Example

VST 2.4 plugin built with JUCE using example classes

Chain: Sampler Synth -> VCF -> Reverb

Relevant file: Source/SynthProcessor.h

Running and controlling plugin: Sensei & Python

```
$ sushi -r -c configs/sushi_config.json &
$ sensei -f configs/sensei_config.json &
```

Start python glue program

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

Ctrl-C to kill main app when done playing.

```
$ killall -2 sushi
$ killall -2 sensei
```

(otherwise, use multiple SSH terminals or tmux)

Sensei and creating a simple App

Configuration files:

SUSHI, SENSEI, Python application

More complex application

```
$ sushi -r -c configs/sushi_config.json &
$ sensei -f configs/sensei_config.json &
$ ./apps/main_app_full
```

Analyzing plugin RT performance

```
$ sushi --timing-statistics -r -c configs/sushi_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 elk_juce_example
```

Analyzing Xenomai mode-switches

Let's do something bad in the RT processing callback:

```
// ...
if (! midiMessages.isEmpty() )
{
    dontAllocateMeinRt = new float[4096];
    // leaks... but we don't care for this example
}
synth.renderNextBlock (buffer, midiMessages, 0, buffer.getNumSamples());
// ...
```

Analyzing Xenomai mode-switches (2)

```
$ sushi -r -c configs/sushi_config_msw.json &
$ watch cat /proc/xenomai/sched/stat
```

Look at the column `MSW` when playing Notes

```
$ gdb --args sushi_b64 --debug-mode-sw -r -c configs/sushi_config_msw.json
$ (gdb) catch signal SIGXCPU
$ (gdb) run
# … should break on mode-switch
$ (gdb) bt
```

Cross-Compilation

If you modify the example plugin, zip the `Source` folder contents, and upload it to this page: http://elkbuild.local:5000/

Add your board address and target path
`root@elkpi-X.local:/target/path`

That server will cross-compile it, and directly place it on your board in the location you've selected.


Thank you!