Pipewire as the heart of Linux-based audio systems

Embedded Recipes Paris - September 28, 2023

Philip-Dylan Gleonec

philip-dylan.gleonec@savoirfairelinux.com

A bit of context

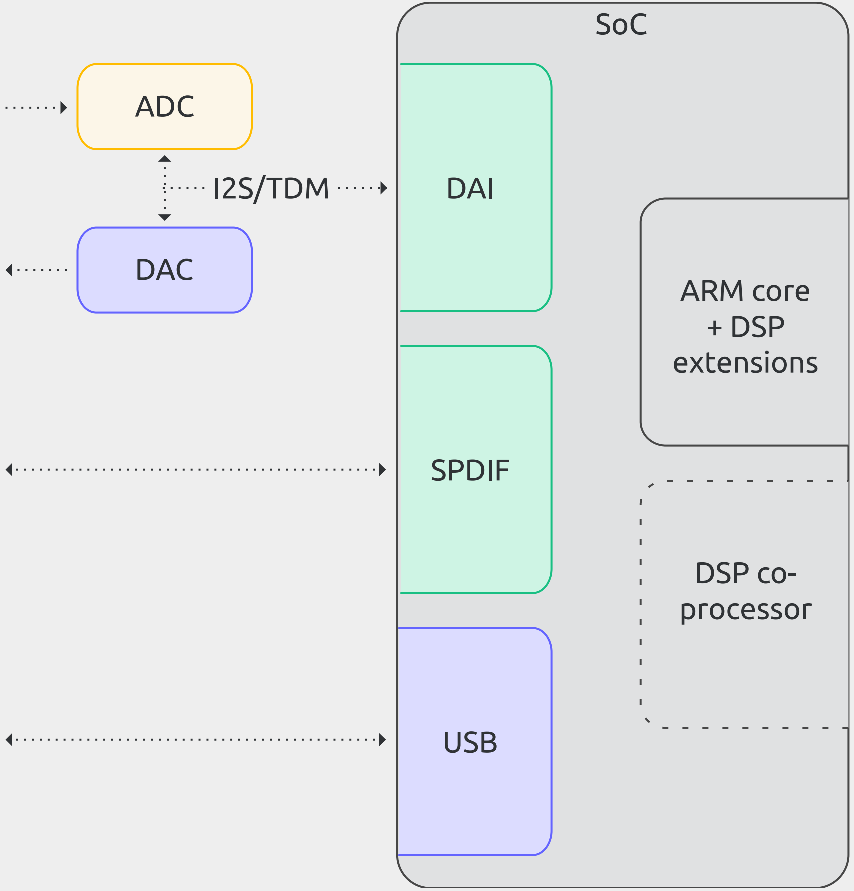
What makes audio complex?

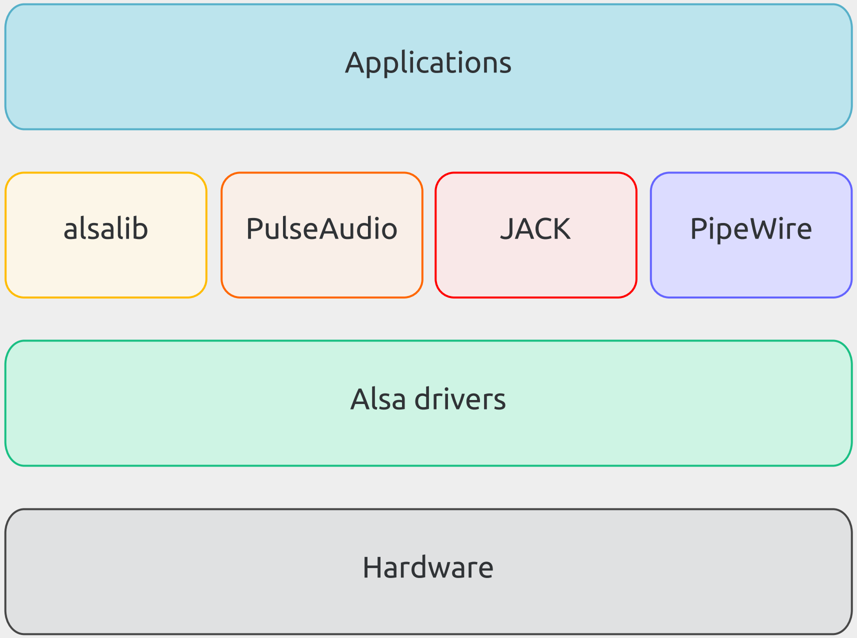
Src: https://avitvision.es/en/biamp/parle-barras-conferencia

- Multiple asynchronous audio interfaces
- Advanced algorithms integration : AEC, noise reduction...
- Real-time processing requirements

Recent SoC advances can make this cheaper

- Integration of DSP extensions in instruction sets reduce need for dedicated ICs
- Interfaces are directly integrated
- Larger developper pool
- More open source resources
- Linux enables cheaper dev and short time-to-market





ALSA

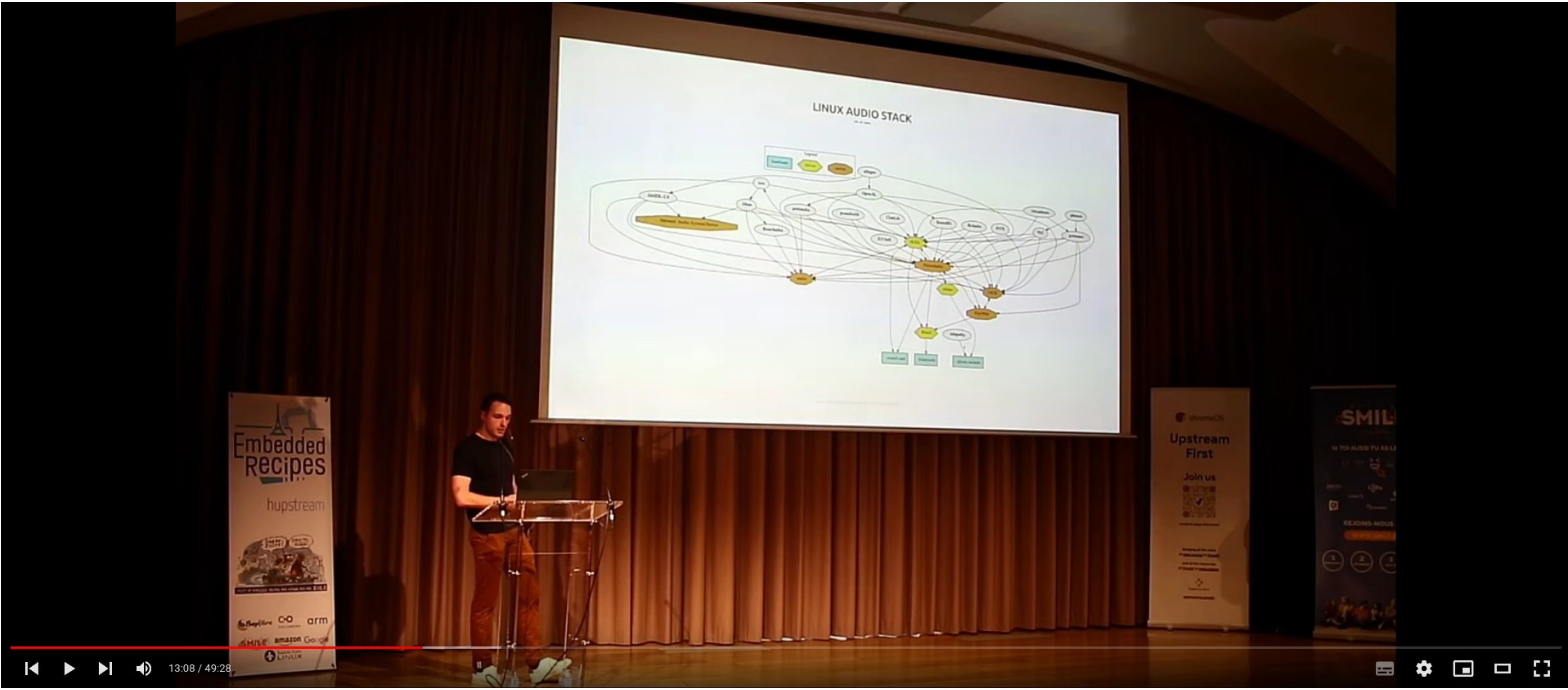
- Linux kernel audio API
- Data and control drivers
- Exclusive access to devices

Sound servers

- JACK, PulseAudio, PipeWire…
- High level audio API
- Concurrent access to devices

Applications

- Use-case specific



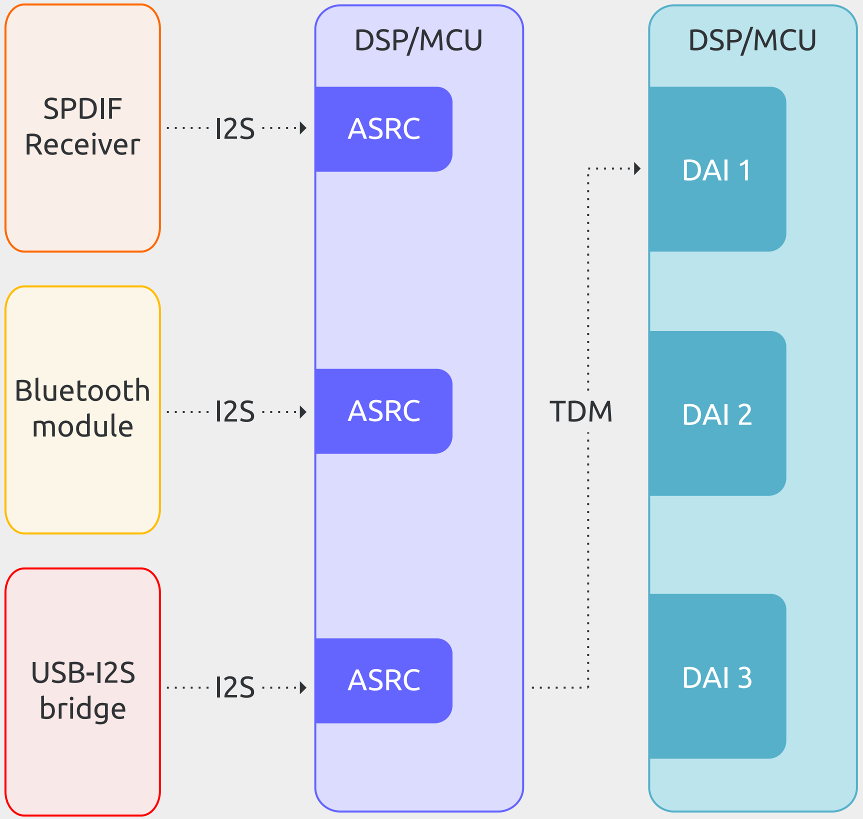
Src: https://youtu.be/ig7MxYi3Bmw?si=f9CfXrEZSTcw3oh3

- Goal : we want a single synchronous interface

- Synchronizing asynchronous sources is expensive
  - Either in CPU resources
  - Either in dedicated hardware (DSP, ASRC...)

- Exposing a single interface is expensive
  - Either in custom machine driver development
  - Either in dedicated hardware (DSP, MCU, FPGA…)

- Specific interfaces might require more work
  - USB audio gadget
  - Bluetooth
  - Audio over network



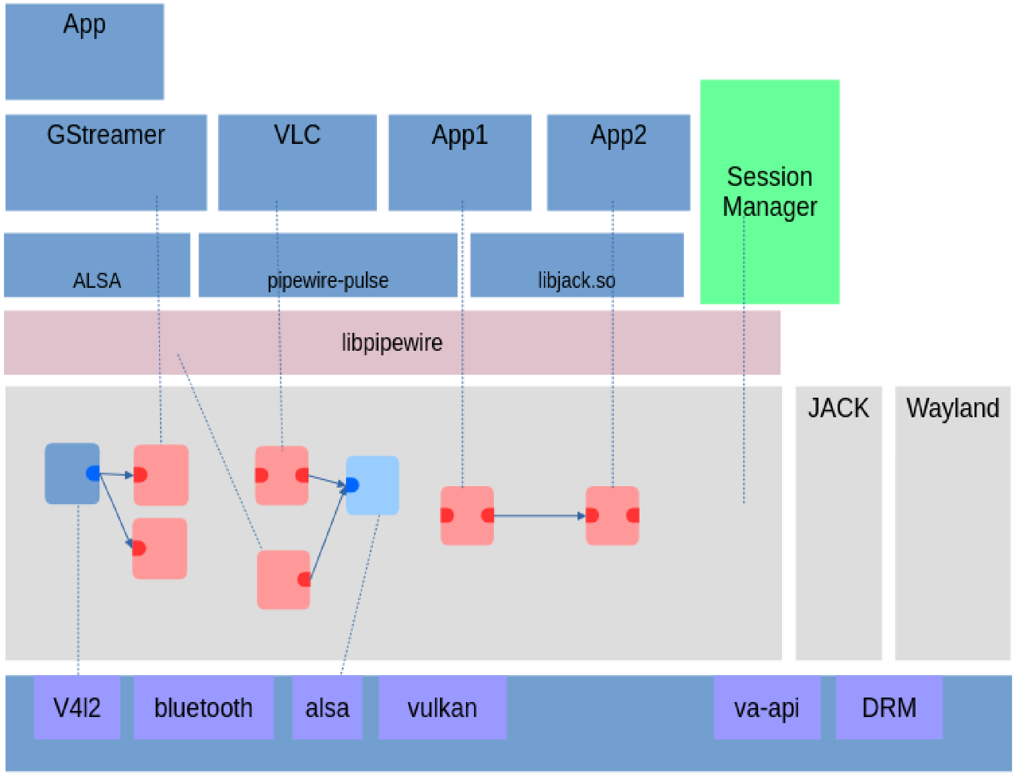
How Pipewire nearly obsoleted my previous talk

This work is based on our intern's and apprentice's results

Elinor Montmasson

Le Bao Tin Ha

Pipewire



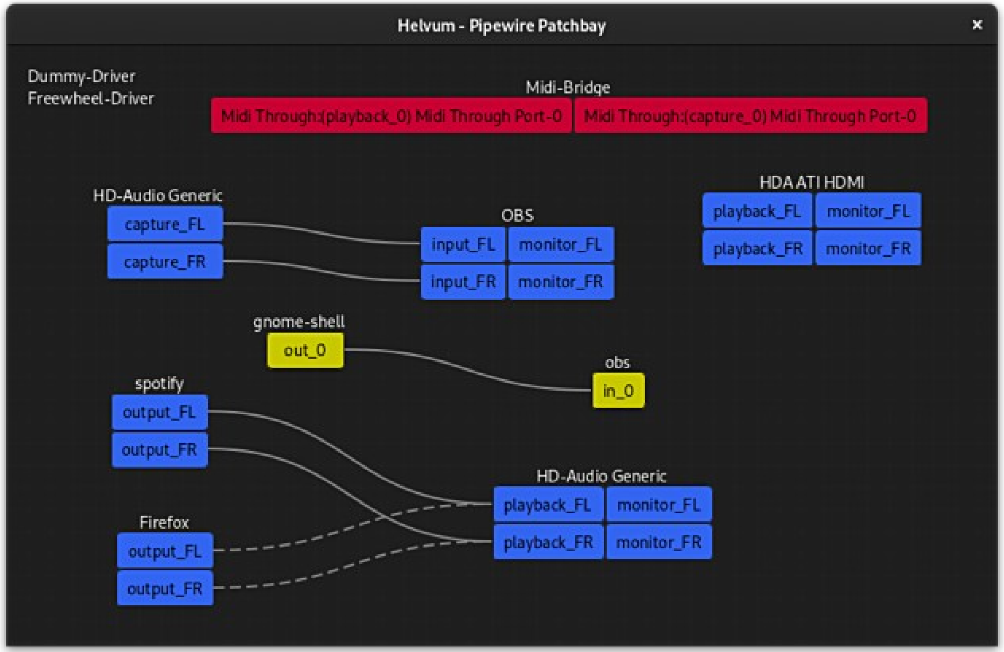
- Pipewire is a multimedia server
- Started in 2015 by Wim Taymans
- Offers a flexible graph approach for multimedia
- Offers a highly modular and extensible daemon
- Able to achieve very low latencies

Src : Taymans, Wim. “PIPEWIRE: A LOW-LEVEL MULTIMEDIA SUBSYSTEM.” (2020)

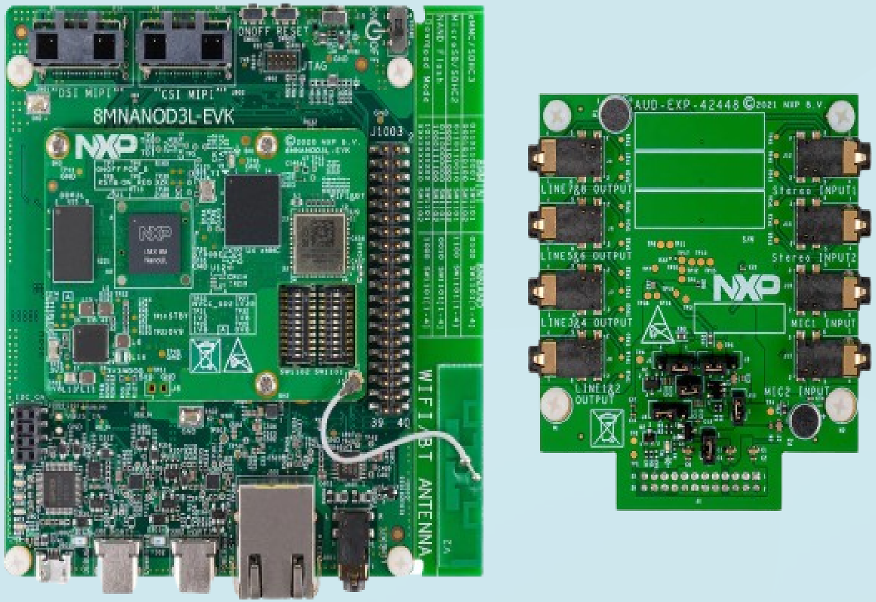
Pipewire

- Why Pipewire for audio?
  - API compatible with JACK and Pulseaudio servers
  - Able to meet real-time requirements
  - Support for Gstreamer applications
  - Active development and community

- Open questions
  - What is the performance for embedded?
  - Which problems does it solve?
  - What new features can it bring?



Src: https://gitlab.freedesktop.org/pipewire/helvum



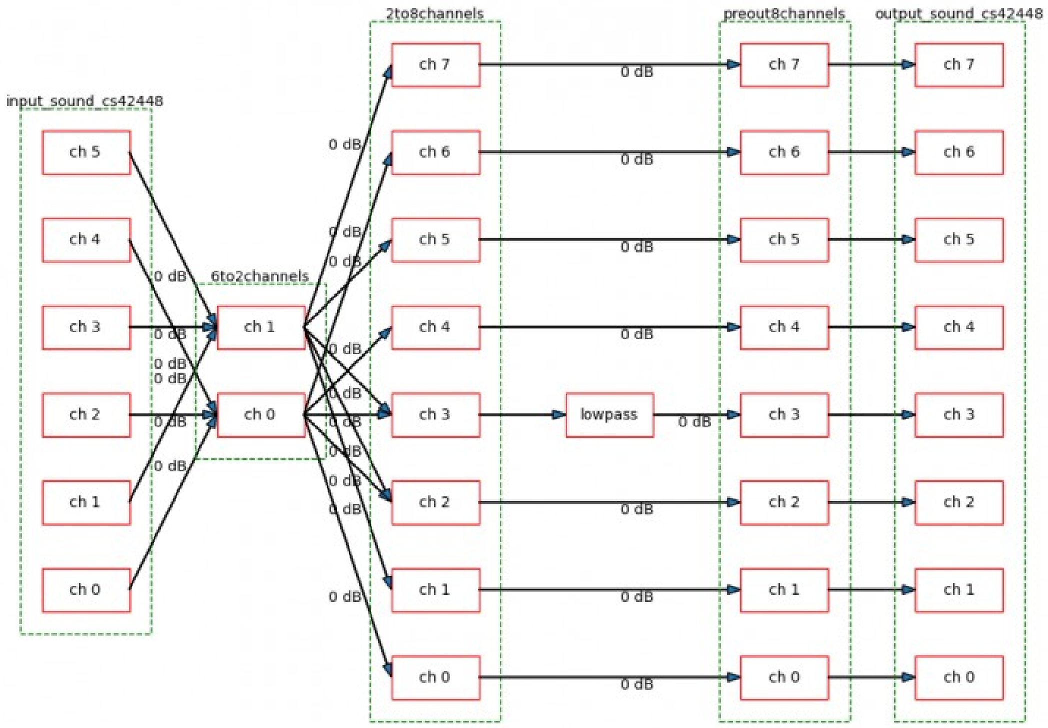
PROJECT

Src: https://nxp.com

Pipewire performance evaluation

- Evaluated on i.MX8M Nano EVK + CS42448
  - One codec, no asynchronous interfaces
- Distribution based on Yocto Kirkstone
- Kernel linux-imx v5.4
- Use-case : karaoke using CamillaDSP framework
  - Developped in Rust
  - DSP Pipeline from configuration file
  - Supports both JACK and Pulseaudio backend
- Measurements done :
  - CPU consumption
  - Latency

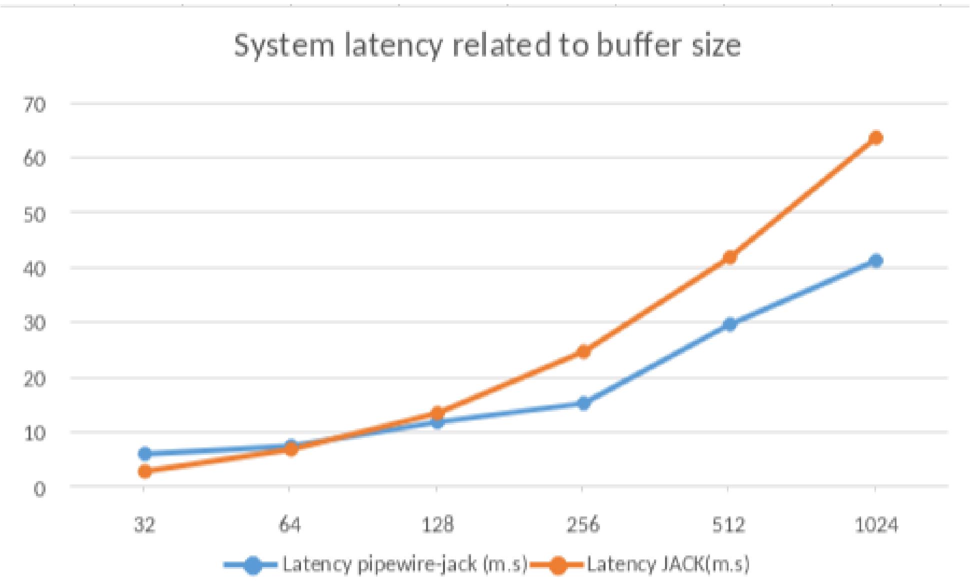
Pipewire performance evaluation



Src: https://blog.savoirfairelinux.com/en-ca/2022/pipewire-in-linux-embedded-project-a-multi-ports-audio-system-demo-on-i-mx8-part-1/

Pipewire performance evaluation

Backend	**CamillaDSP load**	**Server load**
JACK	28%	18%
Pipewire	16%	14%



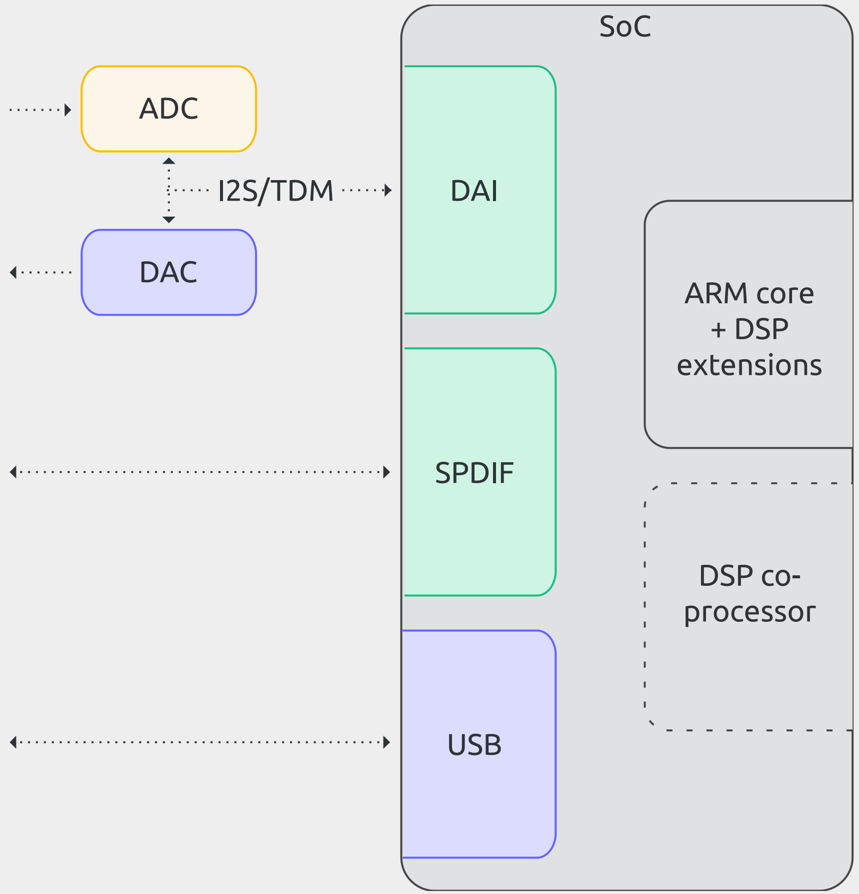
We observe both latency and CPU load reduction!

Src: https://blog.savoirfairelinux.com/en-ca/2023/pipewire-in-linux-embedded-project-a-multi-ports-audio-system-demo-on-i-mx8-part-2/

Pipewire performance evaluation

- 60ms latency measured with Pulseaudio
- → Pulseaudio is not able to compete in this space
- To make measurements « fair », we configured Pipewire and JACK to increase their latency to 60ms
- Pipewire is better than JACK, but worse than Pulseaudio
- Pipewire needs an external Pipewire-pulse process, which is CPU consuming
- Possible configuration is needed for clean Pulseaudio compatibility

Backend	**CamillaDSP load**	**Server load**
JACK	26.1%	14.1%
Pulseaudio	26.7%	35.1%
Pipewire (JACK API)	14.3%	10.5%
Pipewire (Pulseaudio API)	35.6%	41.7% (17.4% PW + 24.3% PW-pulse)



Pipewire performance evaluation

- The previous results highlighted Pipewire's good performance
- The case of asynchronous devices has not been tested
  - We are adding SPDIF and USB gadget interface
- Pulseaudio and JACK resample other interfaces
  - Pipewire sources and sinks can be configured with the clock.name property to be marked synchronous

→ Question : What is the impact of asynchronous devices on Pipewire?

Pipewire performance evaluation

- CPU load measurements have been measured with htop and perf
  - Htop measures global CPU usage
  - Perf measures CPU usage by functions
    - do_resample_full_<type>()
    - do_resample_inter_<type>()
    - do_resample_copy_c()

- 4 scenarios have been evaluated

		CS42448 samplerate	**USB gadget audio samplerate**	**Pipewire graph samplerate**
USB	CS42448 loopback 1	48kHz	**44.1kHz**	48kHz
USB	CS42448 loopback 2	48kHz	48kHz	48kHz
	CS42448 loopback 1	48kHz	NA	48kHz
	CS42448 loopback 2	48kHz	NA	**44.1kHz**

Pipewire performance evaluation

	CPU load with htop	Perf samples related to Pipewire	Perf samples related to resampling	Resampling CPU load
USB CS42448 loopback 1	50% ~ 55%	62.91%	33.37%	26.52% ~ 29.17%
USB CS42448 loopback 2	24% ~ 30%	31.65%	14.38%	7.60% ~ 9.50%
CS42448 loopback 1	9% ~ 12%	24.26%	No sample	0%
CS42448 loopback 2	29% ~ 31%	50.43%	25.73%	14.80% ~ 15.82%

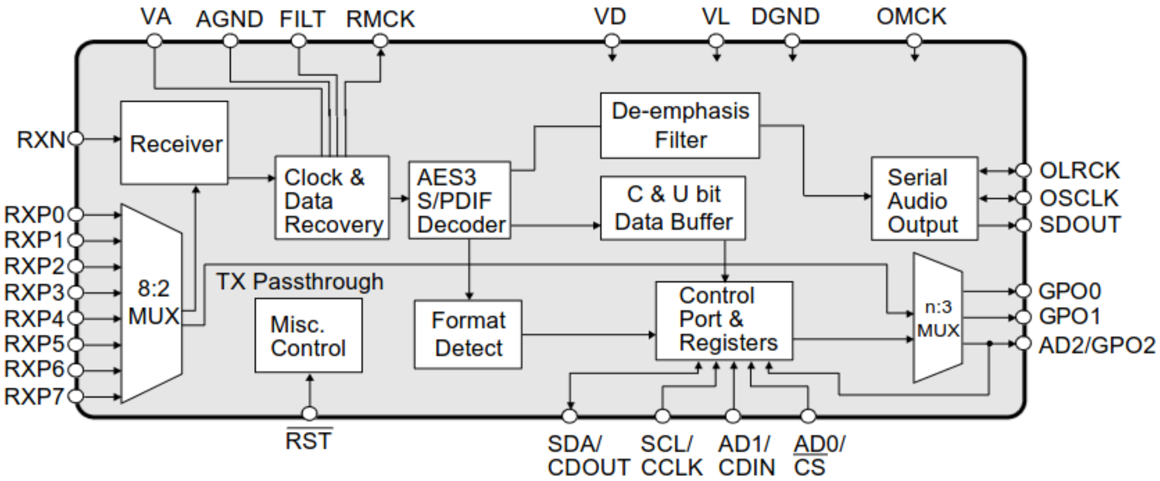
- Pipewire can efficiently avoid resampling for devices with similar clocks (cf. CS42448 loopack 1)
- Pipewire does resampling even if devices have the « same » clock (cf. USB / CS42448 loopback 2)
  - Pipewire can do resampling if needed in its graph frequency (cf. CS42448 loopback 2)

→ It's possible to reduce resampling with Pipewire

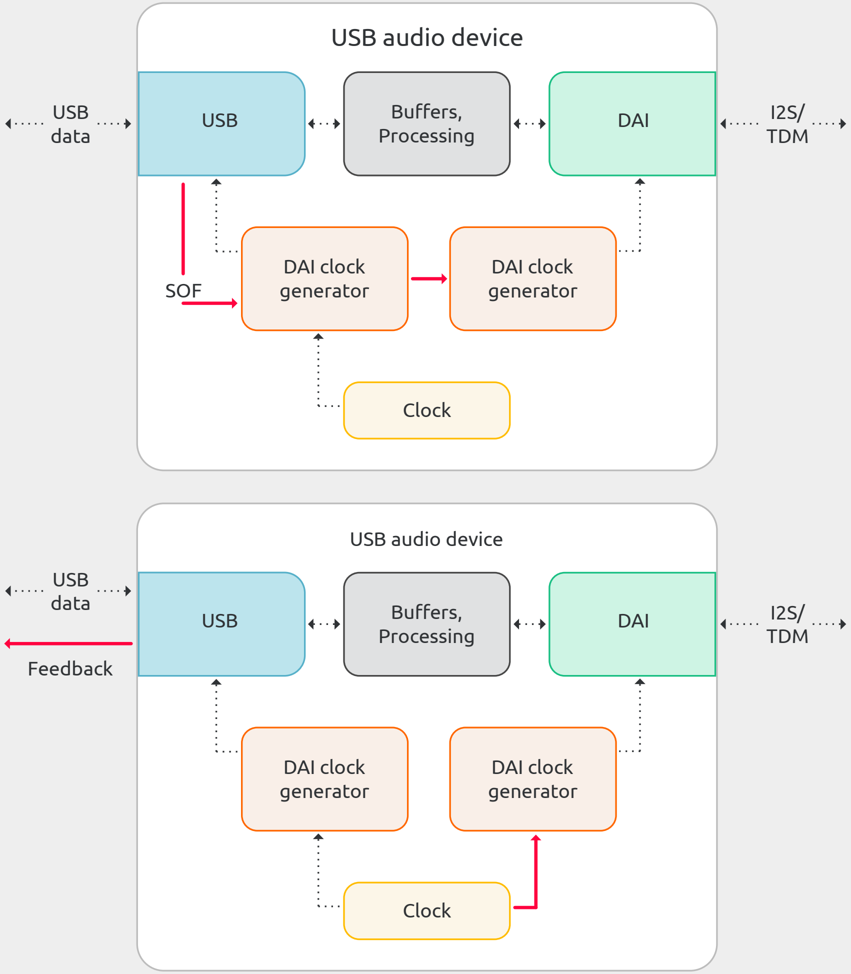
→ Configuration is key

Pipewire performance improvement

- ADCs and DACs can share a common clock
- Digital interfaces cannot
- Need to set-up synchronization
  - Dedicated hardware asynchronous sample-rate converter
  - USB asynchronous tranfers / feedback



Src: https://statics.cirrus.com/pubs/proDatasheet/CS8416_DS578F5.pdf



Pipewire performance improvement - USB

- Usual USB audio transfer uses isochronous transfers
  - Data is transfered at regular interval
  - Audio clock is extracted from this interval
- UAC class supports a feedback endpoint
  - The device signals hosts if it needs more/less samples
- UAC feedback endpoint is supported at the driver level
  - Need to upgrade to a more recent kernel
- This feature is only supported by alsaloop
- … and Pipewire since July

Pipewire performance improvement - USB

	CPU load with htop	Perf samples related to Pipewire	Perf samples related to resampling	Resampling CPU load
USB CS42448 isochronous	24% ~ 30%	31.65%	14.38%	7.60% ~ 9.50%
USB CS42448 asynchronous	20% ~ 23%	35.25%	No samples	0%

- We measured CPU gains on scenario USB / CS42448 loopback 2
  - Sample-rate must be at the same nominal frequency
- USB gadget configuration is set to sync type « async »
- Resampling is effectively removed
- Slight increase of Pipewire CPU usage

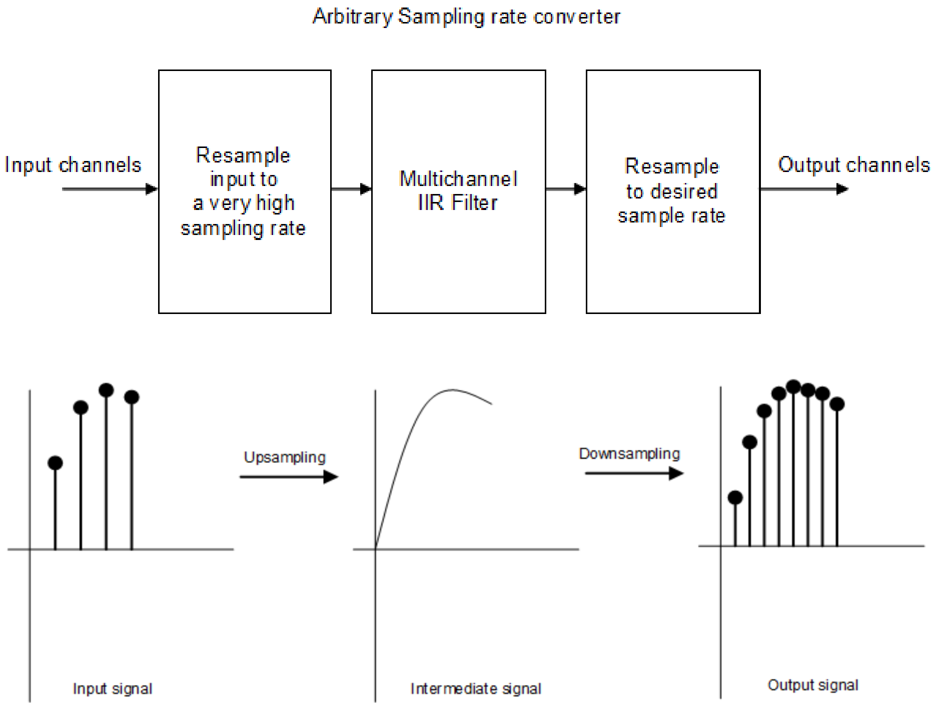
Pipewire performance improvement - ASRC

- ASRC filters are computation intensive

- Some SoC embed hardware ASRC
  - i.MX6, most i.MX8
  - SAMA7G5
  - ADSP-SC58 + ADSP-2158 series
  - Some RZ/G and RZ/A

- Need support in Linux
  - linux-imx specific µAPI for ALSA plugin
  - Machine specific driver integration in mainline

- Need to integrate ASRC with SPDIF
  - SPDIF is a DAI associated with dummy-codec



Src: https://ackspace.nl/wiki/Arbitrary_Sampling_Rate_Converter_in_VHDL

Pipewire performance improvement - ASRC

- Step 1 : add support for dummy-codec in fsl-asoc-card driver
- Step 2 : replace SAI by SPDIF controller as DAI
- Difficulties :
  - Need to adapt route selection to dummy-codec → done
  - Some noise issues with some codecs being investigated
  - No SPDIF / ASRC DMA scripts in i.MX SDMA firmware
- Correctly enables removal of resampling by Pipewire
- The goal is to contribute our solution back to mainline kernel

```
sound-wm8782-asrc {
        status = "okay";
        compatible = "fsl,imx-audio-dummy-codec";
        model = "wm8782-asrc-audio";
        audio-cpu = <&sai3>;
        audio-asrc = <&easrc>;
        dai-format = "i2s";
        frame-master = <&sai3>;
        bitclock-master = <&sai3>;
};
```

```
sound-spdif-asrc {
        status = "okay";
        compatible = "fsl,imx-audio-dummy-codec";
        model = "spdif-asrc-audio";
        audio-cpu = <&spdif1>;
        audio-asrc = <&easrc>;
        spdif-out;
        spdif-in;
};
```

Wrap-up

- How to develop a Linux embedded audio system has evolved

- Pipewire is ready for production use

- Pipewire can solve multiple problems for audio systems
  - Smarter resampling
  - Single interface through combine plugins
  - Proper USB audio support

- Pipewire can free CPU resources for processing

- → Pipewire can be the heart of a modern Linux audio system

Wrap-up

- What's the future for audio on Linux?

- Finish work on ASRC integration
  - i.MX specific solution
  - May take time to stabilize

- Linux Sound Open Firmware (SOF)
  - Offload audio processing to a DSP core
  - Historically only for Xtensa HiFi DSP
  - Recent switch to Zephyr OS
  - NXP has a port for Cortex-A
  - ST has interest for Cortex-M

Conclusion

- Audio systems development has evolved a lot over the last years
- DSP development can now be done on general purpose platforms
- More than ever, Linux is a prime candidate for audio systems implementation
- Proper co-design is still required
  - Pipewire gives more options
- Pipewire is a prime candidate for audio system implementations

Thank you for your attention

Philip-Dylan Gleonec

philip-dylan.gleonec@savoirfairelinux.com