
OPENAV
LAC 2026
OpenAV
PRODUCTIONS

"Reflection on OpenAV Productions Software"

+ Live looping performance software
+ Audio FX plugins
+ Synths (wavetables + routing)
+ USB controller support
+ Open source + $$ release model
+ Linux focussed


SORCER
OPENAV
Wave 1
Wave 2
Sub-Bass
LFO
W 1
W2
Speed
Amp
ADSR
A
D
S
R
Remove
Cutoff
Modulation
Master
Gain
Thres
A
R


FABLA
Waveform
Sample: Classic-808_Kick_long
Pads
13 Classic-808_
14 Classic-808_
15 Classic-808_
16 Classic-808_
9 Classic-808_
10 Classic-808_
11 Classic-808_
12 Classic-808_
5 Classic-808_
6 Classic-808_
7 Classic-808_
8 Classic-808_
1 Classic-808_
2 Classic-808_
3 Classic-808_
4 Classic-808_
OPENAV
Source
A
D
S
R
VOL
SPD
PAN
Master
Ratio
Thres
Atk
Rel
OPENAV
W1
W2
Speed
Amp
ADSR
A
D
S
R
Remove
Cutoff
Modulation
Master
Gain
Thres
A
R

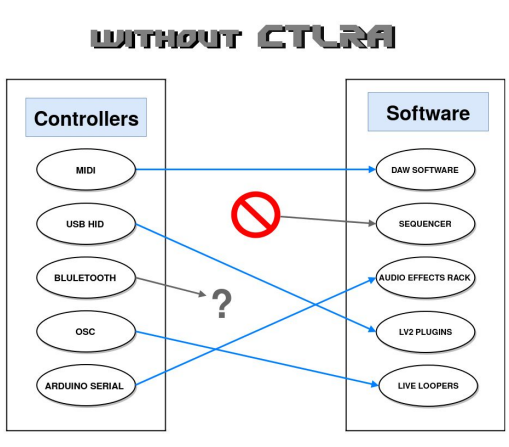

LUPPP
OPENAVPRODUCTIONS
Drums
Sorcer
Fabla Pads
Fabla Bass
Guitar (Mic)
Cajon (Mic)
Voice (Mic)
Track 8
Master
Snd
Key
Mix
Scene 1
Scene 2
Scene 3
Scene 4
Scene 5
Scene 6
Scene 7
Scene 8
Scene 9
Scene 10
Snd
Key
XRec
Tap
Metro
BPM
Return
OPENAV
A
D
S
R
Master
Gain
Thres
A
R

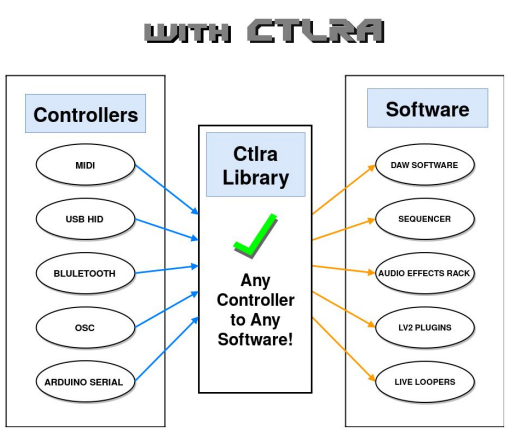

openAVproductions
LUPPP
Drums
Sorcer
Fabla Pads
Fabla Bass
Guitar (Mic)
Cajon (Mic)
Voice (Mic)
Track 8
Master
Snd
Key
Mix
Scene 1
Scene 2
Scene 3
Scene 4
Scene 5
Scene 6
Scene 7
Scene 8
Scene 9
Scene 10
Tap
BPM
BITTA
Crush
DELLA
Feedback
Volume
Time
DRIVA
Odie
Tone
Amount
DUCKA
Thresh
Drop
Time
FILTA
Frequency
KUIZA
Low
Mid
Mid
High
Gain
MASHA
Time
Pass
Vol
PANDA
Threshold
Release
Factor
ROOMY
Time
Damp
Dry Wet
SATMA
Distortion
Tone
VIHDA
Width
Invert
WHAAA
Freq
Drive
Mix
OPENAV
A
D
S
R
Master
Gain
Thres
A
R


LUPPP
Drums
Sorcer
FABLA 2.0
PANIC
REC
openAVproductions
Track 8
Master
OPENAV
36-Ludwig26Kick-5.wav
Play
Mute
BITTA
ARTYFX by OPENAV
Crush
Feedback
Volume
Time
Tone
Amount
Thresh
Drop
Time
MASHA
Time
Pass
Vol
PANDA
Threshold
Release
Factor
ROOMY
Time
Damp
Dry Wet
SATMA
Distortion
Tone
VIHDA
Width
Invert
Freq
Drive
Mix


LUPPP
Drums
Sorcer
FABLA 2.0
A
B
C
D
PANIC
Fade
13
14
15
16
10
OPENAVPRODUCTIONS
Track 8
Master
OPENAV
OPENAV
Master
NINJA
OPENAV
LFO 1
LFO 2
OSC 1
Addy
Filter
Master
Routing
OSC 2
Addy
Adsr 3
Effects
ADSR 1
ADSR 2
OSC 3
Addy

Ctlra
(Hw Acccess)

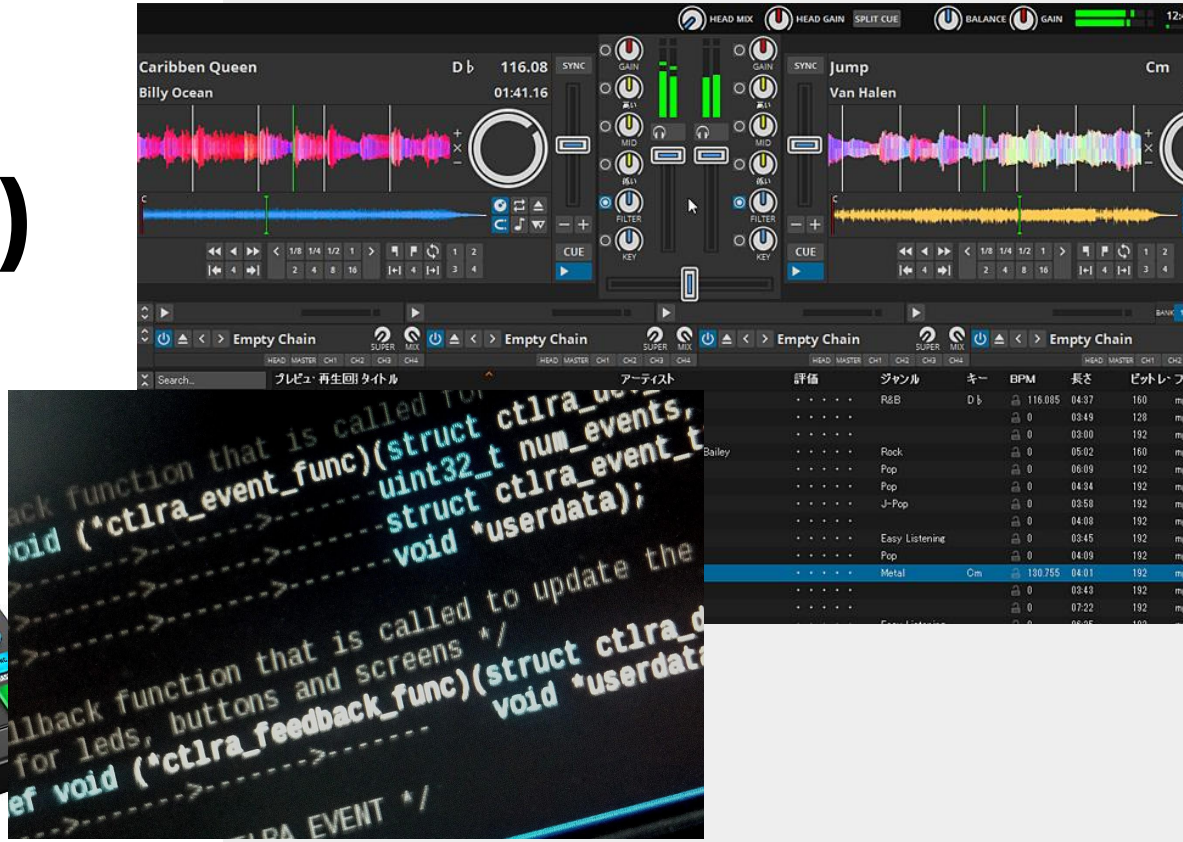




"Next Gen" Controllers!

- USB HID devices
- "HD" Screens
- Huge functionality!
- Application support...

Mixxx
(Ctlra branch)



3%

Of all of the ideas I'd like to build...

motivation, time, and skills!?

Community!

Motivation & Learning

2009

2010

2011

2012

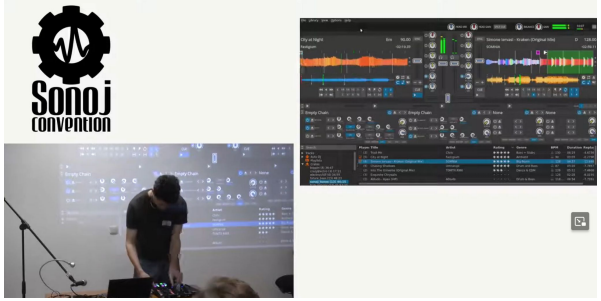
2013

2014

2015

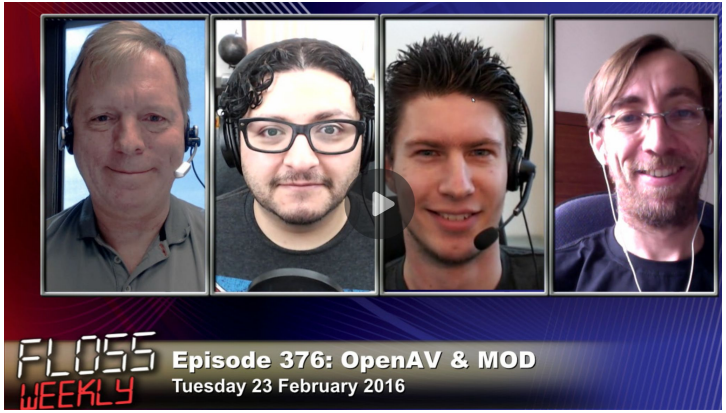
2016 "minilac"

2017



SONOJ '17

2018



MOD

[Floss Weekly]

[MOD Device Collabs]

[Ubuntu User Mag]

[OpenAV Release System]

! LLMs !

"So what" for Linux Audio?

Ctlra code review

📊 SUMMARY

Severity	Count	Issues
🔴 Critical	5	Memory leaks, use-after-free, buffer overreads, NULL dereferences
🟡 High	5	Resource leaks, buffer overflows, missing bounds checks
🟠 Medium	5	Integer overflows, inconsistent patterns, magic numbers
🟢 Low	5	Code quality, documentation, dead code

(while watching the keynote yesterday? Context switching…)

Mem leaks, UseAfterFree, Buffer Overflows!

"Modern" software engineering?

Review the RTFX codebase in order to understand what DSP digital signal processing errors there might be in the DSP code, as well as any software bugs or issues like C++ memory leaks, etc.

📋 SUMMARY OF FINDINGS

Severity	Issue	File	Line	Status
🔴 CRITICAL	Variable shadowing in BitCrusher	`dsp_bitcrusher.hxx`	85	MUST FIX
🔴 CRITICAL	Buffer overflow risk in Delay	`dsp_delay.hxx`	45-47	MUST FIX
🔴 CRITICAL	Logic error in Distortion	`dsp_distortion.hxx`	120	MUST FIX
🟡 HIGH	Hardcoded sample rate in AnalogFilter	`AnalogFilter.cxx`	38-40	SHOULD FIX
🟡 HIGH	Memory management inconsistency	`dsp_delay.hxx`	45-47	SHOULD FIX
🟡 MEDIUM	Unused BPM handling	`bitta.cxx`	95-105	NICE TO FIX
🟡 MEDIUM	Numerical precision constants	`global.h`	25-30	NICE TO FIX
🟢 LOW	Commented code in Waveshaper	`Waveshaper.cxx`	70-75	CLEANUP

- **CRITICAL BUGS SUMMARY - DSP Code Review** bug dsp enhancement
  #57 · harryhaaren opened 1 hour ago
- **HIGH: AnalogFilter Uses Hardcoded Sample Rate** bug dsp
  #56 · harryhaaren opened 1 hour ago
- **CRITICAL: Distortion DSP Logic Error - Always Processes Audio** bug critical dsp memory
  #55 · harryhaaren opened 1 hour ago
- **CRITICAL: Delay DSP Buffer Overflow Risk** bug critical dsp memory
  #54 · harryhaaren opened 1 hour ago
- **CRITICAL: BitCrusher DSP Variable Shadowing Bug** bug critical dsp memory
  #53 · harryhaaren opened 1 hour ago

Now create a PR with one commit for each fix, tag the fix appropriately with a "fix #<id>" to auto close it on merge.

✅ PR Created Successfully!

I've created a **draft Pull Request** with **4 individual commits**, each addressing one of the critical DSP bugs identified in the code review.

Technical Stuff

(Here be language wars ;)

Best tools for the job?

Something that doesn't let me make "silly" mistakes

- Silly mistakes in complex code-bases aren't silly
- But they are still mistakes, and cause bad UX/crashes…

how many of these bugs would exist if the code was written in Rust?

Rust's Safety Guarantees vs. C Bugs

🔴 CRITICAL BUGS - All Prevented in Rust

Bug	Rust Prevention
1. Memory Leak in `ctlra_dev_connect()`	✅ **Prevented** - Rust's ownership system ensures every allocation has catch if a value is allocated but not consumed.
2. Use-After-Free in `ctlra_dev_disconnect()`	✅ **Prevented** - Rust's borrow checker prevents using a value after it' `dev.disconnect(dev)` consumes `dev`, you cannot use it again.
3. Buffer Overread in `ctlra_get_vendors()`	✅ **Prevented** - Rust slices (`&[T]`) carry their length and bounds are bounds-checked.
4. NULL Pointer Dereference in `ctlra_impl_accept_dev()`	✅ **Prevented** - Rust has no null pointers. `Option<T>` is used instead, before dereferencing.
5. Resource Leak in Device Connection	✅ **Prevented** - Rust's `Drop` trait ensures resources are cleaned up w would be wrapped in types that implement `Drop`.

💡 The Bottom Line

Approximately 65% of these bugs (13 out of 20) would be prevented at compile time in Rust.

The remaining 35% would be caught by Rust's tooling (Clippy, rustdoc, compiler warnings) during development.

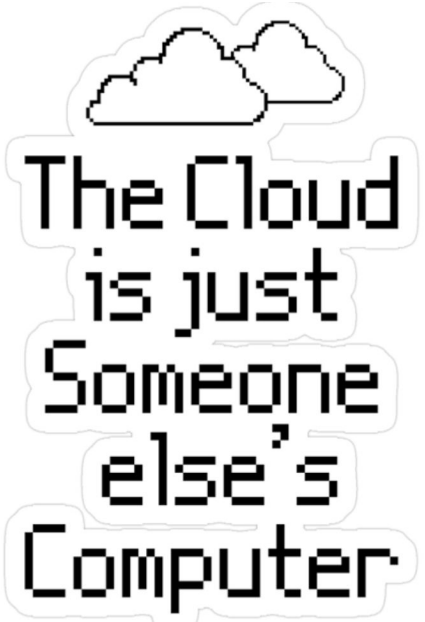
Zero of these bugs would make it to production in idiomatic Rust code.

Owning AI

... or depend on the cloud?

Owning AI

… or depend on the cloud?



Local GPUs?

Run LLMs locally on an open-source stack

- **~30 billion models locally : CPU only, or GPU**
- **~16 VRAM GPU is enough - more is better!**

Qwen

Gemma 4

llama.cpp

Communities & GPUs?

- **Community centric LLM inferencing**
  - Hackerspaces, "makerspaces", whatever :)
- **AI Models : new ones every day?**
  - Focus on "harness" not on model
  - Build workflows, do not focus on "prompting a model"
- **Play, Learn, Have fun**
  - Read blogs/newsletters, share experience

Positive Outlook

- **LLMs give new opportunities**

  … "Mirrors of operator skill"

- **Rethinking SW engineering**

  … building "agentic loops"

- **Own the hardware**

  … own your coding agency

Links to https://ghuntley.com/

Redo OpenAV?

Start in 2026 (not 2009)

With LLM power

So what would I do exactly the same?

+ Keep attending LAC :)

  Community is key

+ Attend other conferences (even more)

  Wider perspectives & skills

So what would I do differently?

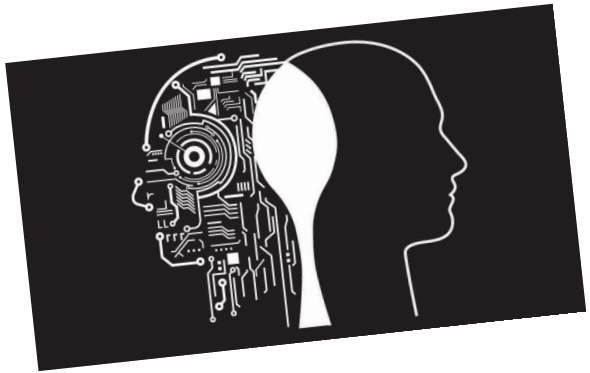
+ **Code in Rust**
  + 90% of "silly" bugs go away!
+ **LLM (local?) to review every commit**
  + Avoid "oopsie" logic errors
+ **More test/spec-driven development**
  + Suits LLM agentic loop building workflows

Community : Attendee Signatures in '14

! Thanks !

? Discussion ?

OpenAV



Bonus Slide

📊 Summary: Rust Prevention Rate

Category	Total Bugs	Prevented in Rust	Caught by Tooling	Still Possible
🔴 Critical	5	**5 (100%)**	0	0
🟡 High	5	**4 (80%)**	1	0
🟠 Medium	5	**3 (60%)**	2	0
🟢 Low	5	**1 (20%)**	4	0
TOTAL	**20**	**13 (65%)**	**7 (35%)**	**0**