Latency reduction in Video streaming with Linux’s camera and encoder APIs

Tl;dr -> Plan9 was right.

Tim Panton - tim@pi.pe @steely_glint:matrix.org @steely-glint@chaos.social #FOSDEM2026

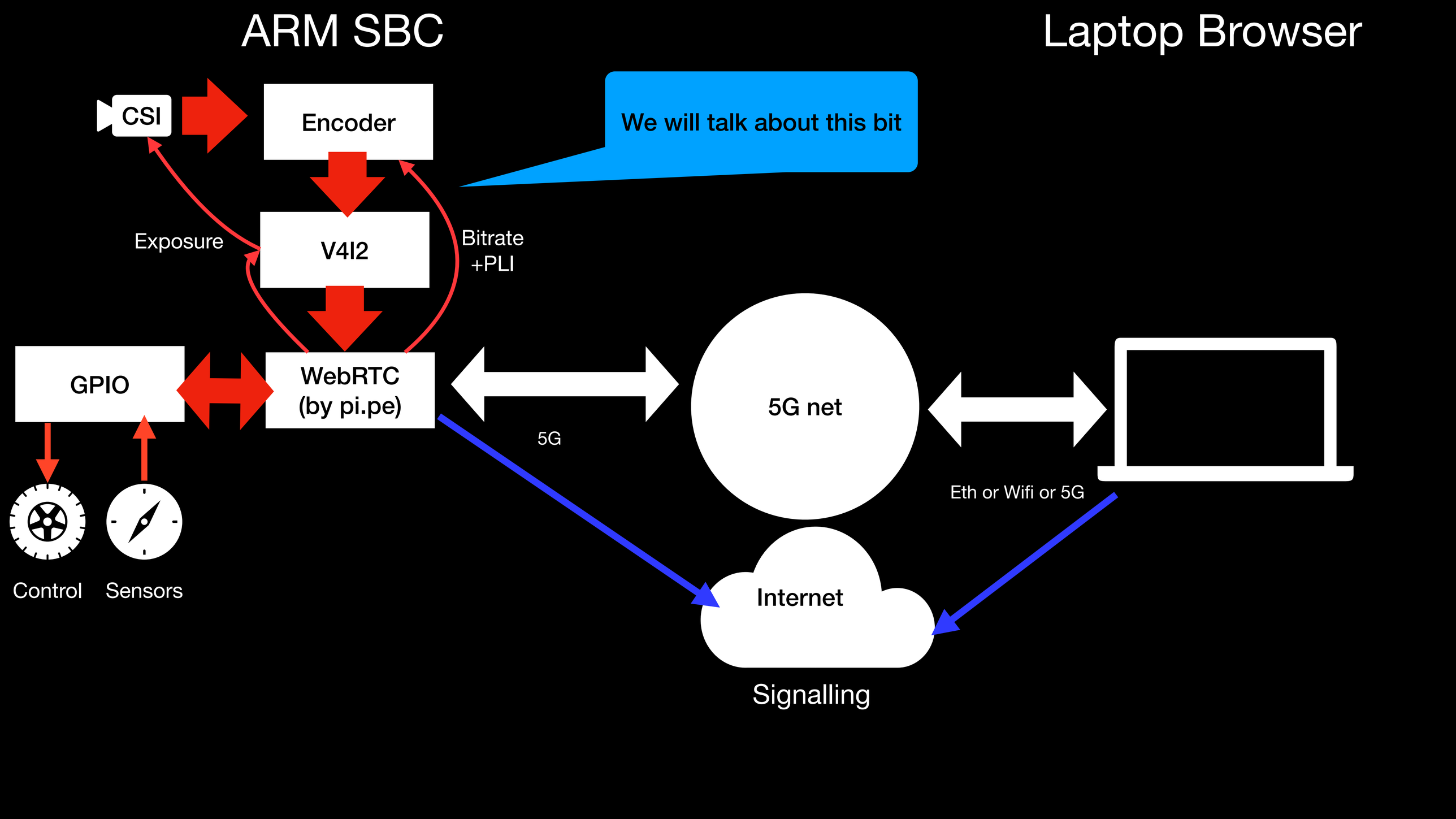
Tim Panton
CTO
pi.pe GmBH

Licenses a WebRTC stack for small cameras
Some of which move quite fast

Subject of the talk

Race Car Camera

- Over drivers shoulder
- Audio and Video to pit/trailer VIPs/supporters
- High quality
- Low latency
- Long range
- High speeds
- Public 5g networks


ARM SBC
Laptop Browser
CSI
Encoder
We will talk about this bit
Exposure
V4l2
Bitrate
+PLI
GPIO
WebRTC
(by pi.pe)
5G
5G net
Eth or Wifi or 5G
Control
Sensors
Internet
Signalling

Step 1- lab prototype

Gstreamer on Raspi

- Gstreamer pipeline
  - Read from v4l2 camera
  - Encode to H264
  - Packetize to RTP
  - Send to localhost
  - RTP read by |pipe| java stack (srtplight)
  - Encrypts and sends DTLS/SRTP
- Pros:
  - Simple - good isolation
  - Works in Lab
- Cons:
  - Fails on variable networks (5G)
  - Have to send keyframes often

https://github.com/steely-glint/srtplight

Isolation types

A side note

- Process isolation - runs in different process
- Memory isolation - we are using a memory safe language when possible
- License isolation - we may have proprietary algorithms for vehicle behaviour (intensely competitive space!)

All of these are desireable

Step 2 - wifi

Rpicamsrc gstreamer

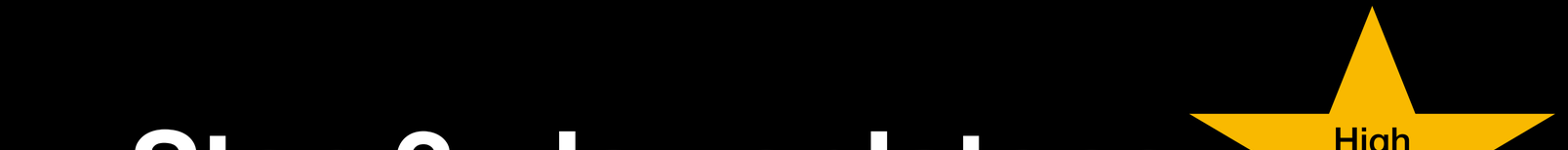
- Rpicamsrc
  - Gstreamer node
  - Talks to hardware encoder+camera
  - Replaced the v4l2 cam src
- Wrapped in 40 lines of C to set
- Exposes bitrate and full frame pads
- Exced from Java - which sends ASCII

- Pros:
  - Copes with Variable bitrate
  - Simple to test ASCII
  - Good isolation
- Cons:
  - It is a pipeline ~100ms latency
  - Every packet goes via localhost

https://gstreamer.freedesktop.org/documentation/rpicamsrc/index.html

Step 3 - lower latency

Pure v4l2



- V4l2-ctl
  - Put camera into H264 mode
  - Control bitrate
  - Request full frames
- Read frame from /dev/video0
- Packetize H264 in Java
- Exec v4l2-ctl when needed

- Pros:
  - Lower latency
  - Clean interface (Filesystem)
  - Superb isolation
- Cons:
  - Only works with Broadcom blob
  - Raspi deprecated it

https://www.rfc-editor.org/rfc/rfc3984

Step 4 - New Hardware - Khadas Vim4 Amlogic a311d2

back to Gstreamer with a hack (sigh)

- AML encoder GStreamer node does not support dynamic bitrate control or fullframe requests
- Couldn’t find source that would build
- Did a hack….
  - Found the source to .so Gstreamer uses
  - Tweaked it so it has a 2 byte shared memory seg
  - .so reads this before encoding each frame
  - Sets bitrate and/or fullframe
- Java opens memory seg as a Random Access file
- Writes to it when needed

- Pros:
  - M2 socket + Hardware encoder (unlike pi5)
  - Works
  - Acceptable isolation
- Cons:
  - Higher latency
  - Ugly

https://github.com/pipe/encoder_libs_aml_vim3n4

Step 5 - In process

Vim4 - back to reading from V4l2

Where we are now

- Uses Java FFM (NOT JNI) to access encoder .so
  - Sets bitrate / fullframe flag
  - Passes video frames in, gets H264 out
- But we can’t read() from /dev/video50
  - Have to use v4l2 shared memory buffers
  - Ioctls called via FFM

- Pros:
  - Lower latency (170ms G2G)
  - Pure Java - no C to maintain
  - No need to change shipped .so
- Cons:
  - Limited isolation (FFM and ioctl())
  - More code

https://github.com/pipe/v4l2Reader

Step 6 - Inprocess call .so

Vim4NPU - V4l2 ioctls unavailable

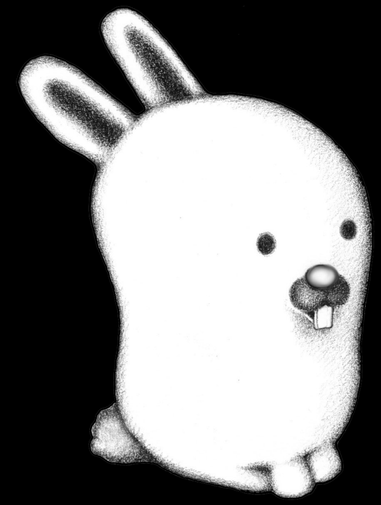
Future

- Encoder remains same
- Switched from V4l2 to libMedia
  - V4l2-ctl doesn’t
  - Shared memory buffers via .so
  - Multiple method calls to setup and run
  - .so compiled in C++ so have to mangle names
  - LD_PRELOAD needed

- Pros:
  - Android compatible ?
- Cons:
  - Very limited isolation
  - Lots more ugly code
- Why!?!?
- Working with Khadas to improve this…

<GRUMBLE>

Plan 9 OS generalizes the Unix principle of “everything is a file" to …. everything



Plan 9 from Bell Labs

- I miss that attitude
- Allows multiple languages to access the same API
  - A C++ .so really does not
- Driver writers do the thing once (arguably in the right place)
- How do we get back there ? (Will rust help or hinder?)

</GRUMBLE>

- Open source is great - I remember when we wrote a TCP/IP stack because it was cheaper than licensing Intel’s for their RTOS.

Thanks! Questions?

I’m a bit deaf, so SHOUT!

- Contact:
  - tim@pi.pe
  - @steely_glint@chaos.social
  - @steely_glint:matrix.org
- Consulting on open source WebRTC protocols
  - SRTP : https://github.com/steely-glint/srtplight
  - ICE : https://github.com/steely-glint/slice
  - SCTP : https://github.com/pipe/sctp4j
  - WHIP : https://github.com/pipe/whipi
  - G2G : https://github.com/pipe/G2G
  - Pion/gstreamer etc
- Building things with |pipe|

Image credits:
Sotherbys , Classic Driver , Car and Classic , EWRC and HDCarWallPaper