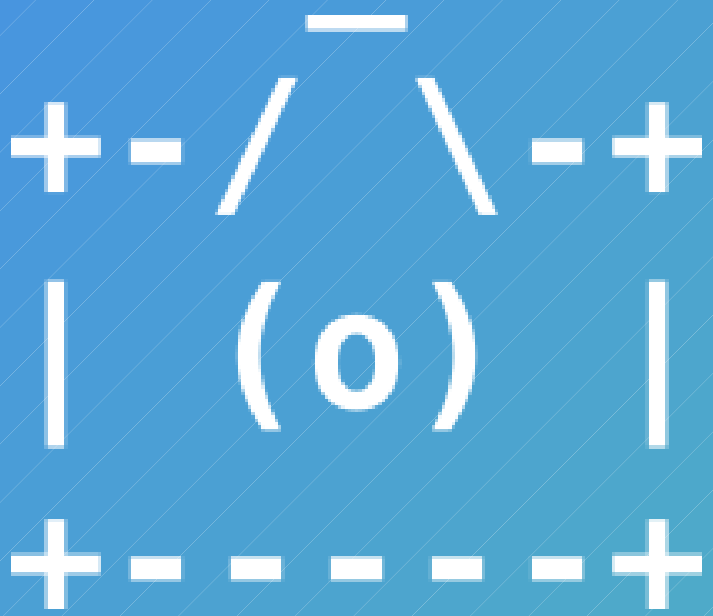


libcamera

Making complex cameras easy

FOSSASIA 2023
Singapore

Umang Jain
umang.jain@ideasonboard.com

Hello!

And welcome...

I'm Umang Jain

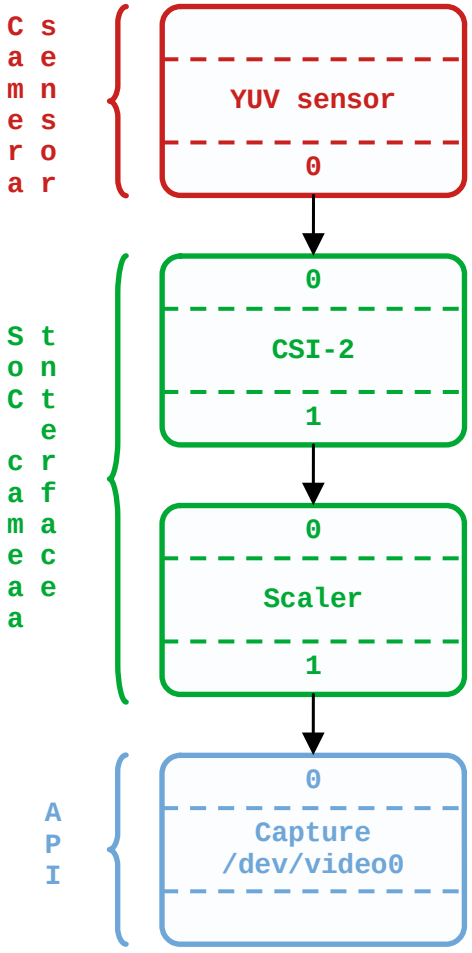
- Desktop turned Embedded developer
- Contributor to GNOME, OSTree, flatpak and immutable OSes
- Embedded, upstream linux-media and libcamera these days...

- Complex Cameras
- Challenges
- Complications
- Features & Developments
- Q & A

- This talk references many external projects, packages, and software.
- All logos, trademarks, and other identifying marks belong to their respective owners.
- In many places I'm talking about work that /other/ people have done or contributed
  - No intention or desire to take credit for their work

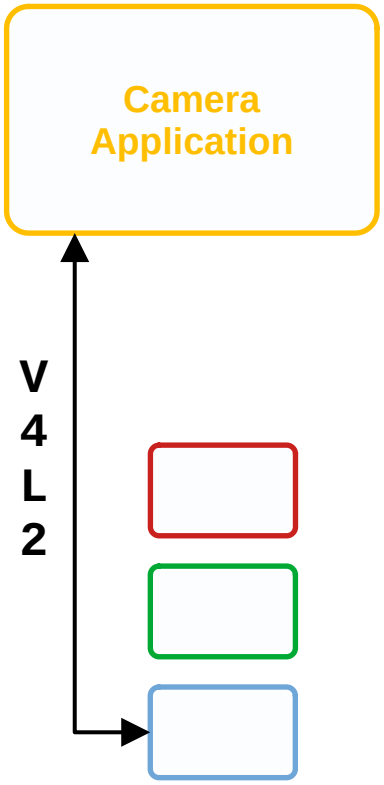
Legalese…

- **Complex Cameras**
- Challenges
- Complications
- Features & Developments
- Q+A



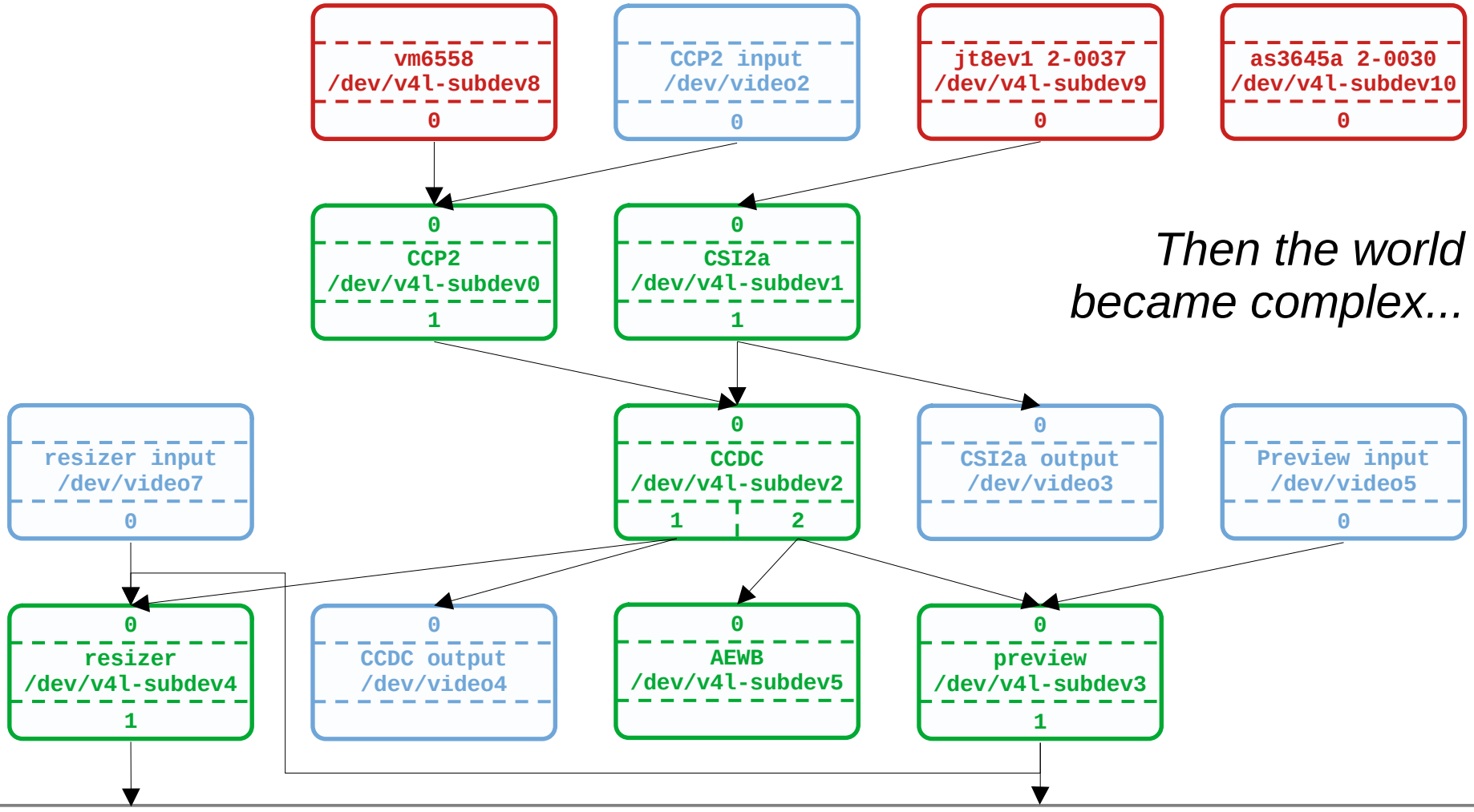
In the beginning were simple pipelines...

… and they were simple to control, with a single API.



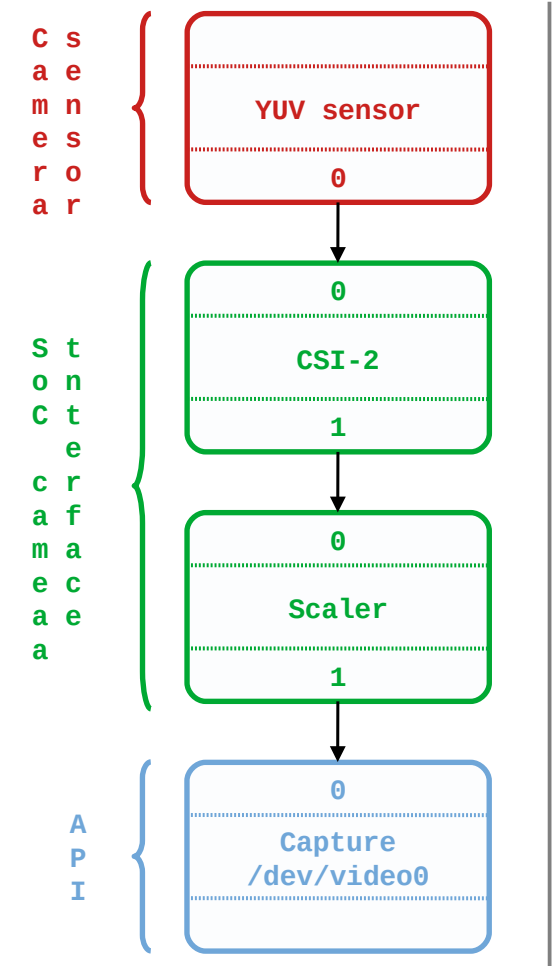
IDEAS ON BOARD

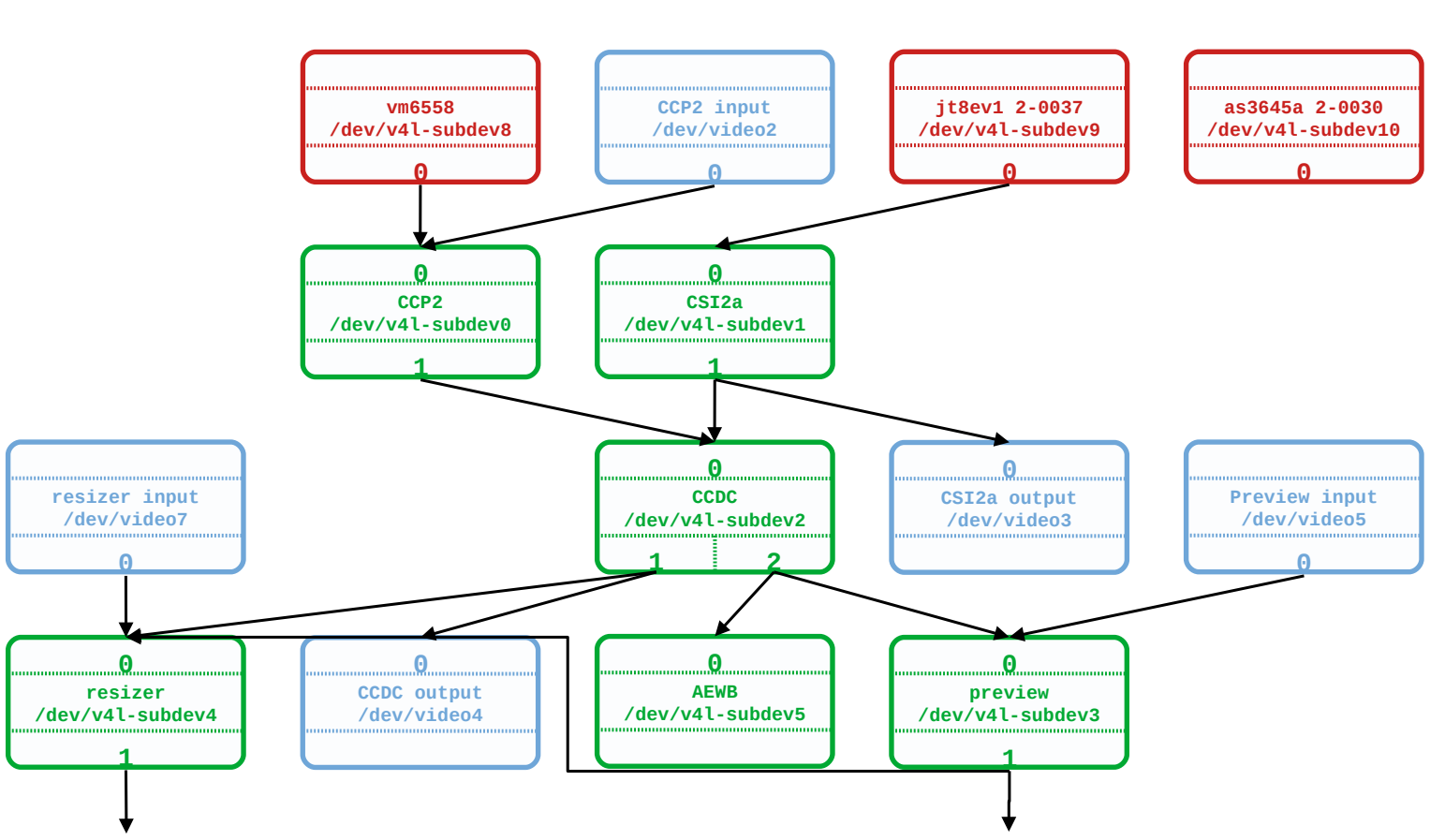
Then the world became complex



IDEAS ON BOARD

OMAP3 Camera in Nokia N900

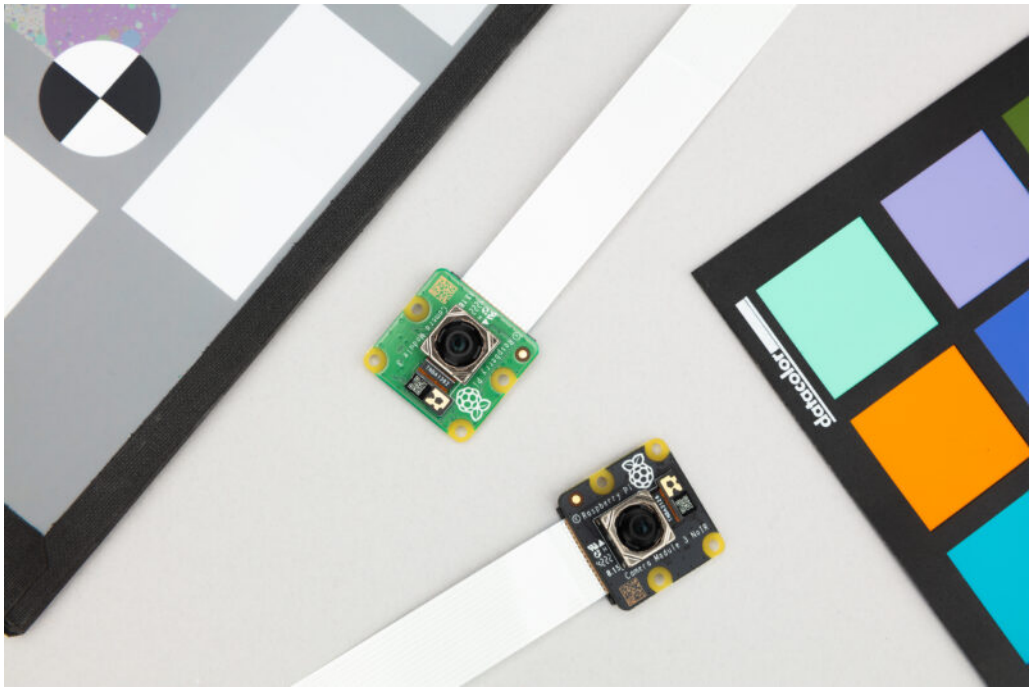




OMAP3 Camera in Nokia N900 - 2009

IDEAS ON BOARD

Cameras are Complex



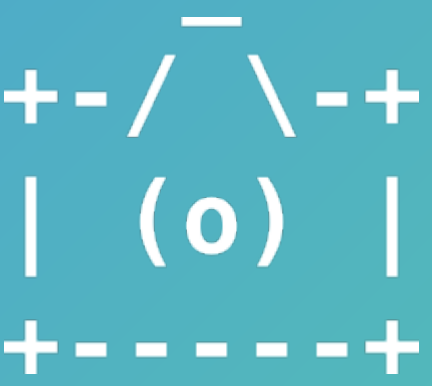
Raspberry Pi Autofocus Camera Module v3

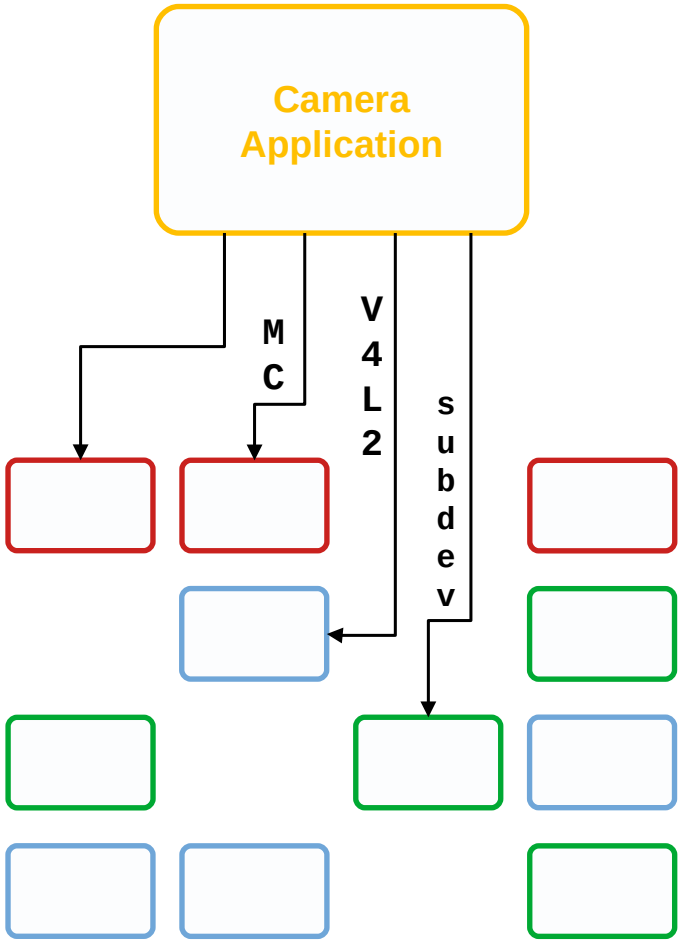
https://www.raspberrypi.com/app/uploads/2023/01/FOR-DEVELOPERS-800x533.jpg

IDEAS ON BOARD

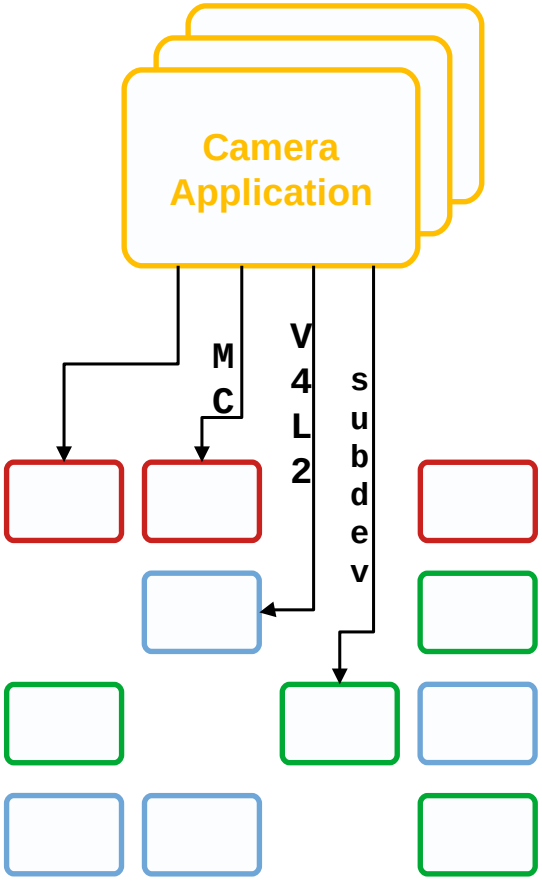
Cameras are Complex

- Complex Cameras
- **Challenges**
- Complications
- Features & Developments
- Q+A

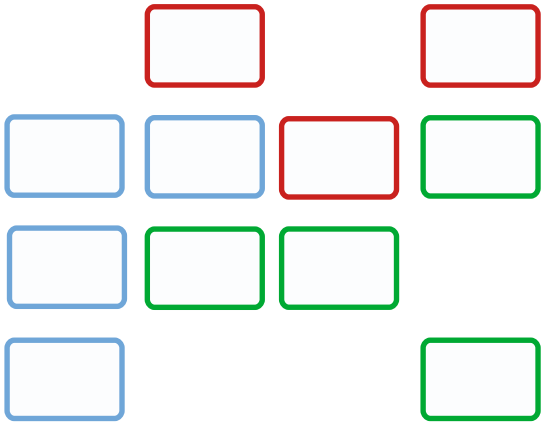




Applications can manage those complexities ...

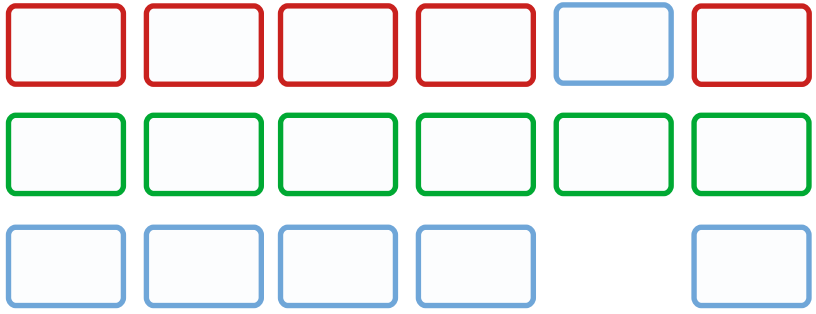


Platform A



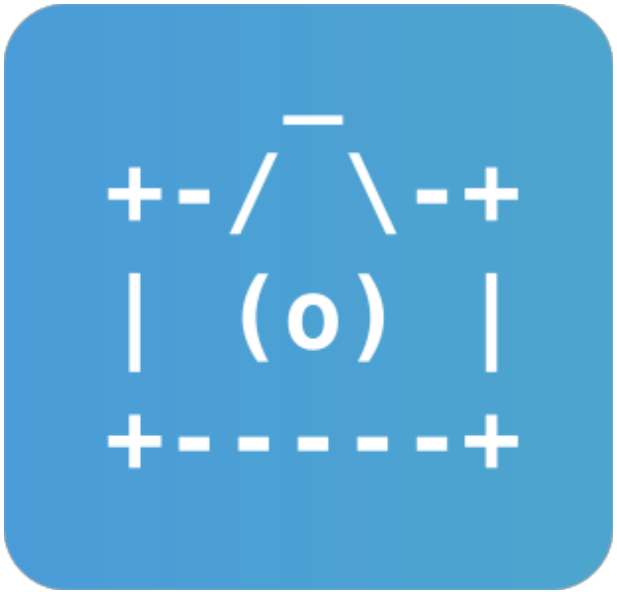
Platform C

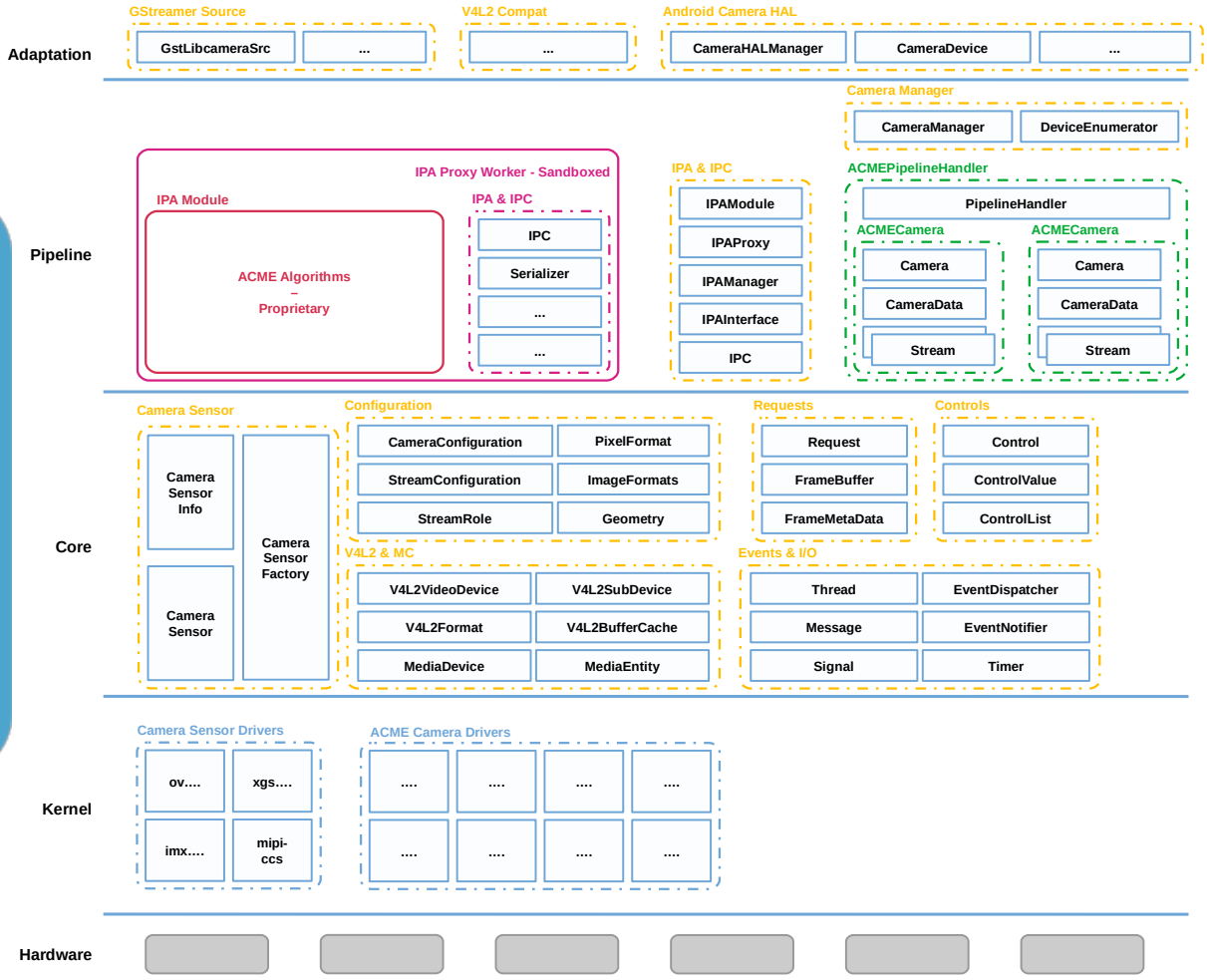
Platform B



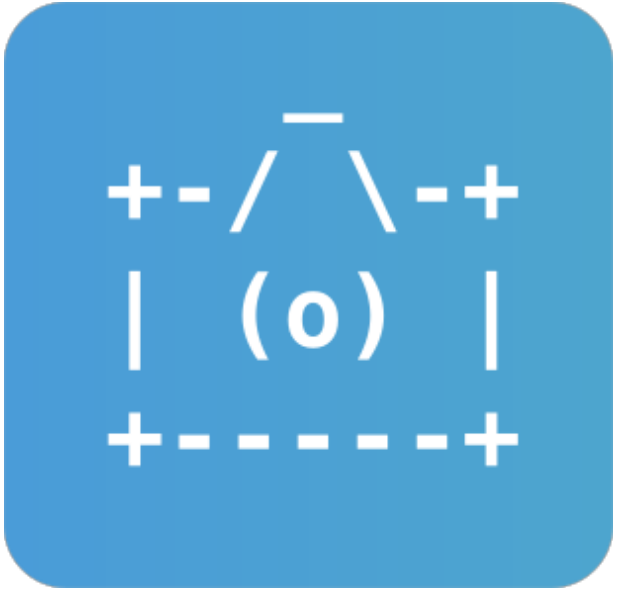
IDEAS ON BOARD

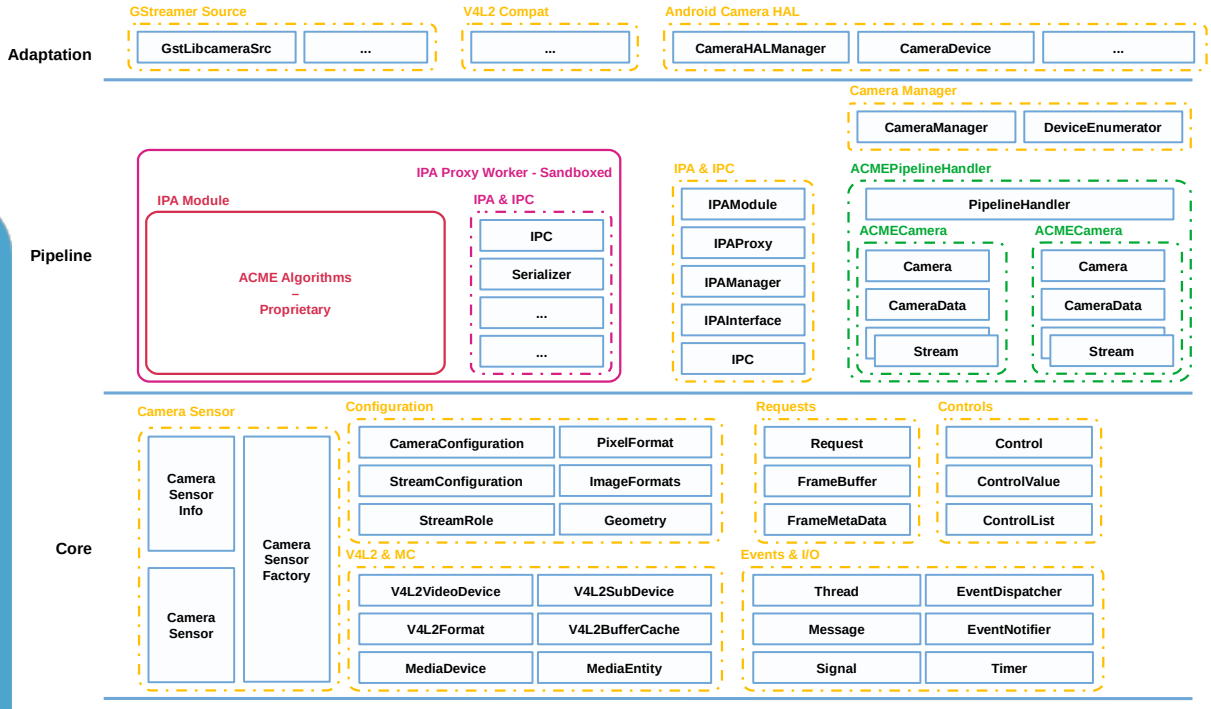
But it doesn't scale





libcamera fills that gap





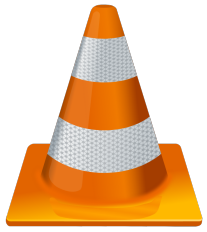
libcamera
An open source camera stack and framework for Linux, Android, and ChromeOS

libcamera fills that gap

- Complex Cameras
- Challenges
- **Complications**
- Features & Developments
- Q+A

- V4L2 started between 1998, 2002
- Consistent API supports many Video capture devices
  - Wide support of pixel formats …
  - Same API for (existing, simple) Cameras, Digital TV DVB, Set Top Box …
- It exists
  - There hasn’t been anything else at the kernel level (upstream)
- Widely used (thoroughly tested)

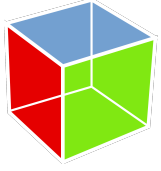
Everybody loves V4L2 *



Media/Camera Applications



Browsers




Multimedia and Application Frameworks


Conferencing Utilities

V4L2 is already used everywhere

- Really - they love ‘/dev/video0’

- Media Controller
  - Entities, links, pads, format negotiation (propagation)

- Subdevices
  - Direct control over specific internal components
  - Which one do you configure?

- Multiple video nodes for a single “Camera device”
  - UVC - ‘Metadata video node’
  - CSI-2 Receiver
  - ISP - Statistics, Parameter Buffers, Multiple image streams
  - M2M Dewarper

Not everybody loves V4L2 *

- 3A algorithms need to be handled in userspace
  - Crucial for RAW sensors. YUV sensors are becoming obsolete
- Laptops are now using complex cameras
  - Dell, Lenovo, HP, Surface …
- Embedded devices already use complex cameras
  - OEM/ODM … need custom solutions to manage each camera
- No portable mobile camera applications
  - Mobile is dominated by Android, with mostly binary camera stacks

V4L2 alone isn’t enough

Source: Flickr
Attribution-NoDerivs 2.0 Generic (CC BY-ND 2.0)

Many applications don’t yet support libcamera

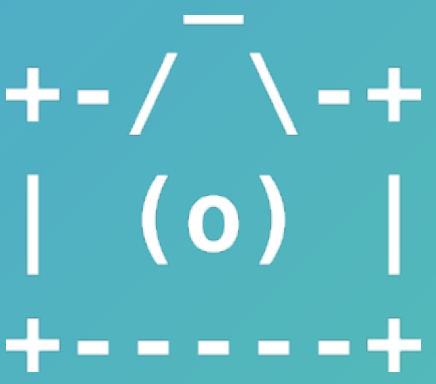
Adding the support takes effort
- maintainers haven’t expected this

C applications don’t want to use C++
- People can be scared of the ++

Is it even finished yet?
- Releases, ABI stability …

But … now there’s a new API to use

- Complex Cameras
- Challenges
- Complications
- **Features & Developments**
- Q+A



Gstlibcamerasrc brings the whole GStreamer ecosystem to libcamera devices:

- Encoding / Streaming
- Composing / Mixing
- Audio

Camera Viewer

```
gst-launch-1.0 libcamerasrc ! 'video/x-raw,width=1280,height=720' ! glimagesink
```

JPEG Network streamer

```
gst-launch-1.0 libcamerasrc ! \
    video/x-raw,colorimetry=bt709,format=NV12,width=1280,height=720,framerate=30/1 ! \
    jpegenc ! multipartmux ! \
    tcpserversink host=0.0.0.0 port=5000
```

JPEG Network Receiver

```
gst-launch-1.0 tcpclientsrc host=$DEVICE_IP port=5000 ! \
    multipartdemux ! jpegdec ! autovideosink
```

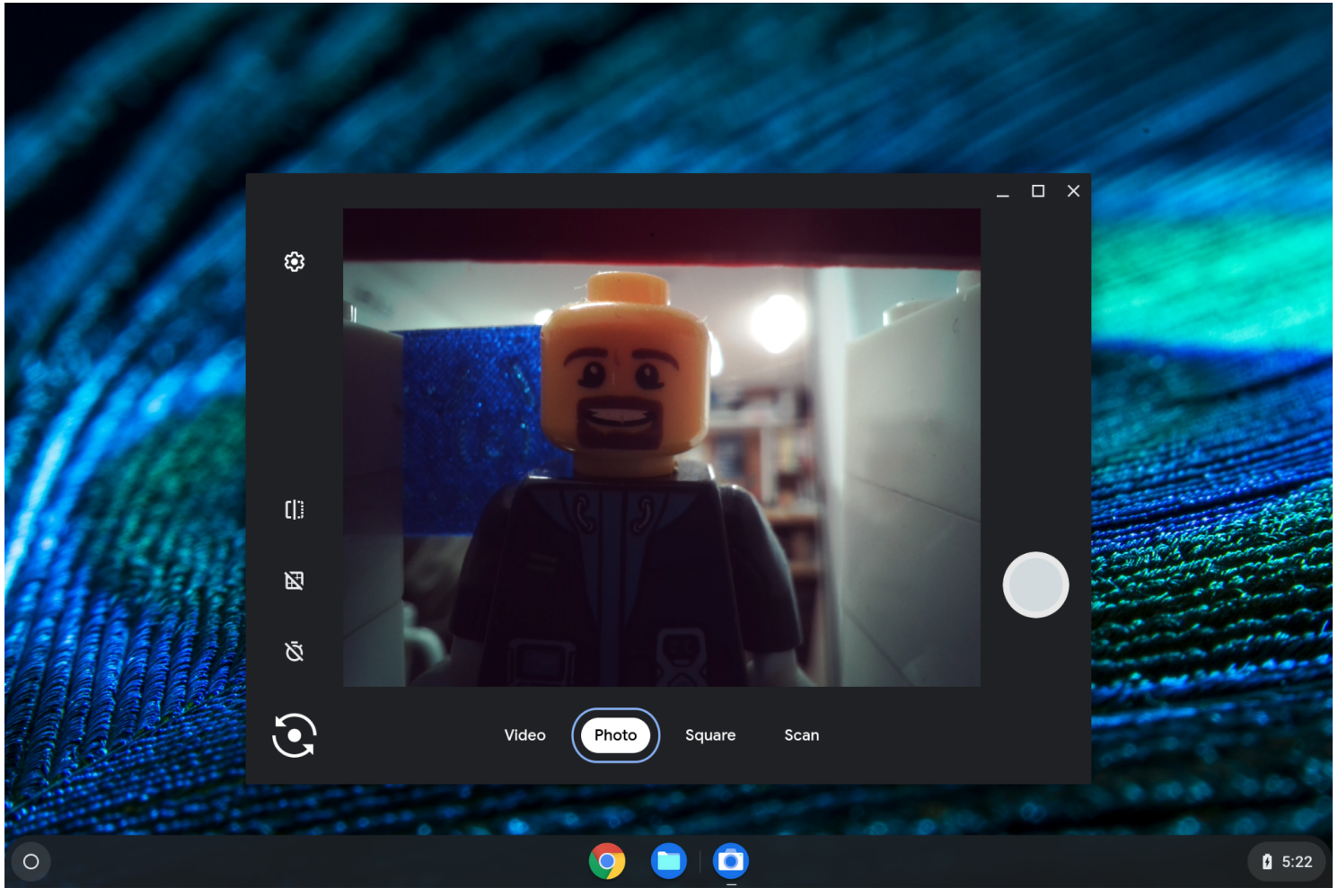
libcamera provides a GStreamer element

- Few users like C++
- Python is “friendly”
- Fast to prototype
- ‘Picamera2’* simplifies the libcamera Python API

```
#!/usr/bin/python3

# Capture a JPEG while still running in the preview mode. When you
# capture to a file, the return value is the metadata for that image.

import time

from picamera2 import Picamera2, Preview

picam2 = Picamera2()

preview_config = picam2.create_preview_configuration(main={"size": (800, 600)})
picam2.configure(preview_config)

picam2.start_preview(Preview.QTGL)

picam2.start()
time.sleep(2)

metadata = picam2.capture_file("test.jpg")
print(metadata)

picam2.close()
```

*** Picamera2 is the libcamera-based replacement for Picamera which was a Python interface to the Raspberry Pi's legacy camera stack.**

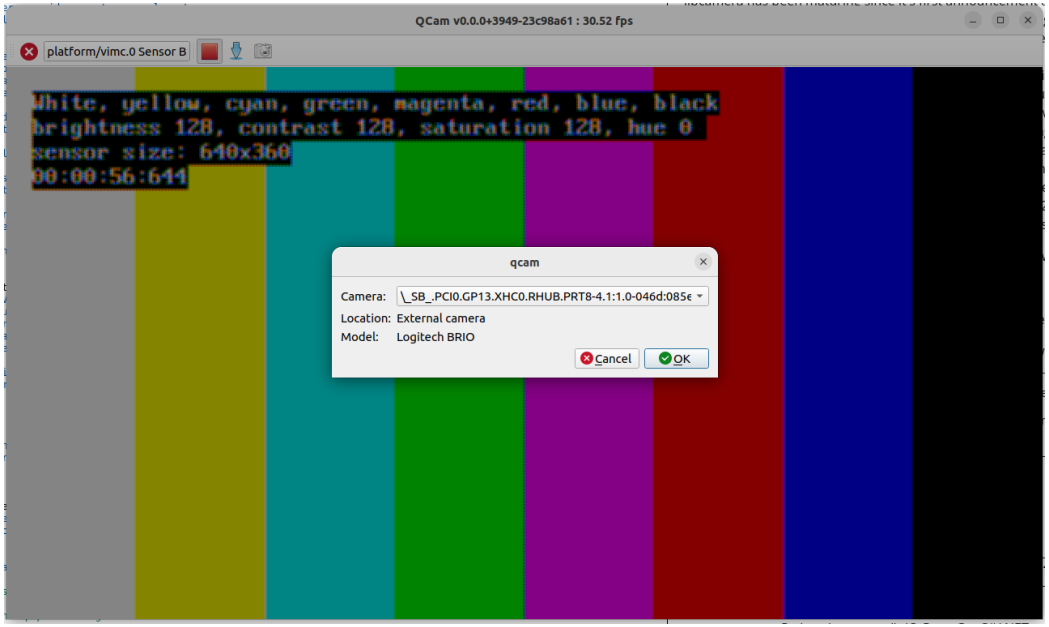
We have Python support

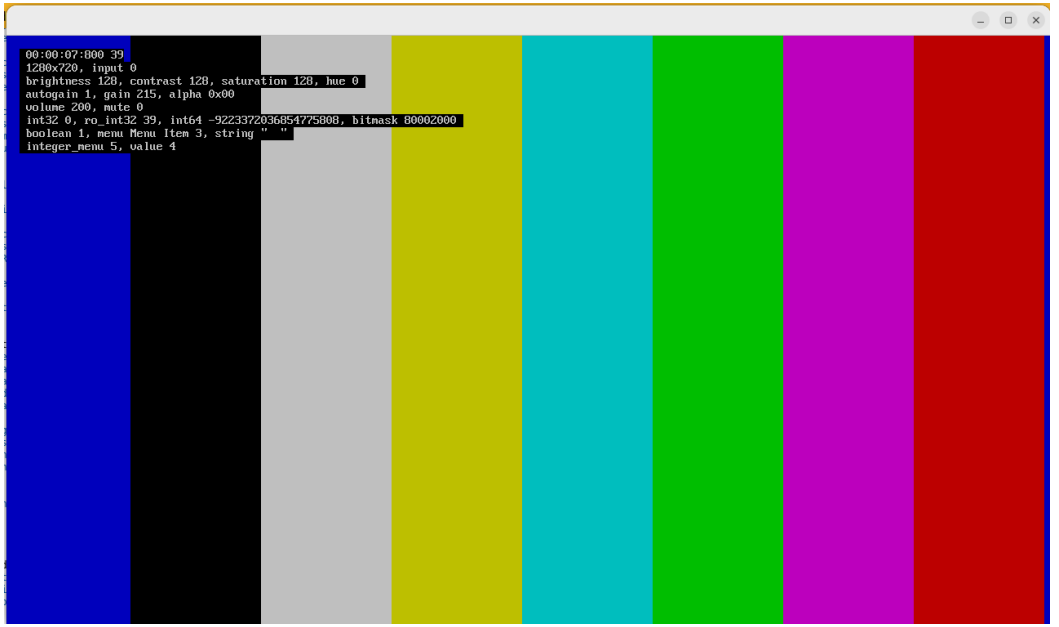


An Android HAL implementation

- LD_PRELOAD solution
  - LIBCAMERA_PUBLIC int **open**(const char *path, int oflag, ...)
  - LIBCAMERA_PUBLIC int open64(const char *path, int oflag, ...)
  - LIBCAMERA_PUBLIC int openat(int dirfd, const char *path, int oflag, ...)
  - LIBCAMERA_PUBLIC int __openat_2(int dirfd, const char *path, int oflag)
  - LIBCAMERA_PUBLIC int openat64(int dirfd, const char *path, int oflag, ...)
  - LIBCAMERA_PUBLIC int **dup**(int oldfd)
  - LIBCAMERA_PUBLIC int **close**(int fd)
  - LIBCAMERA_PUBLIC void ***mmap**(void *addr, size_t length, int prot, int flags, int fd, off_t offset)
  - LIBCAMERA_PUBLIC void *mmap64(void *addr, size_t length, int prot, int flags, int fd, off64_t offset)
  - LIBCAMERA_PUBLIC int **munmap**(void *addr, size_t length)
  - LIBCAMERA_PUBLIC int **ioctl**(int fd, unsigned long request, ...)
- **'libcamerify'**
  - $ libcamerify -d -d myV4L2Application --myArgs

... and a V4L2 compatibility layer





$ qcam -r gles

(Using VIMC)

$ cam -c3 -C -S –stream pixelformat=YUYV

(Using VIVID)

As well as test applications

```
159  int main()
160  {
161          /*
162           * -----------------------------------------------------------------------
163           * Create a Camera Manager.
164           *
165           * The Camera Manager is responsible for enumerating all the Camera
166           * in the system, by associating Pipeline Handlers with media entities
167           * registered in the system.
168           *
169           * The CameraManager provides a list of available Cameras that
170           * applications can operate on.
171           *
172           * When the CameraManager is no longer to be used, it should be deleted.
173           * We use a unique_ptr here to manage the lifetime automatically during
174           * the scope of this function.
175           *
176           * There can only be a single CameraManager constructed within any
177           * process space.
178           */
179          std::unique_ptr<CameraManager> cm = std::make_unique<CameraManager>();
180          cm->start();
181
182          /*
183           * Just as a test, generate names of the Cameras registered in the
184           * system, and list them.
185           */
186          for (auto const &camera : cm->cameras())
187                  std::cout << " - " << cameraName(camera.get()) << std::endl;
```

```
189          /*
190           * -----------------------------------------------------------------------
191           * Camera
192           *
193           * Camera are entities created by pipeline handlers, inspecting the
194           * entities registered in the system and reported to applications
195           * by the CameraManager.
196           *
197           * In general terms, a Camera corresponds to a single image source
198           * available in the system, such as an image sensor.
199           *
200           * Application lock usage of Camera by 'acquiring' them.
201           * Once done with it, application shall similarly 'release' the Camera.
202           *
203           * As an example, use the first available camera in the system after
204           * making sure that at least one camera is available.
205           *
206           * Cameras can be obtained by their ID or their index, to demonstrate
207           * this, the following code gets the ID of the first camera; then gets
208           * the camera associated with that ID (which is of course the same as
209           * cm->cameras()[0]).
210           */
211          if (cm->cameras().empty()) {
212                  std::cout << "No cameras were identified on the system."
213                            << std::endl;
214                  cm->stop();
215                  return EXIT_FAILURE;
216          }
217
218          std::string cameraId = cm->cameras()[0]->id();
219          camera = cm->get(cameraId);
220          camera->acquire();
```

https://git.libcamera.org/libcamera/simple-cam.git/

And a sample ‘hello world’ for the API

Pipewire integration brings desktop use cases

XDG-Camera-Portal

Raspberry Pi

WebRTC

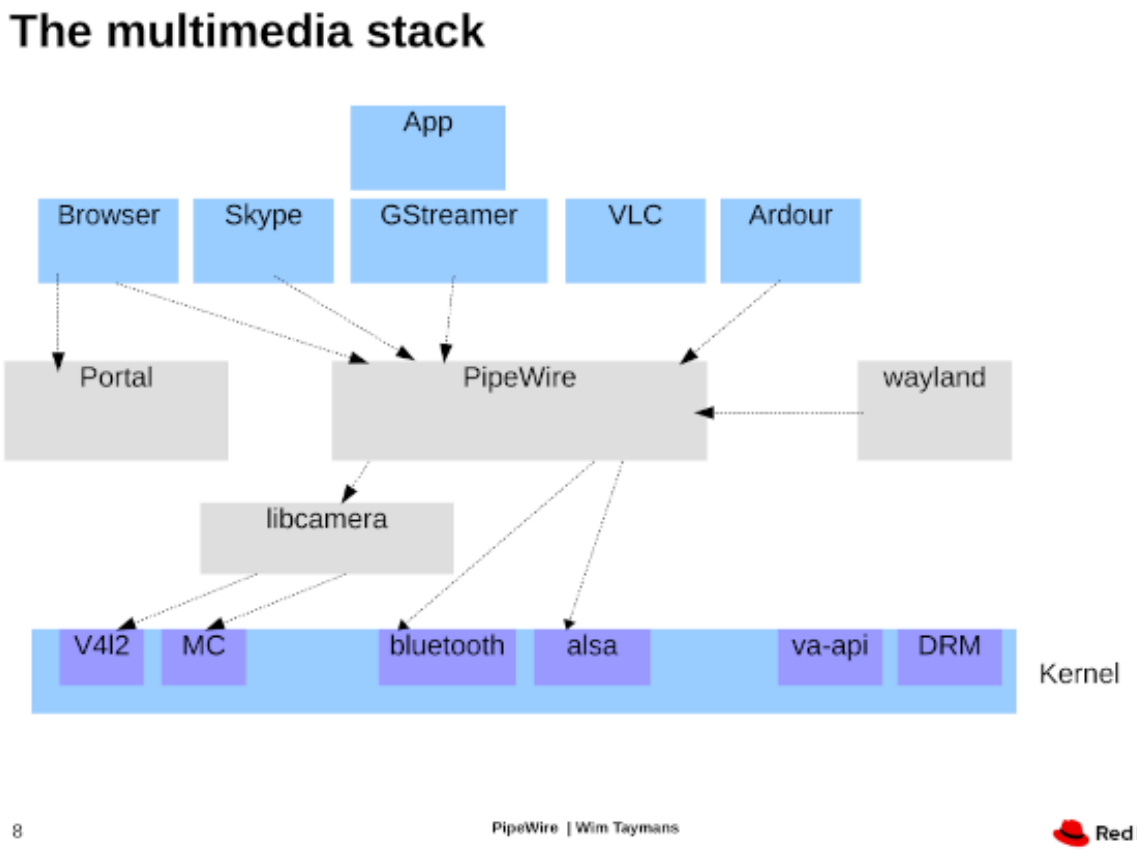
Jitsi Meet

zoom

https://flatpak.github.io/xdg-desktop-portal/#gdbus-org.freedesktop.portal.Camera

IDEAS ON BOARD

Such as video conferencing through Chromium



https://blogs.gnome.org/uraeus/2021/10/01/pipewire-and-fixing-the-linux-video-capture-stack/

The Linux media stack

Snapshot

Take pictures and videos

Screenshots

https://gitlab.gnome.org/Incubator/snapshot

IDEAS ON BOARD

snapshot: a convergent camera app

- Integration with xdg-portal enables flatpak support
- Application sandboxing with security checks in place

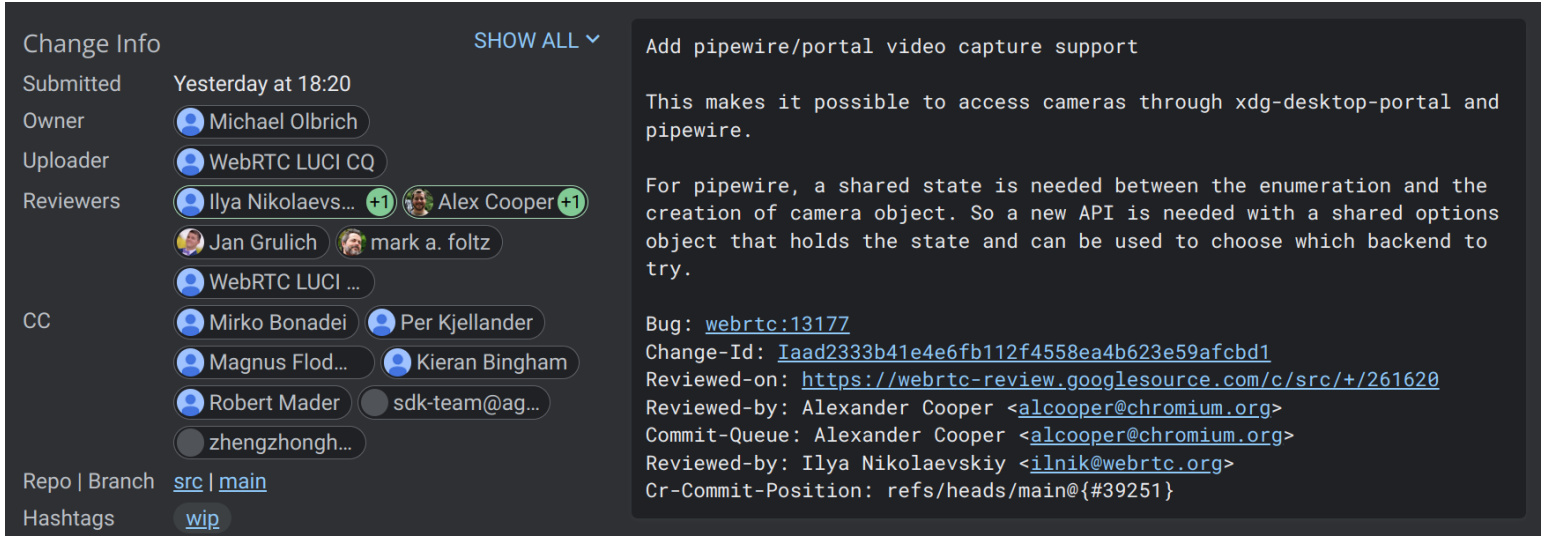
```
# Add the gnome-nightly repo
flatpak remote-add --user --if-not-exists gnome-nightly https://nightly.gnome.org/gnome-nightly.flatpakrepo

# Install the demo build
flatpak install --user camera-devel-aarch64.flatpak
# or
flatpak install --user camera-devel-x86.flatpak
```

Flatpak support

Gerrit CHANGES YOUR DOCUMENTATION BROWSE

Merged ☆ 261620 Add pipewire/portal video capture support

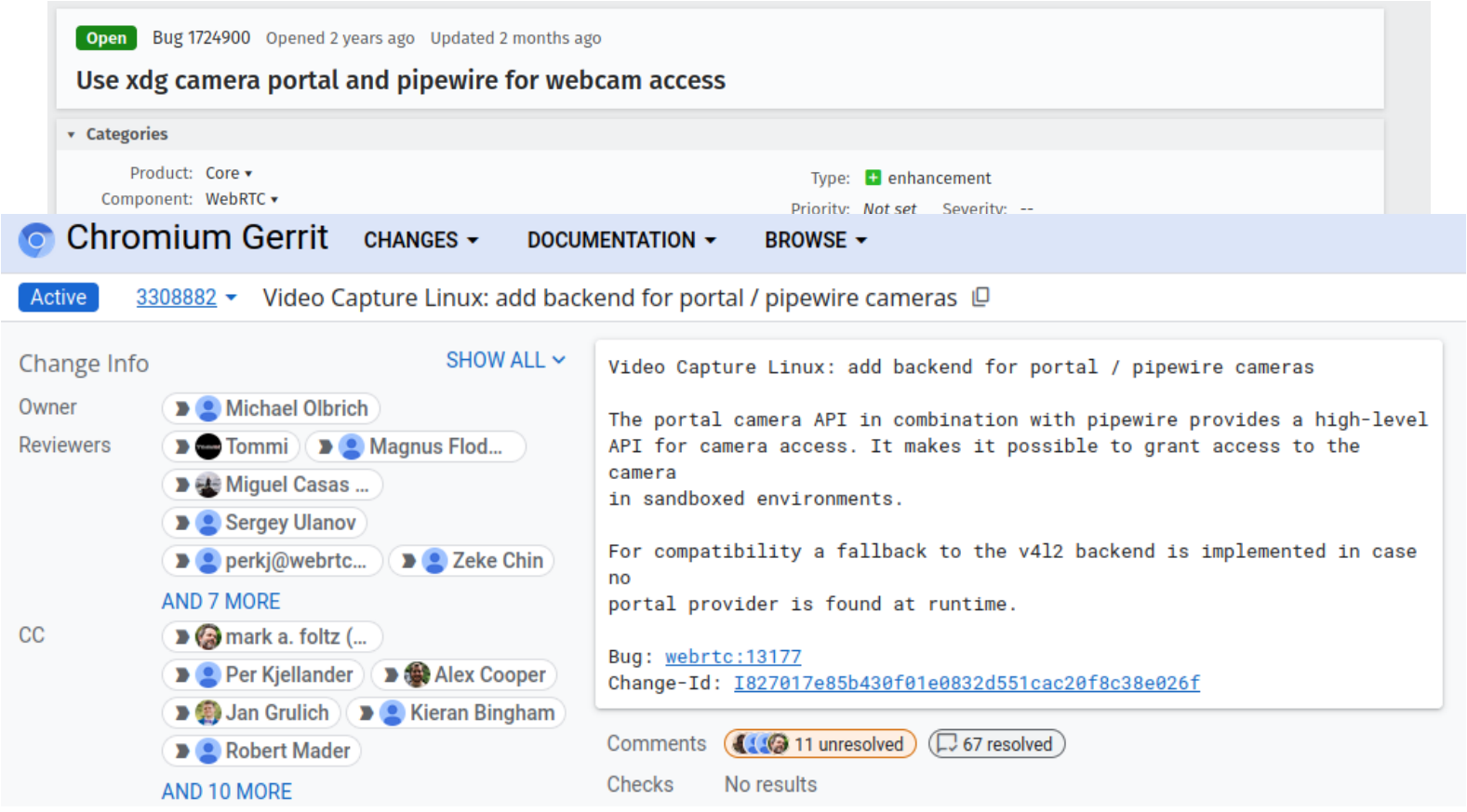
Support for *video capture through xdg-desktop-portal* *merged in webRTC*



IDEAS ON BOARD

WebRTC now supports xdg-portals!

In a few months (?) you'll be able to access cameras from your browsers !!



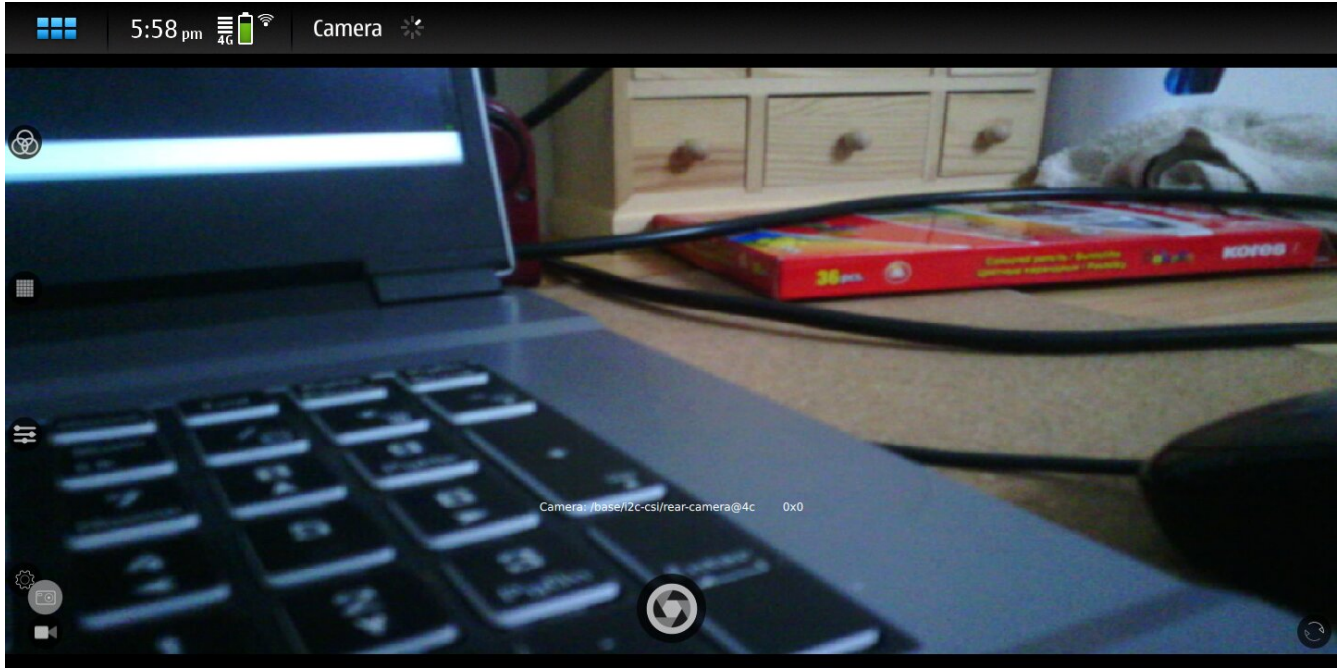
IDEAS ON BOARD

(Near future) Access cameras from browsers

Adam Piggs' pinhole

- Fork of Sailfish OS *harbour-camera*
- Qt/QML libcamera-based mobile camera application
- Developed for the original PinePhone
  - *But since it uses libcamera it runs un-modified on the Pro*
- Manual controls (exposure time, brightness etc) in the UI

Adam Piggs' pinhole



Maemo Leste developer rafael2k testing pinhole on the Pinephone

Other notable developments: pinhole

Waydroid

Waydroid uses a container-based approach to boot a full Android system on a regular GNU/Linux system

- Test ground for mobile linux capture
- Rockchip RK3399
- *complex camera* with ISP and RAW sensors
- Supported by libcamera

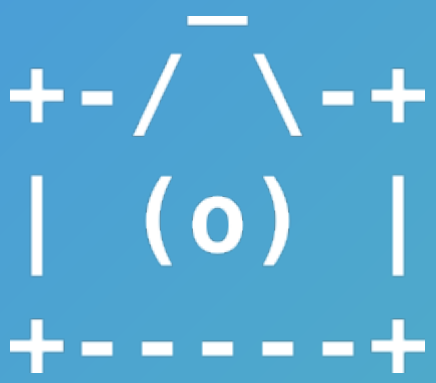
IDEAS ON BOARD

And finally The PinePhone Pro...

Demo video here

Thank you!

libcamera



libcamera

Questions?

Getting involved
https://libcamera.org/contributing.html