FOSS on Mobile Devices - Camera BoF

Jacopo Mondi

FOSDEM 2022

Hello, I’m Jacopo

- Linux camera engineer

- *jacopo@jmondi.org*
- *irc: jmondi oftc/libera*
- *matrix: @jmondi:nibble.pw*

Hello, I’m Jacopo

- Linux camera engineer

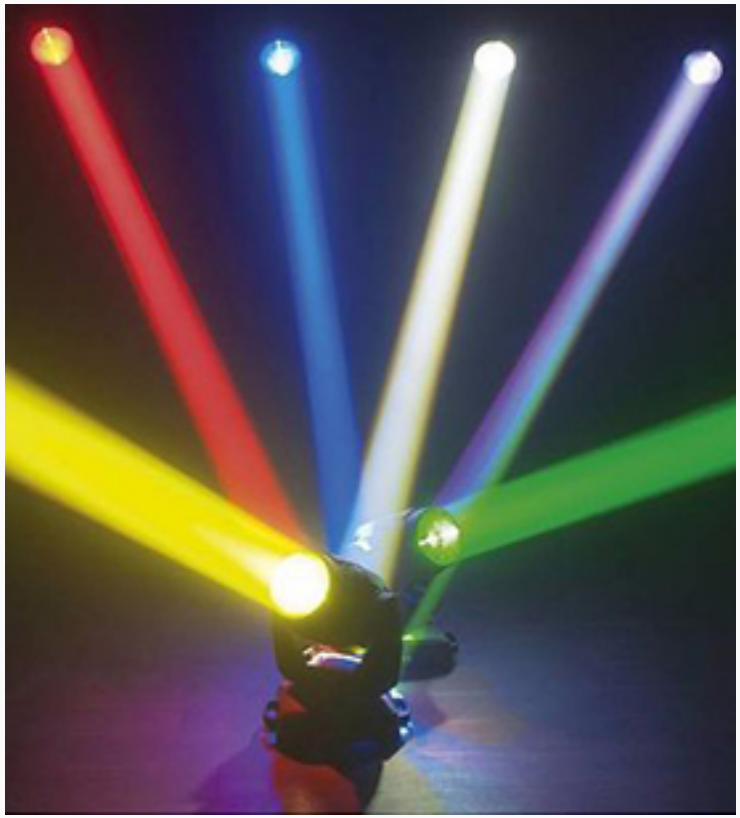
- libcamera
- V4L2 camera driver

Agenda

- The image capture pipeline
- The 3A loop
- A taxonomy of camera systems
- The complex camera interface
- The case for a camera stack

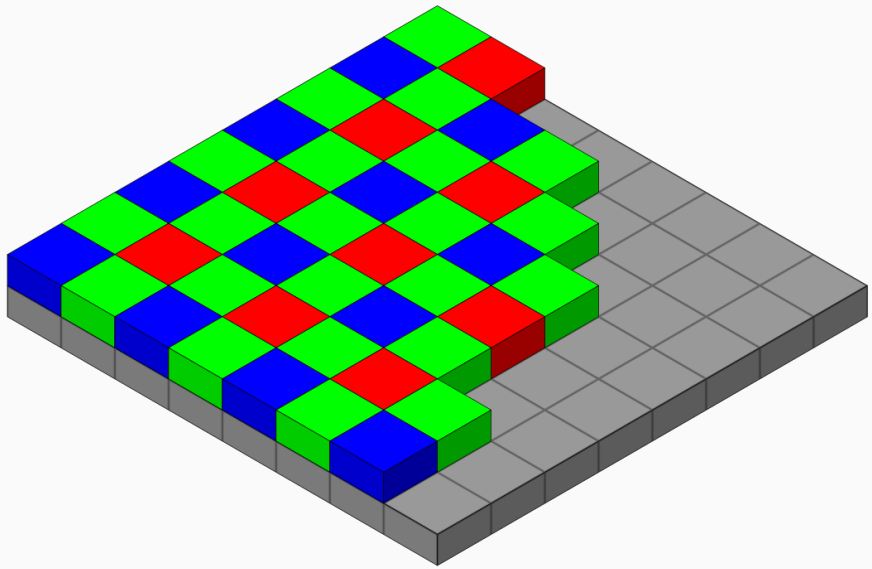
The image capture pipeline

It all starts with light beams being converted into electric signals



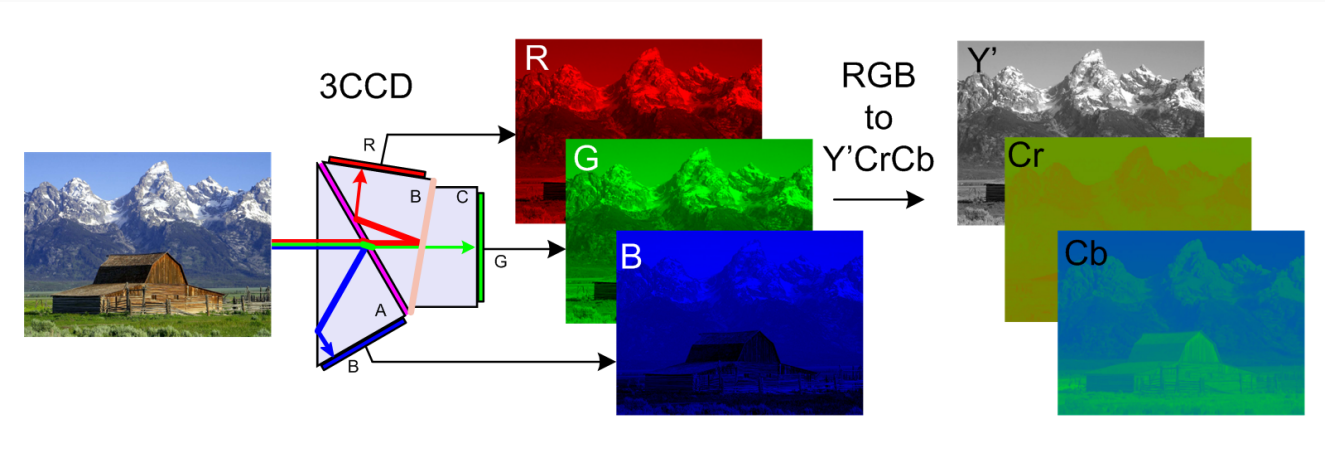
The image capture pipeline

Electrical information captured by the image sensor represents color channels in the RAW Bayer pattern of the sensor



The image capture pipeline

The Bayer pattern gets 'debayered' and converted into a known color space



The image capture pipeline

Electrical information are manipulated in both analog and digital domains, with a large number of correction and enhancement techniques

- Defective pixel correction
- Lens shading
- Black level correction
- ... you name it

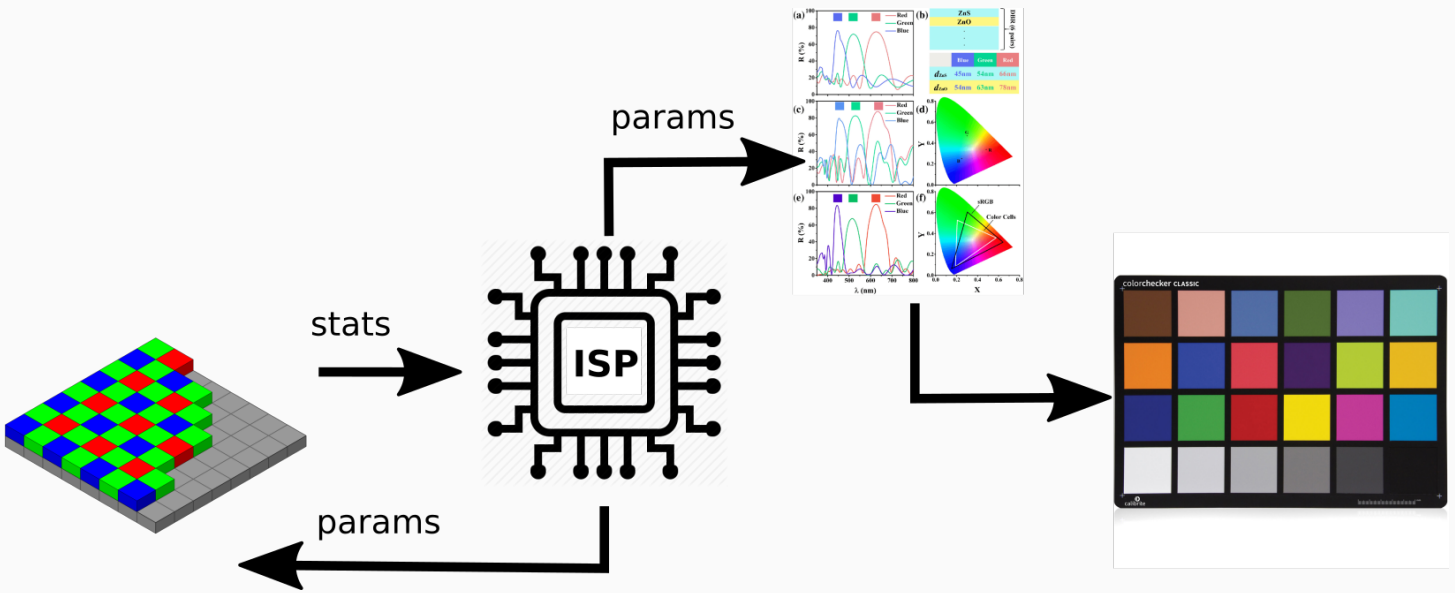
The image capture pipeline

A long way to reach the final image

- Cropping or sub-sampling applied on the image sensor pixel array
- Cropping or re-scaling in the digital domain
- Rotation/flipping/mirroring etc
- Color format conversion (eg RAW-to-YVV)
- Color space conversions (RGB-to-YUV)
- Packaging and compression (JPEG, DNF etc)

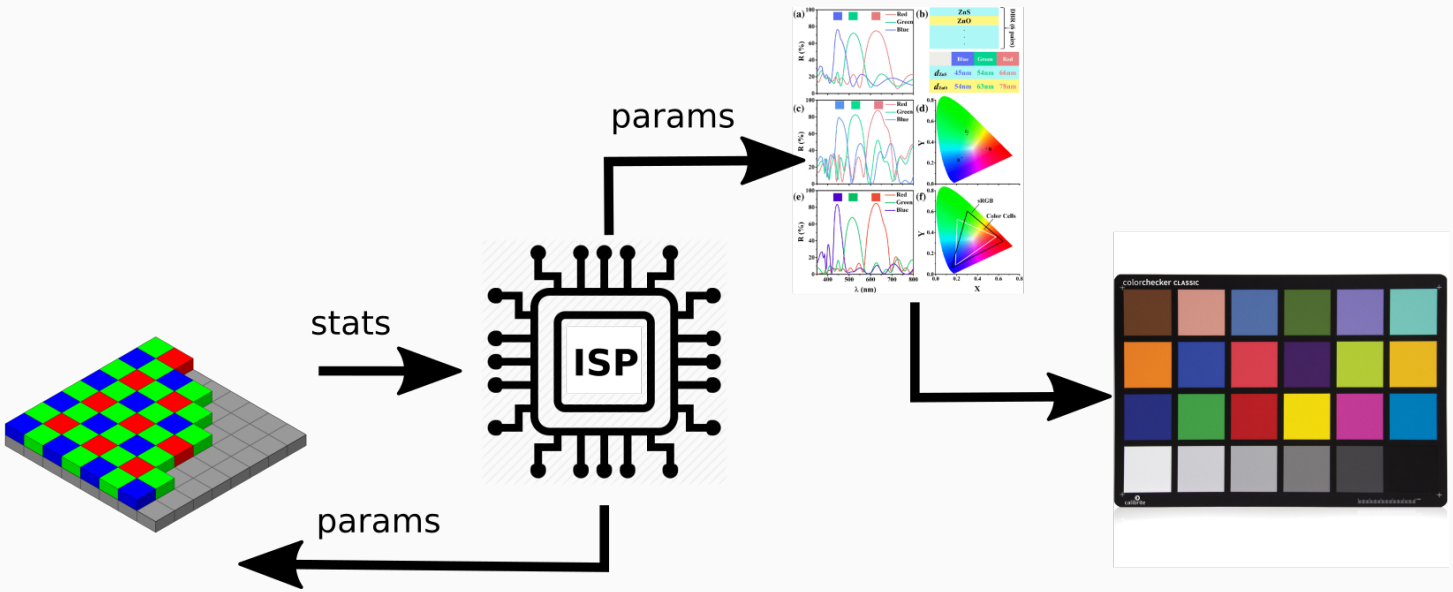
The 3A loop

Image capture is a closed-loop system



The 3A loop

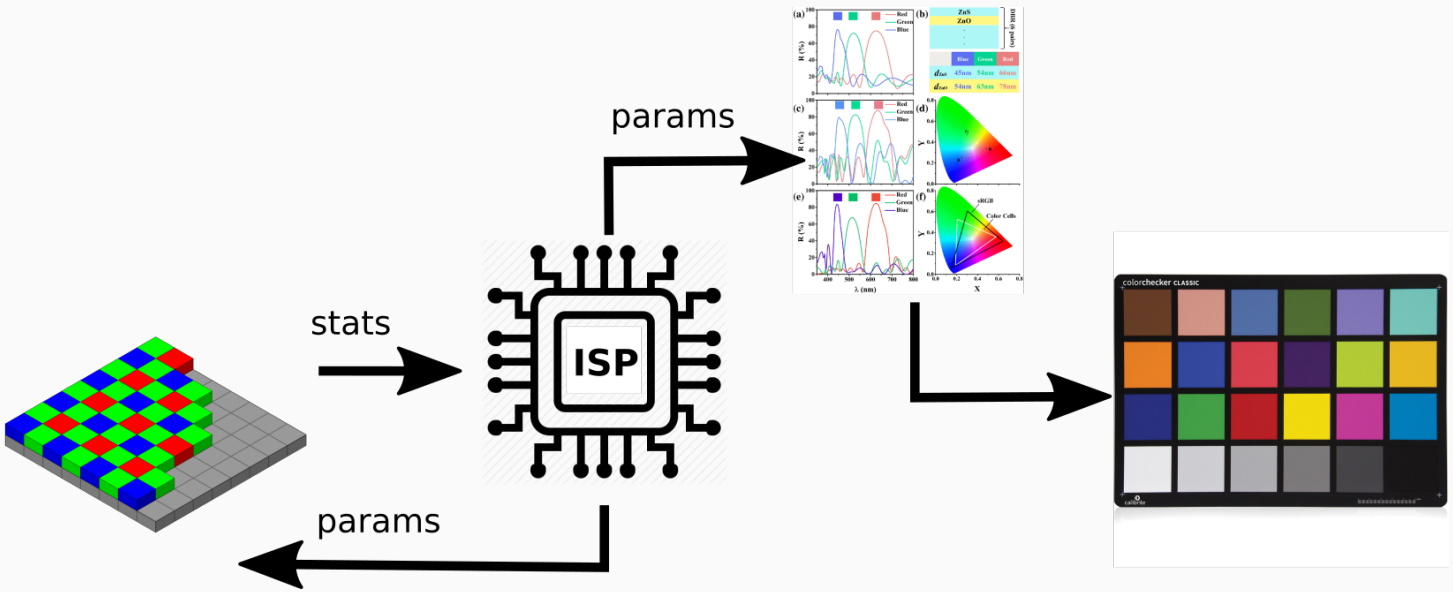
Image capture is a closed-loop system



ISP *Image Signal Processor*

The 3A loop

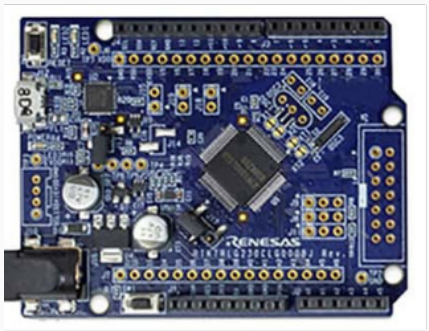
Image capture is a closed-loop system

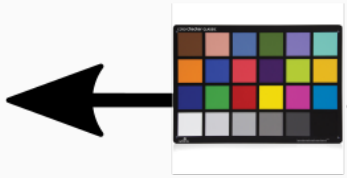


3A loop

A taxonomy of camera systems

Designs with a smart sensor





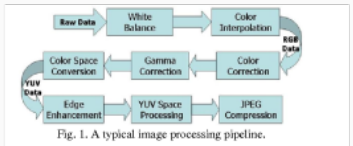


Fig. 1. A typical image processing pipeline.

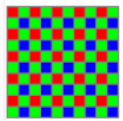


Fig. 2. Raw data format.


Fig. 3. GretagMacbeth ColorChecker.

A taxonomy of camera systems

- User visible functionalities depends on the selected sensor

A taxonomy of camera systems

- User visible functionalities depends on the selected sensor
- Sensor are (still) programmed through binary blobs

A taxonomy of camera systems

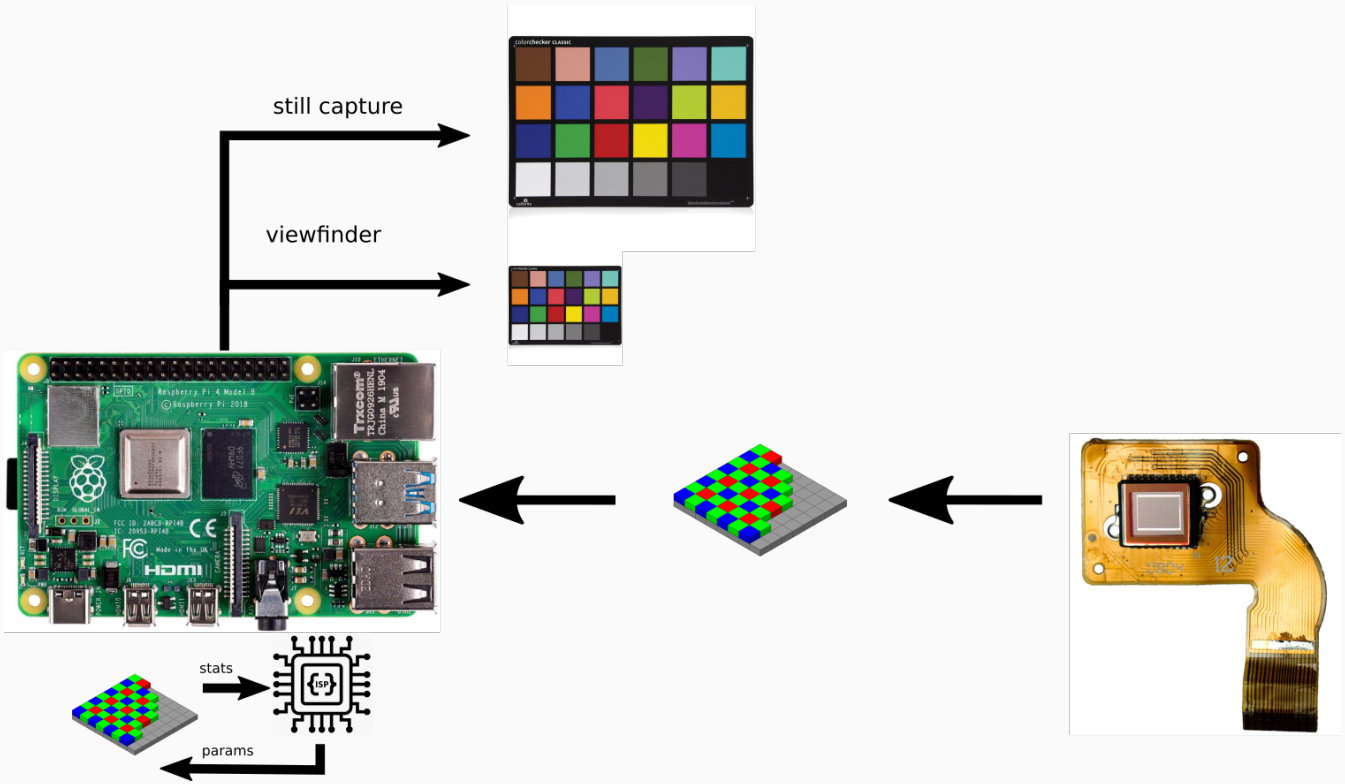
- User visible functionalities depends on the selected sensor
- Sensor are (still) programmed through binary blobs
- Vendor lock-in

A taxonomy of camera systems

- User visible functionalities depends on the selected sensor
- Sensor are (still) programmed through binary blobs
- Vendor lock-in
- Higher BOM costs

A taxonomy of camera systems

Designs with an ISP



A taxonomy of camera systems

- The SoC usually runs Linux

A taxonomy of camera systems

- The SoC usually runs Linux
  - *The code that controls the ISP is potentially accessible*

A taxonomy of camera systems

- Features and performances

A taxonomy of camera systems

- Features and performances
  - *High speed interconnects, higher clock rates*
  - *Extended features set*

A taxonomy of camera systems

- Higher chances of code re-use and standardization

A taxonomy of camera systems

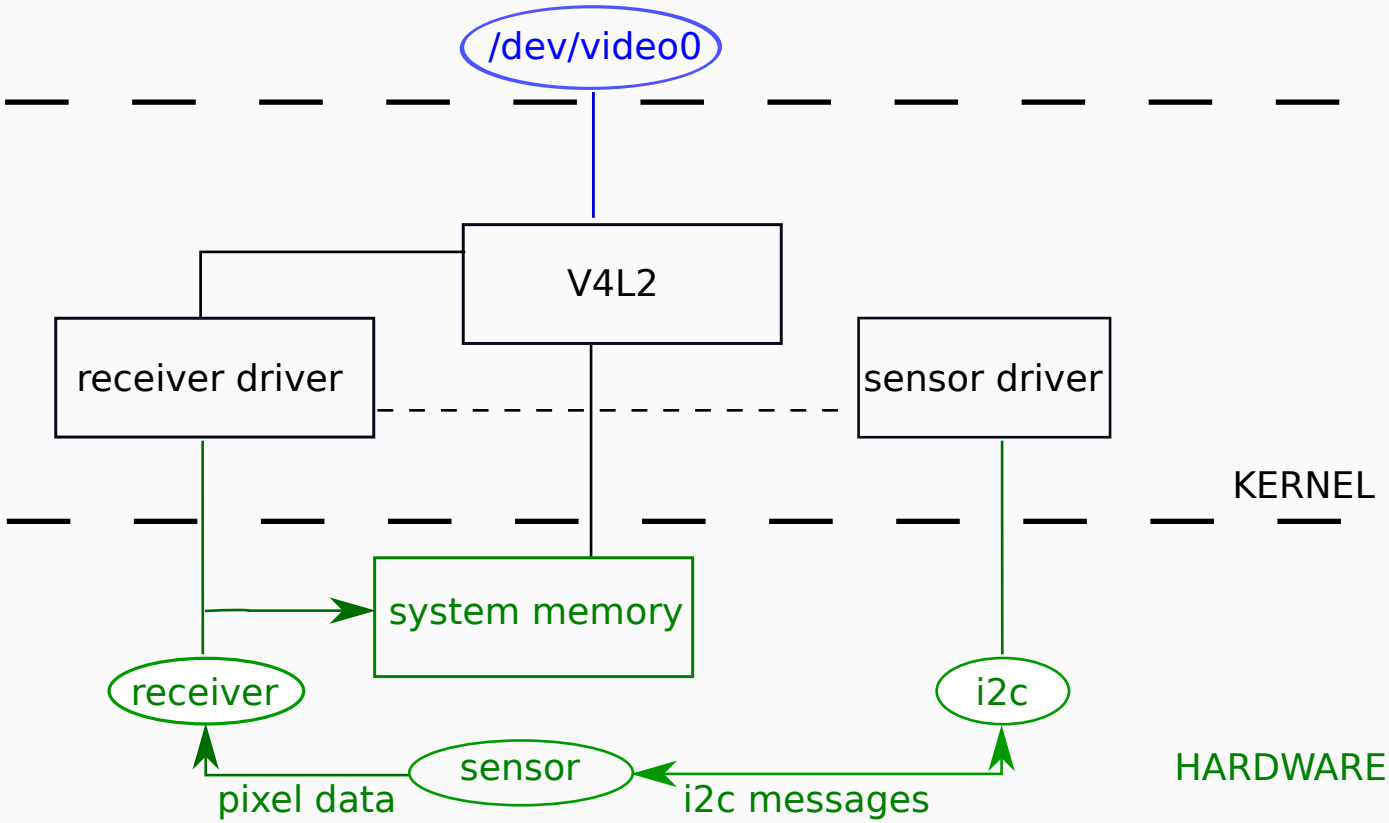
- Higher chances of code re-use and standardization
  - *Reusable 3A algorithms*

A taxonomy of camera systems

- Higher chances of code re-use and standardization
  - *Reusable 3A algorithms*
  - *Open HW designs*

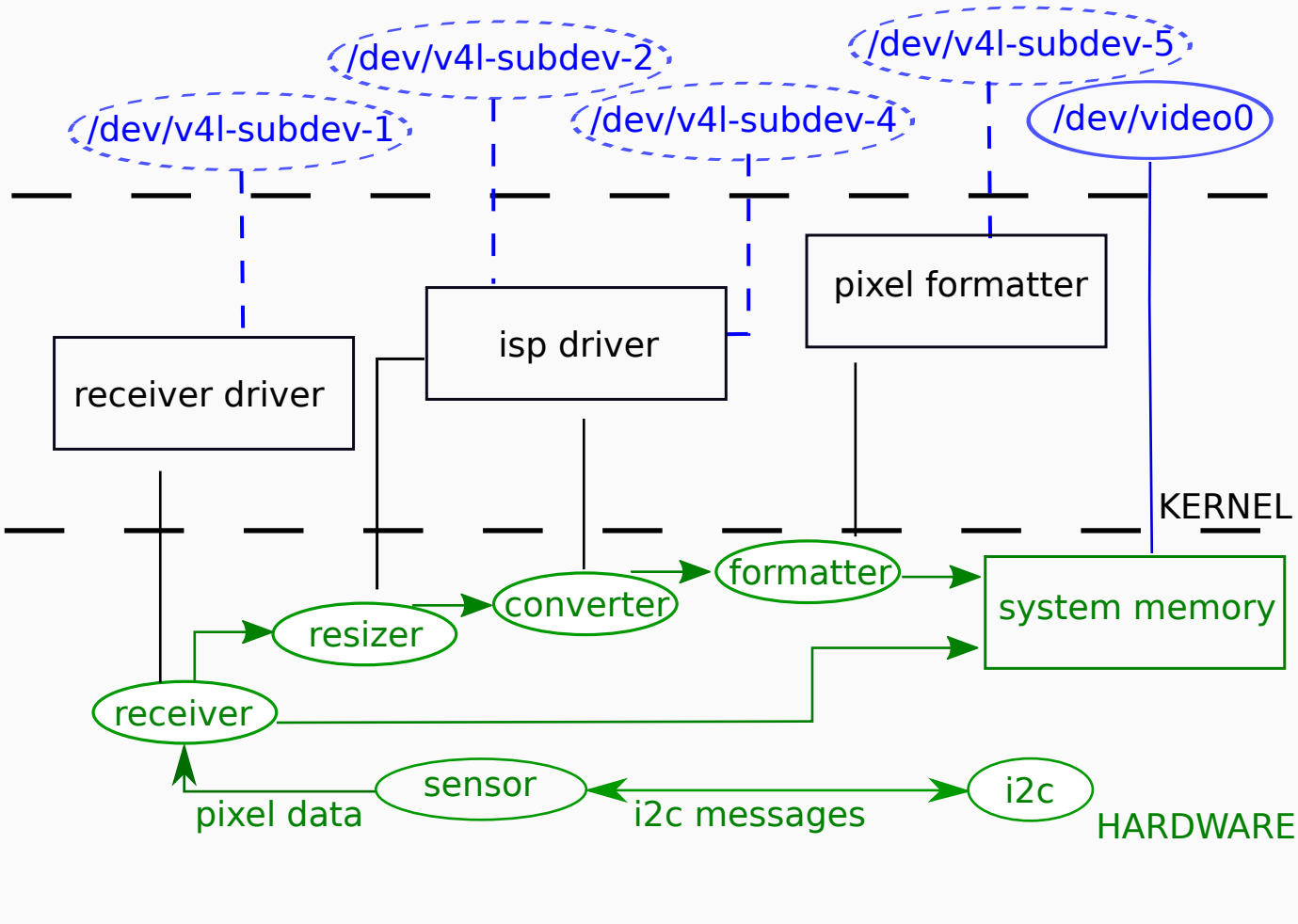
The complex kernel interface

The single devnode abstraction



The complex kernel interface

The media controller abstraction

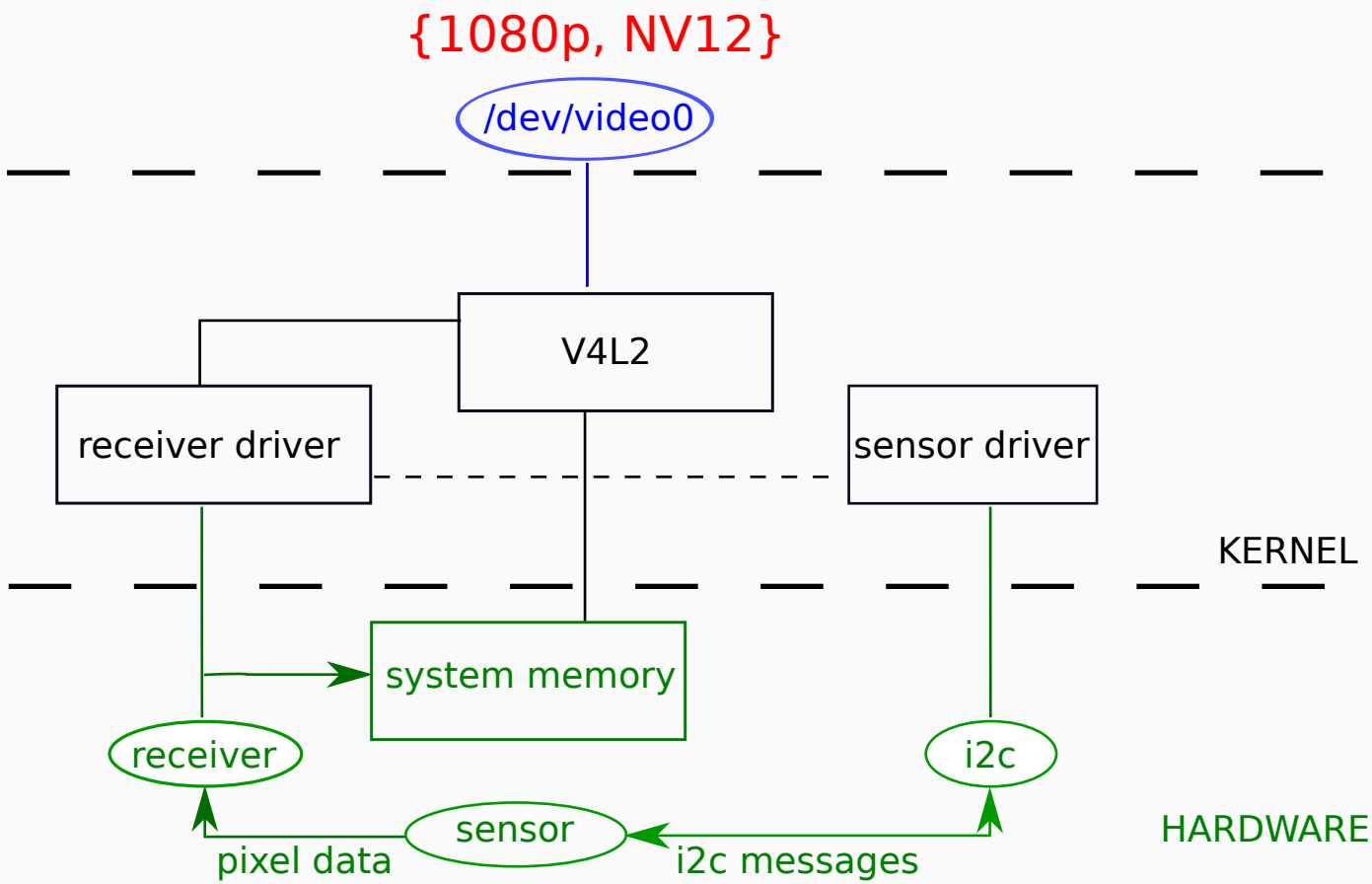


The complex kernel interface

Why so complex ??

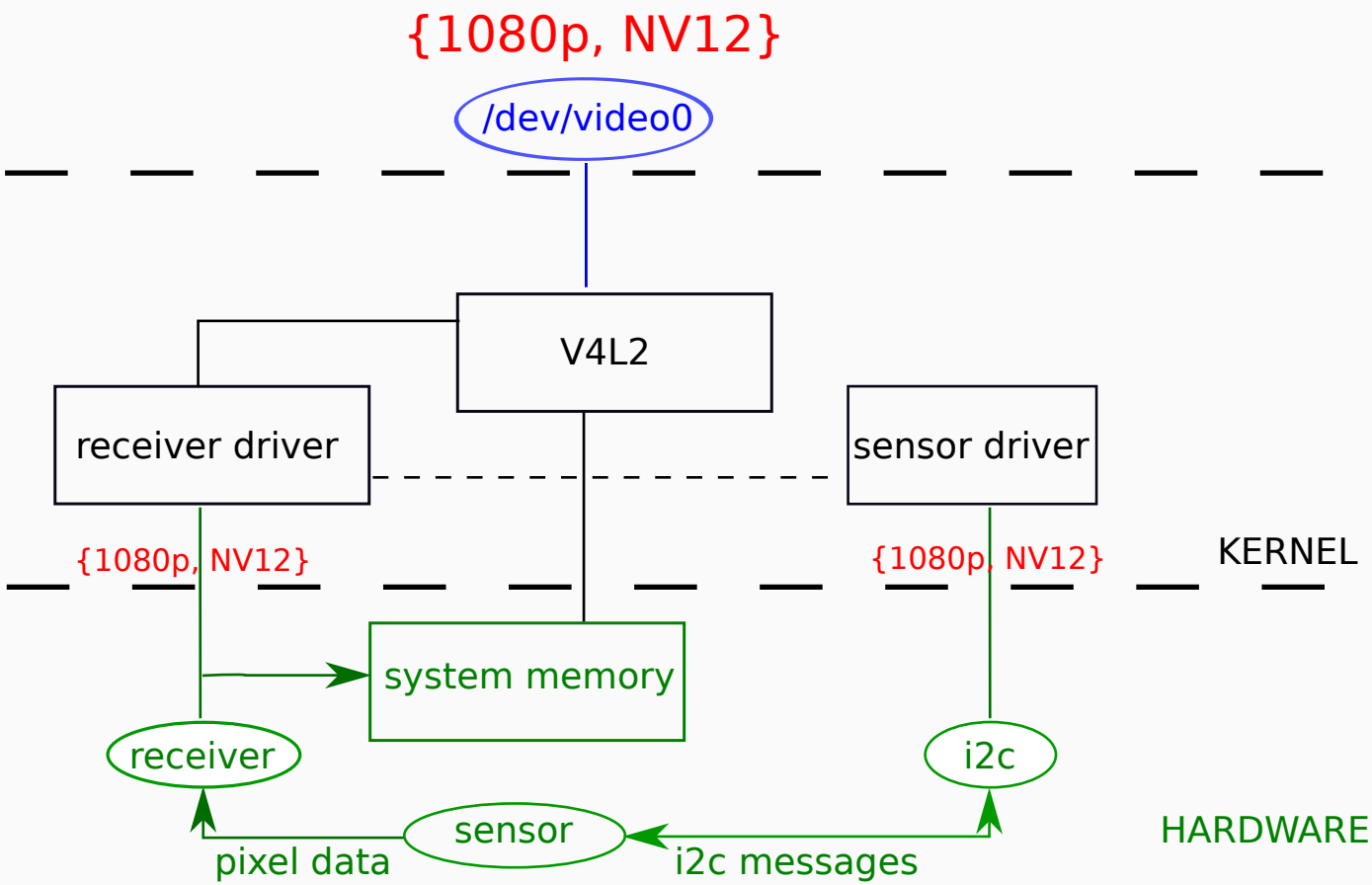
The complex kernel interface

The single devnode abstraction



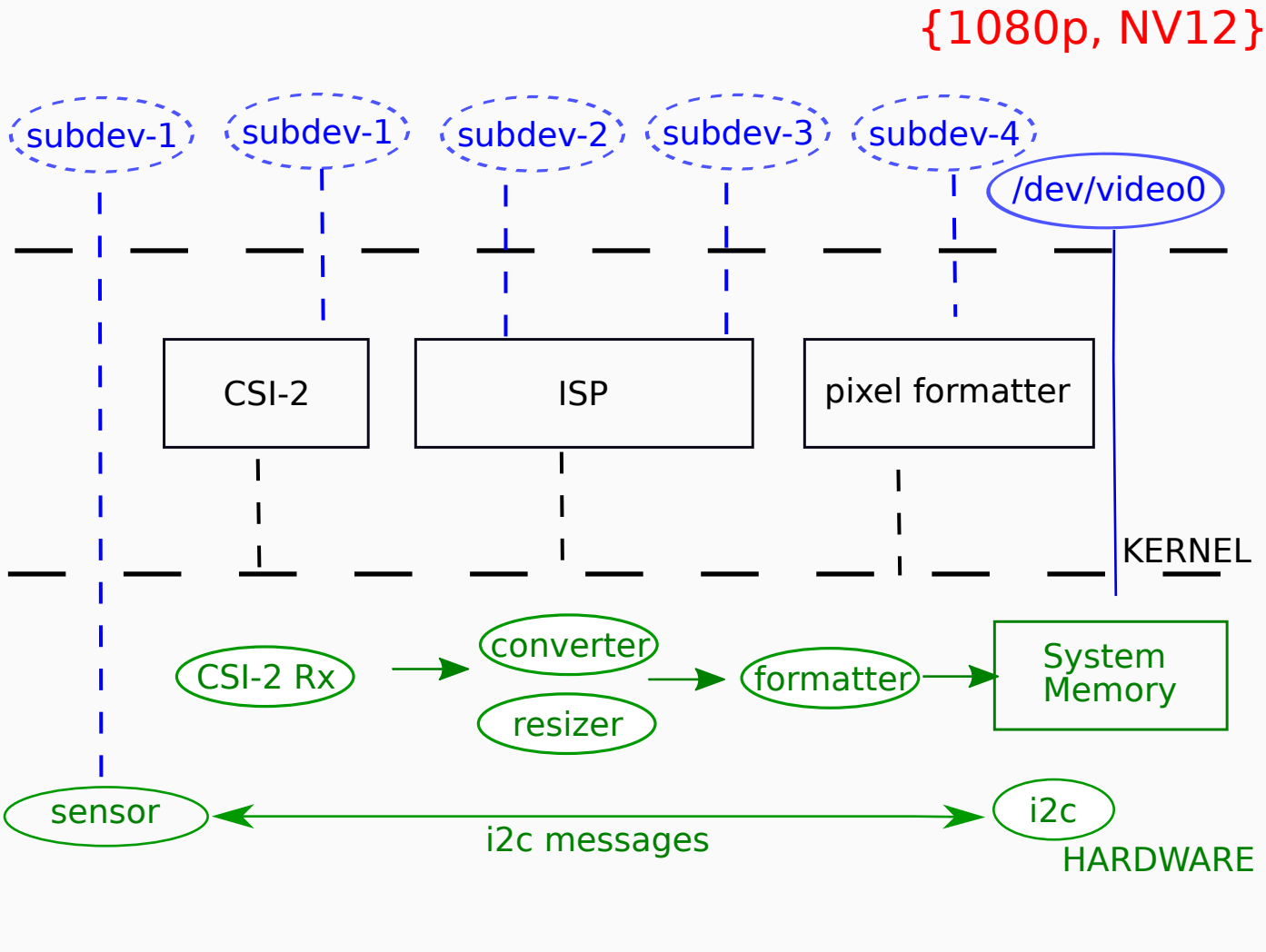
The complex kernel interface

The single devnode abstraction



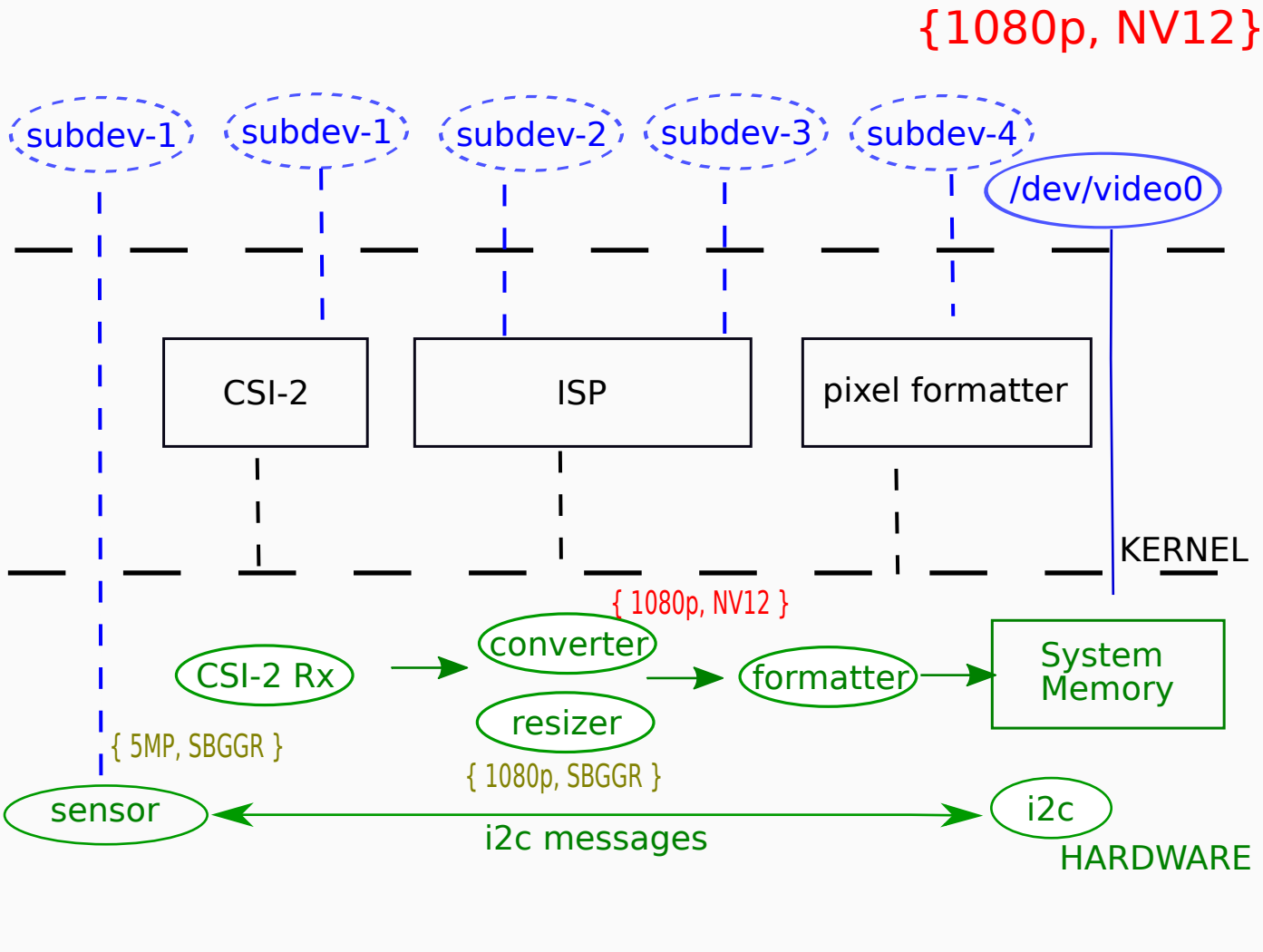
The complex kernel interface

The media controller abstraction



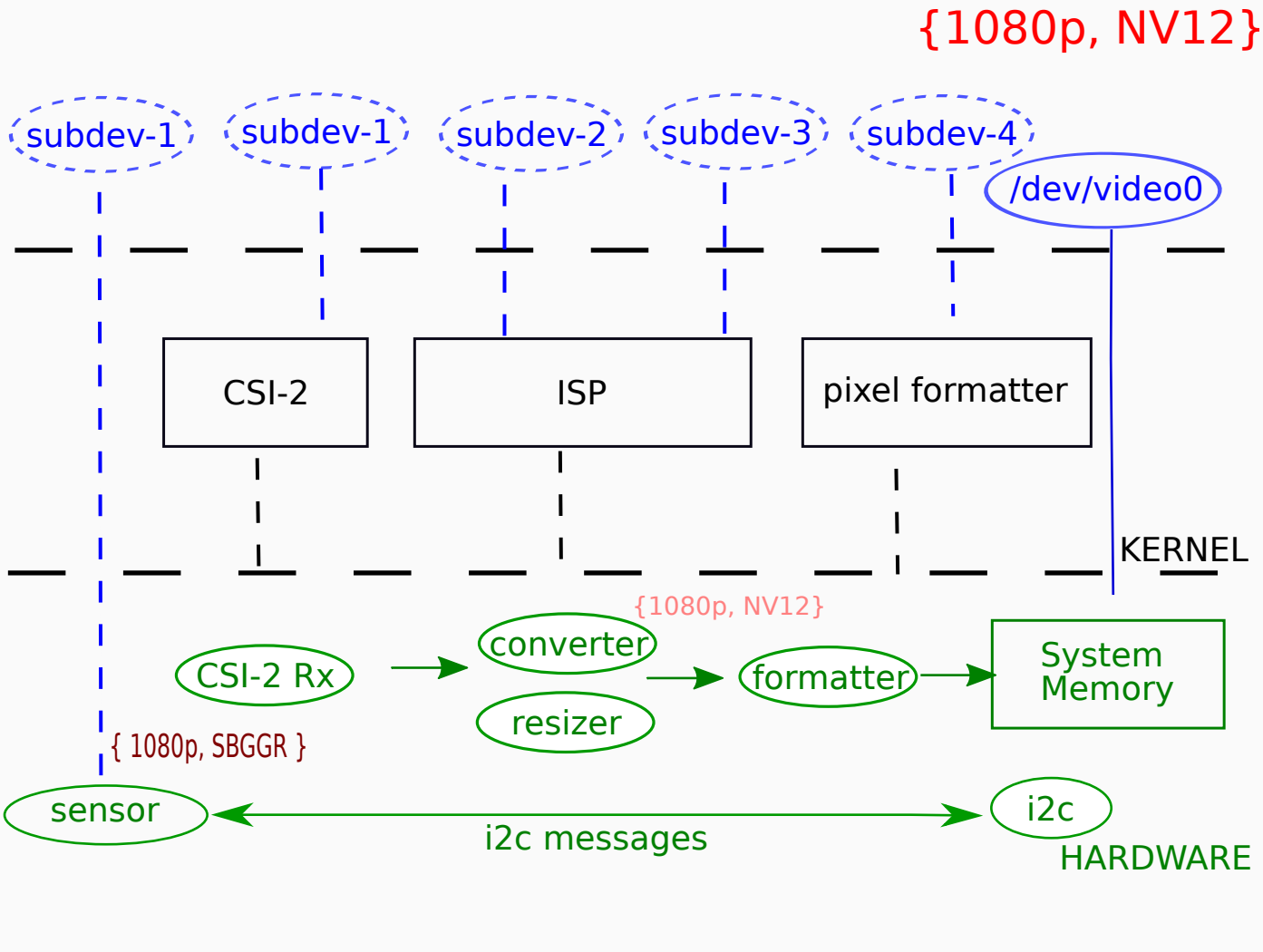
The complex kernel interface

The media controller abstraction



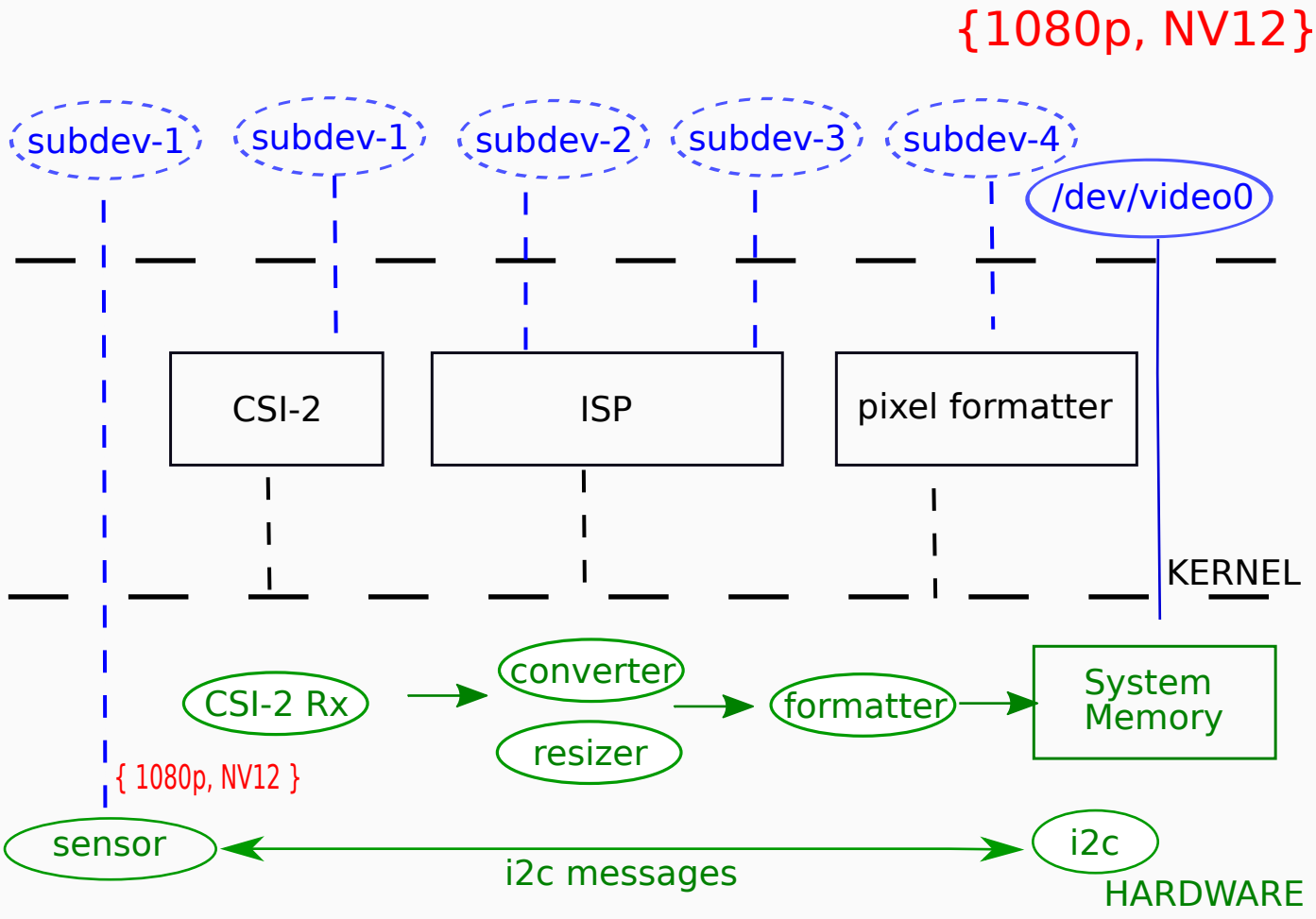
The complex kernel interface

The media controller abstraction



The complex kernel interface

The media controller abstraction



The case for a camera stack

Complex cameras

- User-space has to configure precisely each point in the pipeline

The case for a camera stack

Complex cameras

- User-space has to configure precisely each point in the pipeline
- An operating system should enable use cases, not hard-code them

The case for a camera stack

Complex cameras

- User-space has to configure precisely each point in the pipeline
- An operating system should enable use cases, not hard-code them
- Platform-dependent configuration that has to be applied to the system to realize the desired use case

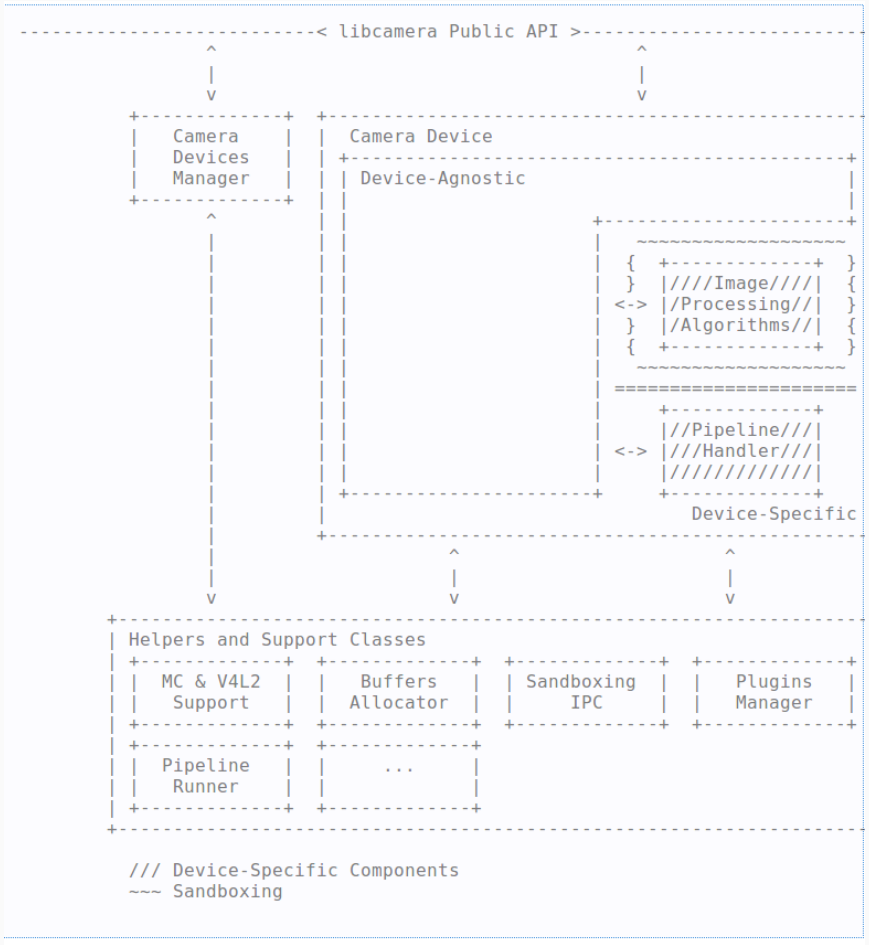
The case for a camera stack

Applications got stuck

- Regular desktop application (Cheese): UVC cameras
- Gstreamer is the most common way to interface with cameras
- On embedded systems the 'system configuration' is usually a collection of scripts

The case for a camera stack

libcamera



The case for a camera stack

libcamera

- Isolate device-specific components

The case for a camera stack

libcamera

- Isolate device-specific components
  - A camera pipeline handler is hardware support
  - Vendor should ideally provide one as part of their BSPs
  - To get them on-board quite some critical mass is needed

The case for a camera stack

libcamera

- Provide a generic API and a standardized set of controls to applications

The case for a camera stack

libcamera

- Provide a generic API and a standardized set of controls to applications
  - Abstract away platform specific details
  - Adaption layers: gstreamer, Android Camera3, V4L2 emulation

The case for a camera stack

libcamera

- Library of 3A algorithms

The case for a camera stack

libcamera

- Library of 3A algorithms
  - Framework for open source implementations to compete with proprietary ones
  - State of the art: Raspberry Pi IPA library

Mobile Devices

Does this apply to FOSS mobile devices too ?

Thank you for your attention

I hope you will enjoy the rest of the discussion