Image Capture On Embedded Linux Systems

Jacopo Mondi

FOSDEM 2018

Who am I

Hello, I’m Jacopo

- *jacopo@jmondi.org*
- *irc: jmondi freenode.net*

- Linux kernel and embedded software engineer
- Renesas Linux kernel team

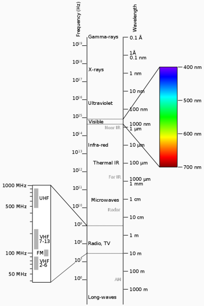
Motivations

- Light, color, pixels
- Image sensor
  - Anatomy
  - Integration
- Image Data transmission
- Video4Linux2
  - Basic architecture
  - Memory management
  - Image streaming
  - Media controller APIs

Light, colors, pixels

Color is not an absolute value

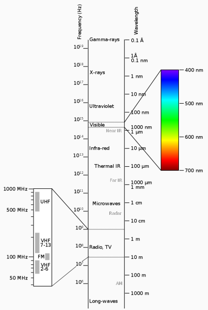
- Neural response to physical properties of electromagnetic radiations
- Visible light has a well defined interval (390 nm to 700 nm)
- Not all species and not all humans perceive colors in the same way



Light, colors, pixels

The human eye is more sensitive to three frequencies

- *short*: blue color
- *medium*: green color
- *long*: red color



Mathematical correlation between photo-optic properties and perceived color

Spectral power distribution

- We can describe a radiant emission of visible light as the intensity of a photo-optic property in function of the frequency of its component

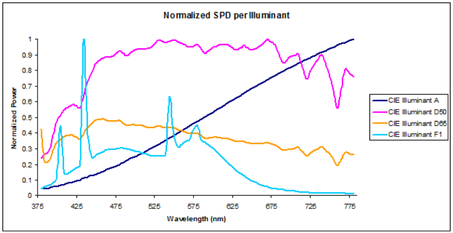


Figure: Spectral power distribution of standard illuminants. From: commons.wikimedia.org

Light, colors, pixels

The (very simple) **LMS** color space

- Samples of neural stimulus received by the human eye
- Samples on well-known wavelength
- The ***L****ong* ***M****edium* ***S****hort color space*

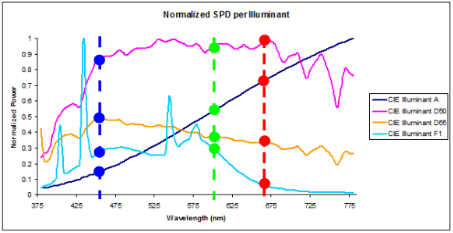


Figure: Long-Medium-Short wave length sampling

Light, colors, pixels

LMS Colorspace

- LMS is theoretical tool too simple scheme to represent real use cases

CIE 1931 defined colorspaces:

- *RGB color space*:
  Red **G**reen **B**lue primary colors
- *XYZ color space*:
  luma component **Y** and associated *chrominances* **x** and **z**

Notable color spaces: *sRGB, Adobe RBG, CYMK...*

Light, colors, pixels

Colorspaces and color encodings

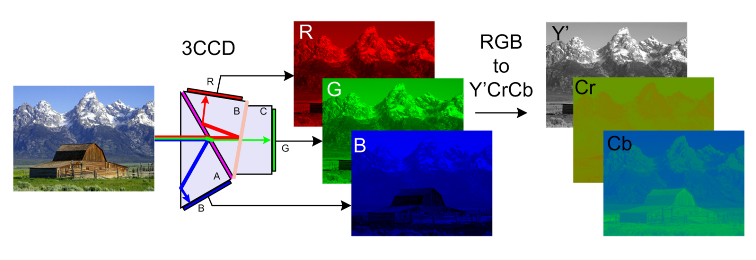


Figure: From: commons.wikimedia.org

Light, colors, pixels

Color encoding schemes

- We have a mathematical model to represent values of a ”color” with a tuple
- As we live in a digital world, we can now use those values to transmit the most basic information an image is composed of: **a pixel**

Question: if we have to describe a single pixel with at least 3 digital values, how big would an image composed b bee by 1280x800 pixels?

Light, colors, pixels

Color encoding schemes

- We have a mathematical model to represent with a tuple of values a ”color”
- As we live in a digital world, we can now use those values to transmit the most basic information an image is composed of: **a pixel**

Answer: 24,5Mbit with a very limited color resolution (0-255) $\rightarrow$ That’s bad

Light, colors, pixels

It is highly unpractical to sense all 3 color components for each pixel of a sensor's pixel matrix

- Image resolution vs sensor size ratio
- Required bandwidth for digital information transmission
- Production costs and dimension not justified by resulting performances for most use cases

Light, colors, pixels

Bayer filter

- A *Bayer filter* is an arrangement of light filters on top of a CMOS sensor photo-receptors
- Each 'pixel' transports a single color information

Figure: From: http://www.cambridgeincolour.com

Light, colors, pixels

Bayer filter

- The full pixel color is re-constructed by *demosaic* and *interpolation* with neighbor pixels
- Reduces the required transmission bandwidth and sensor size

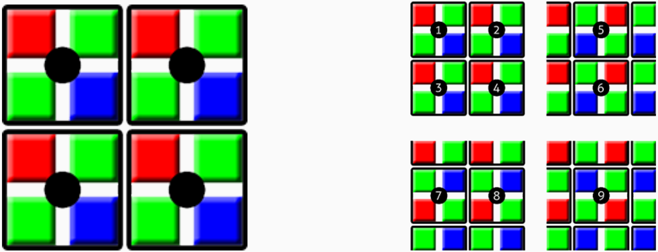


Figure: From: http://www.cambridgeincolour.com

Integration diagram

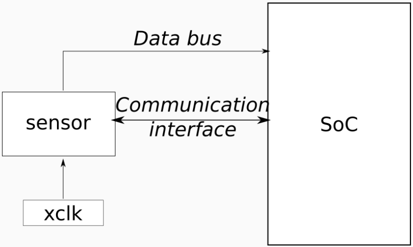


Image sensors

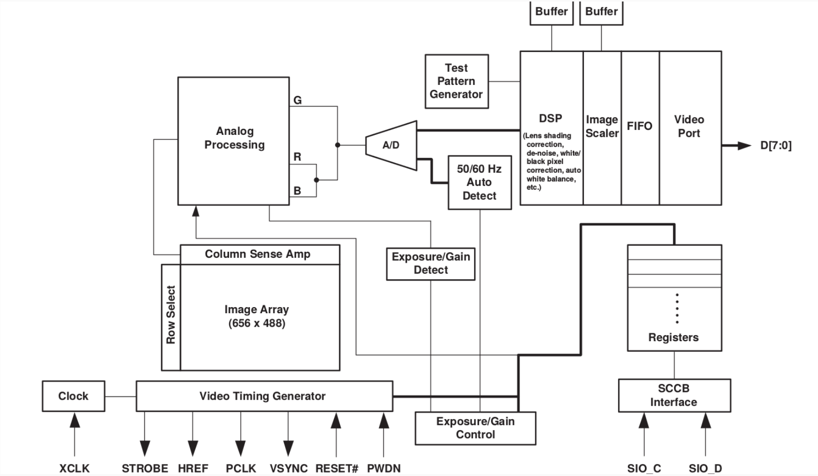


Image sensors

- Grid of CMOS photo-detectors: *Pixel Grid Array*
- Bayer pattern: RGB color filter array
- Color filter disposition: RGGB - BGGR - GRBG - RGGB etc.

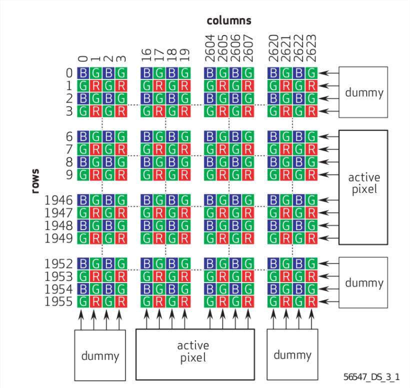


Image sensors

- Image format control:
  - Pixel encoding: RGB555, YUV422, YUV420 etc
  - Image manupulation: Cropping, binning, zoom
  - Advances features (ie. 3A), mirroring, flipping etc

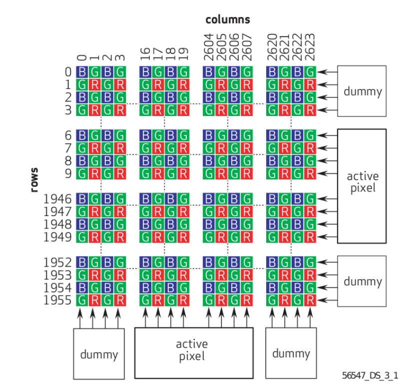


Image sensors data transmissoin interfaces

- Mainly two data bus categories:
  - Parallel (BT.601 - BT.656)
  - MIPI Serial camera interface (MIPI CSI-2)

- Parallel bus:
  - Lower data rate, lower resolutions, more wires..
  - Easier integration, cheaper, ”easy” debug
  - Usually found in industrial/automation contexts, hobbyist projects, older system in general

- MIPI Serial bus:
  - Higher data rate, highly integrated, less wires
  - Hard to integrate, hard to debug, more expensive
  - Mobile devices, cameras, and new designs in general

Image sensors data interface: BT.601

- VSYNC - HSYNC/HREF
  vertical/horizontal synchronization signals
- PCLK: pixel clockout reference clock
- 8+ parallel data lanes

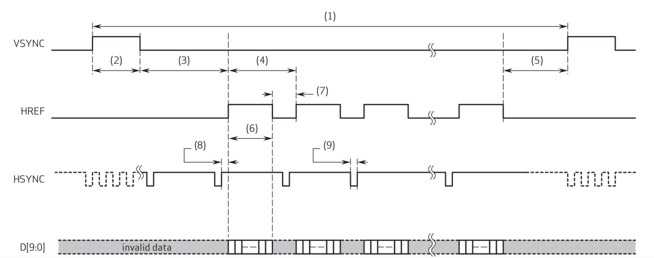


Image sensors data interface: MIPI CSI-2

MIPI CSI-2 specifications not only define the physical layer, image formats and data transmission protocol.

- Physical layers: D-PHY, C-PHY
- Data transmission protocol:
  - Media bus image formats:
    RGB or YUYV permutations
  - Short packets for synchronization signals (line/frame start/end)
  - Long packets for actual data with header for data description

Image sensors data interface: MIPI CSI-2

Physical layers

- MIPI D-PHY
  - Differential lines signals
  - Up to 1Gbps per lane
  - 1 clock signal and 1 to 4 data lanes
- MIPI C-PHY
  - Differential data lanes with embedded clock (3 pin)
  - Up to three ”trios”
  - Up to 5.7Gbps per ”trio”

Image sensors data interface: MIPI CSI-2

- Packet-oriented protocol
  - Short packets for synchronization:
    Frame start/end - Line start/end
  - Long packets for data
- A data stream is a sequence of pixel data enclosed in a <FS> <FE> sequence
- Each data packet is identified by a DT and a VC specified in its header

Image sensors data interface: MIPI CSI-2

Different data stream can be multiplexed on the same physical bus

- Data type (DT)
  - Data type identifier as defined by CSI-2 specs
  - Different image formats interleaved in the same stream
- Virtual channel (VC) interleaving
  - Channel identifier: [0-3 or 0-6]
  - Multiple streams interleaved
  - Each stream can be data type interleaved

Image sensors data interface: MIPI CSI-2

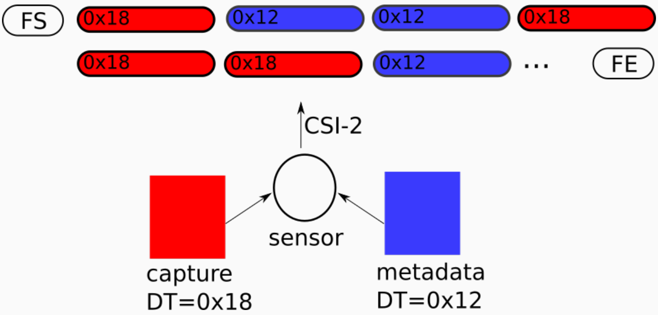
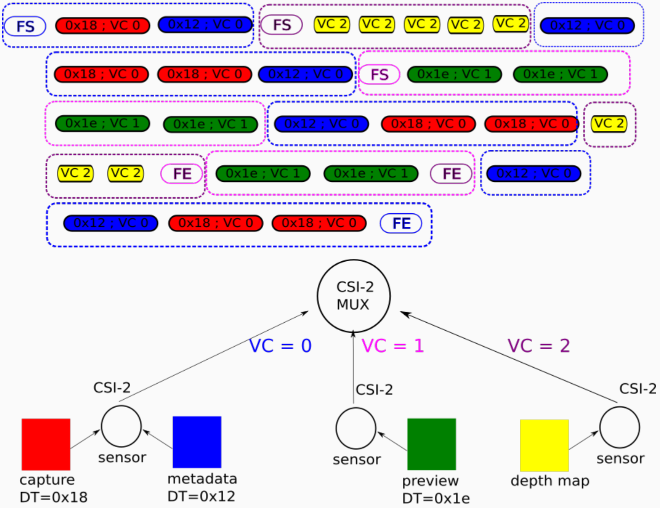
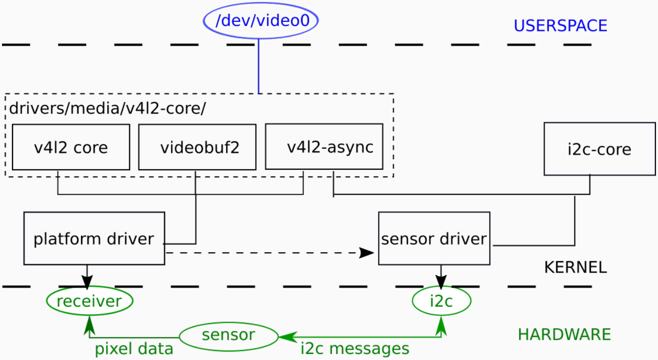


Image sensors data interface: MIPI CSI-2



Video4Linux2: a basic use case

Basic use case: single sensor connected to a video receiver



Video4Linux2: basic workflow

Preliminary operations

- Open the video device node
- Control the video device through V4L2 IOCTL:
  - Query capabilities to make sure the device can stream
  - Set image format on platform and sensor drivers
  - Set image size on platform and sensor drivers
  - Set stream parameters (frame rate) on platform and sensor drivers

Video4Linux2: memory allocation

Video memory requirements

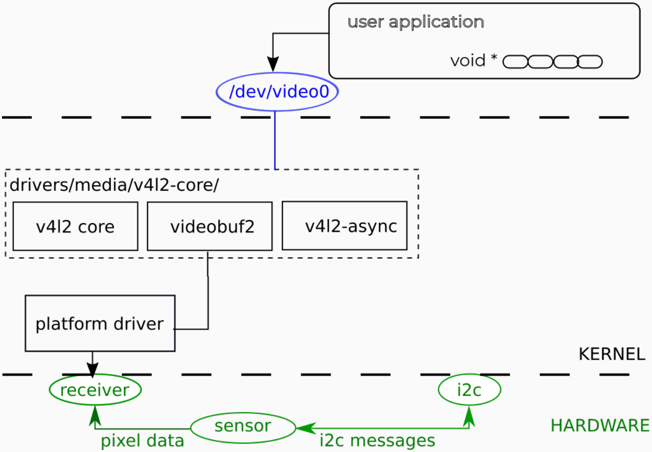
- DMA capable memory (often implies contiguity if DMA engine do not support s/g operations)
- Accessible by CPU and devices (in case of IOMMU)
- Possibly shared between different subsystems to reduce userspace copies

Video4Linux2: memory allocation

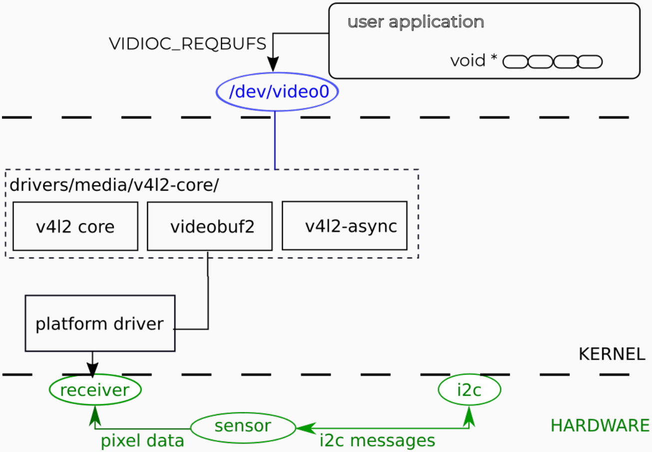
Three memory allocation model

- Kernel uses pointers to userspace buffers
- mmap of kernel buffers in userspace land
- Buffer sharing through DMABUF

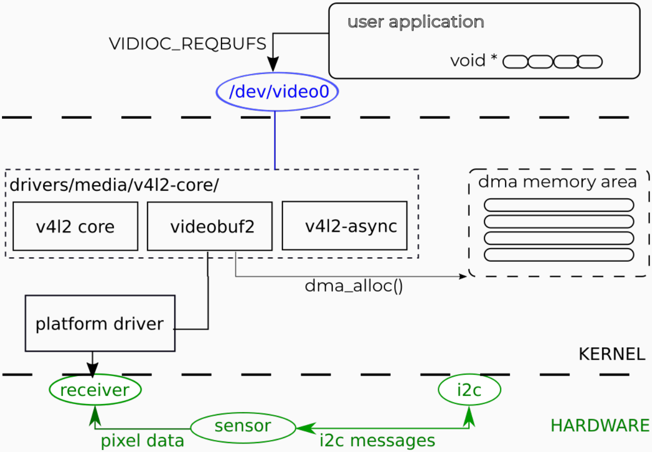
Video4Linux2: memory allocation - mmap



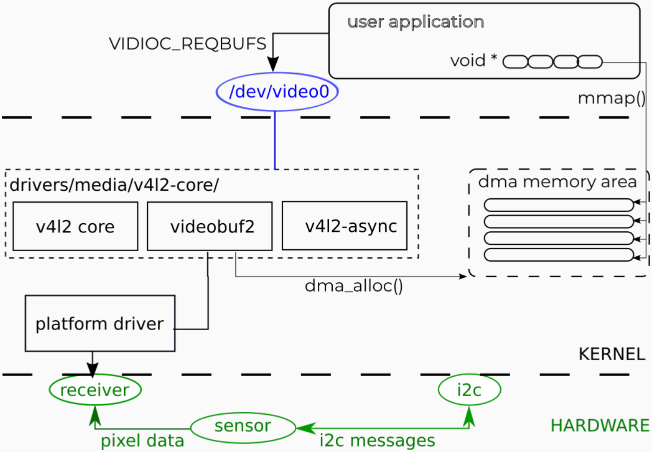
Video4Linux2: memory allocation - mmap



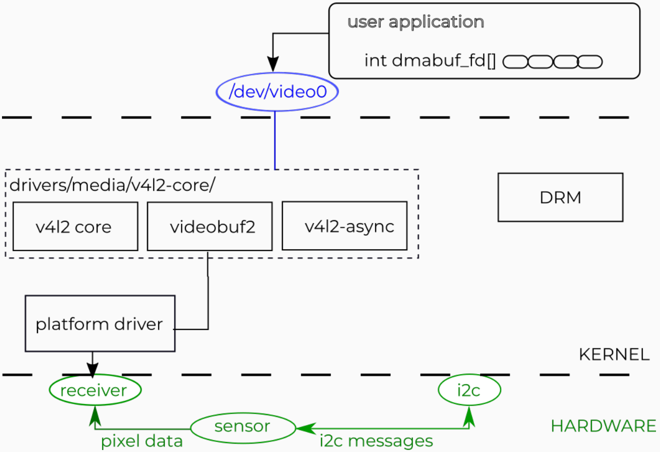
Video4Linux2: memory allocation - mmap



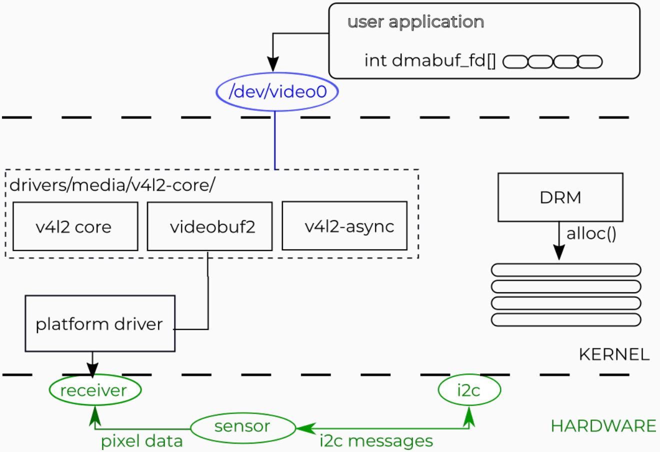
Video4Linux2: memory allocation - mmap



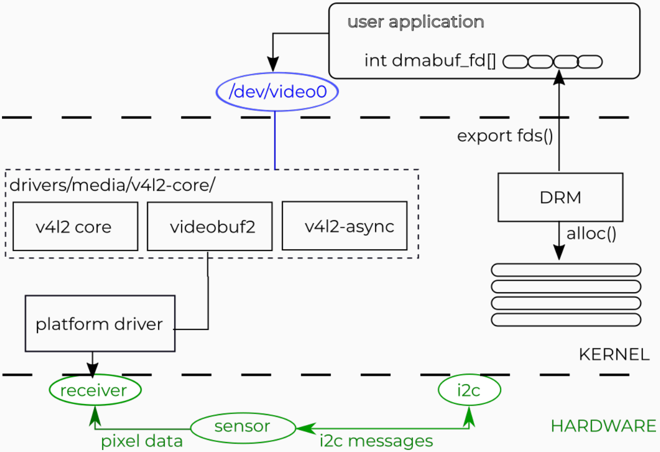
Video4Linux2: memory allocation - dmabuf



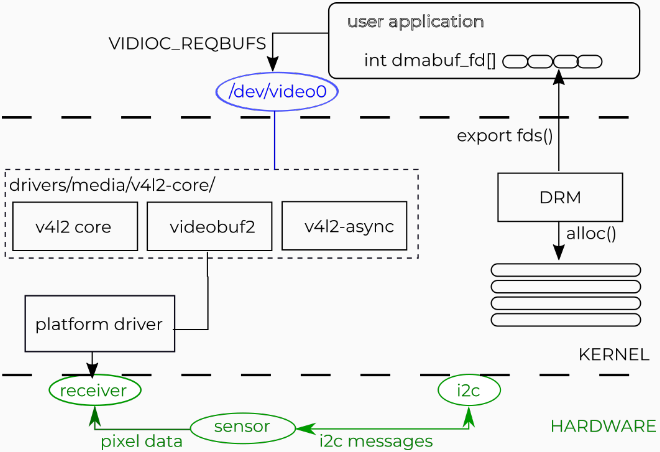
Video4Linux2: memory allocation - dmabuf



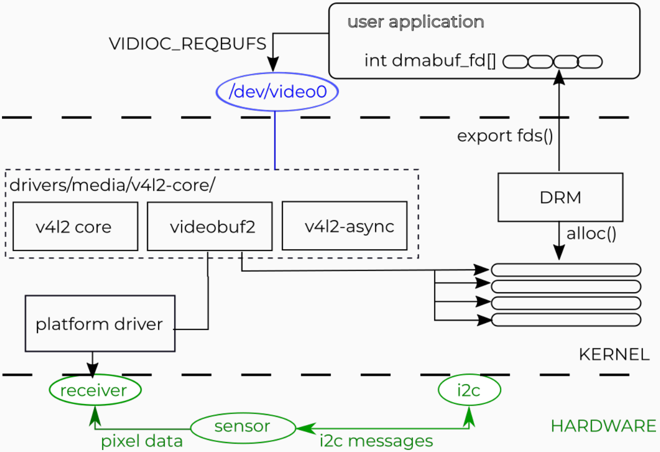
Video4Linux2: memory allocation - dmabuf



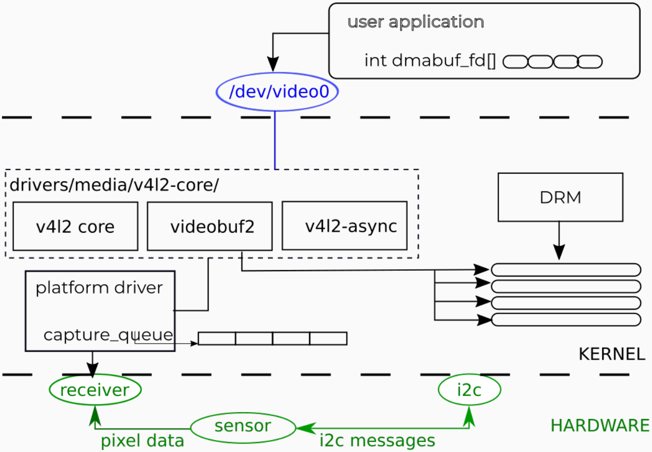
Video4Linux2: memory allocation - dmabuf



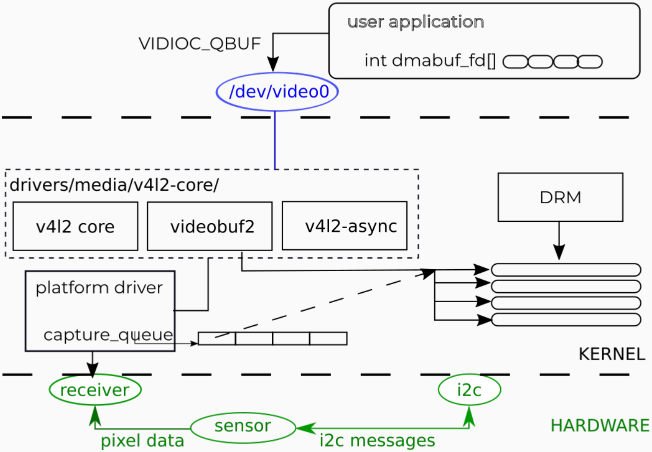
Video4Linux2: memory allocation - dmabuf



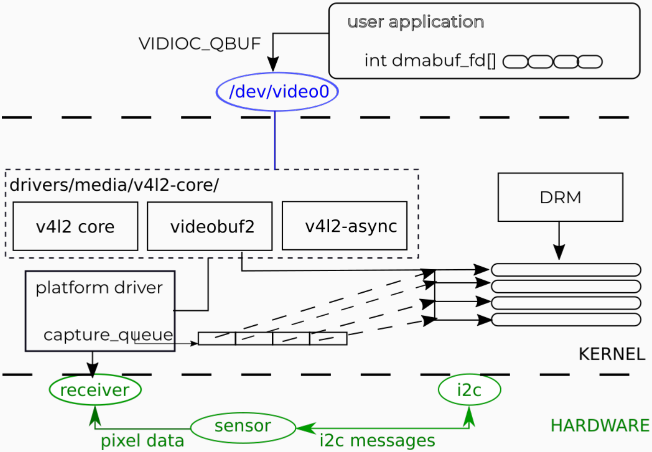
Video4Linux2: zero copy image streaming



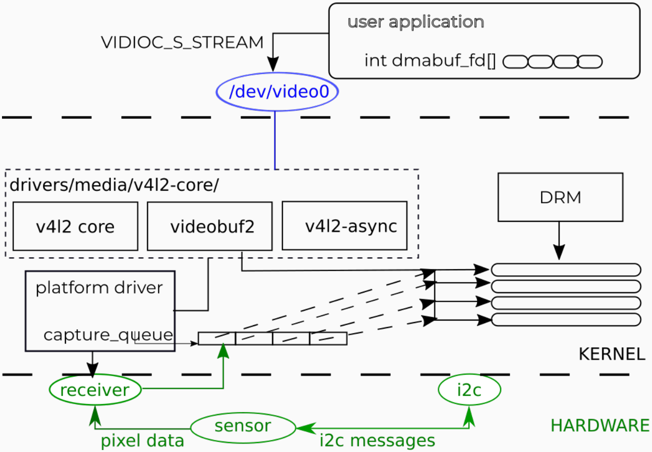
Video4Linux2: zero copy image streaming



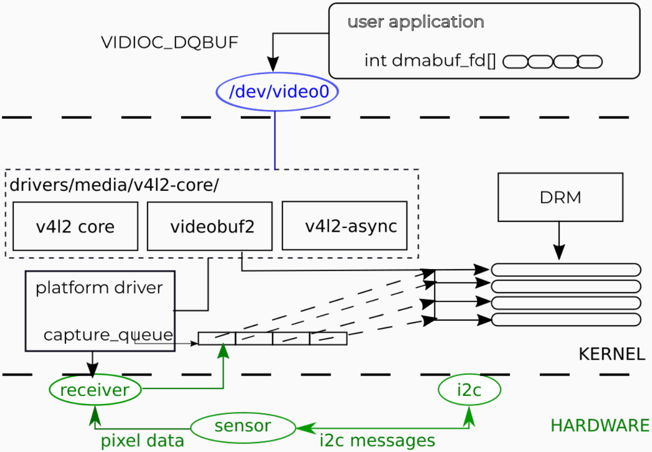
Video4Linux2: zero copy image streaming



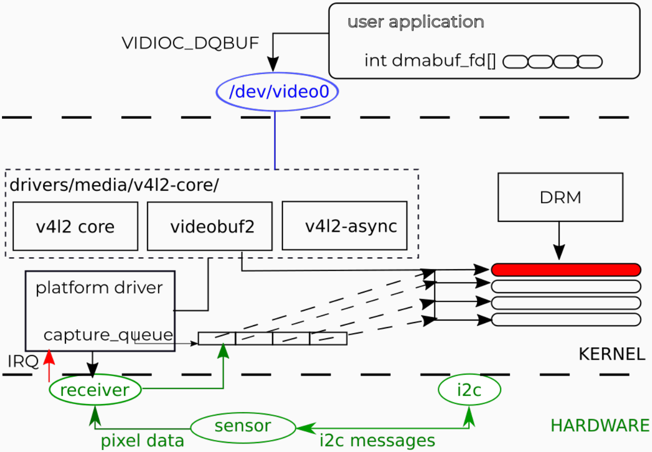
Video4Linux2: zero copy image streaming



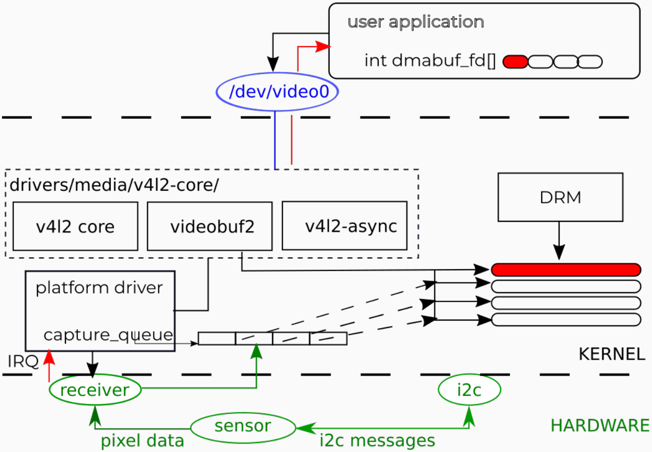
Video4Linux2: zero copy image streaming



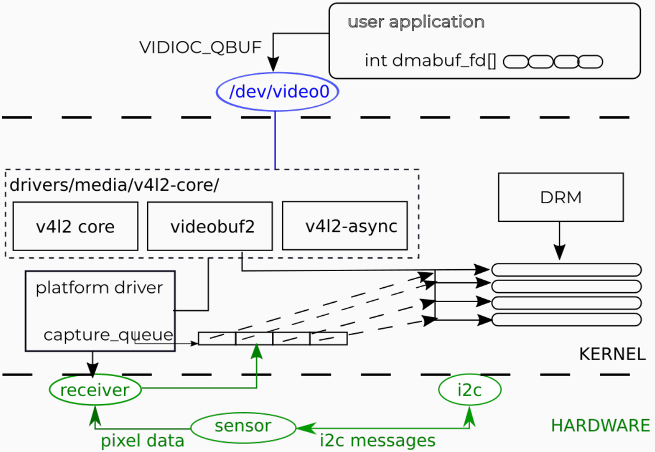
Video4Linux2: zero copy image streaming



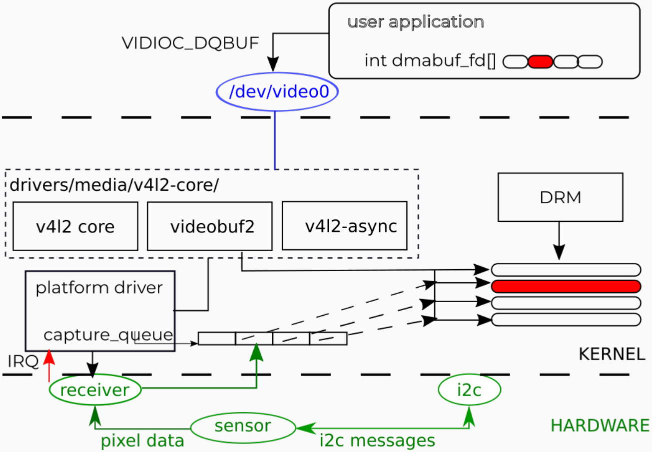
Video4Linux2: zero copy image streaming



Video4Linux2: zero copy image streaming



Video4Linux2: zero copy image streaming



Video4Linux2: platform drivers

`drivers/media/platform/`

- Transfer image data from internal buffers to system memory
- Perform transformations on the received images before presenting them to userspace
- Implement user space API through video device abstraction (or through media controller...)
- Handle IRQs and program receiver interface DMA to actually capture images

Video4Linux2: image sensor drivers

`drivers/media/i2c/`

- Control the image sensor through I2c transaction
- Respond to platform driver calls to set/get streaming parameters
- Start/stop sensor when platform driver requires data

Video4Linux2: Device tree bindings

`Documentation/devicetree/bindings/`

- Bindings defines a driver/subsystem ABI
- How drivers expects to be instantiated
- Which and how supply parameters to a driver

`Documentation/devicetree/bindings/media/`

- Video devices: how do they connect to each other

Video4Linux2: Device tree bindings

Platform drivers - Image receivers

```
.. /bindings/media/video-interfaces.txt

device {
      #address-cells = <1>;
      #size-cells = <0>;

      port@0 {
                reg = <0>;
                endpoint {
                        "endpoint properties";
                      remote-endpoint = <&phandle-0>;
                };
      };
```

Video4Linux2: Device tree bindings

Endpoint properties

- Parallel input bus

```
vsync-active = <1> / <0>;
hsync-active = <1> / <0>;
data-active = <1> / <0>;
...
```

- CSI-2

```
data-lanes = <1 2 ...>
clock lanes = <0>
link-frequencies = <210000000>
```

Video4Linux2: platform drivers

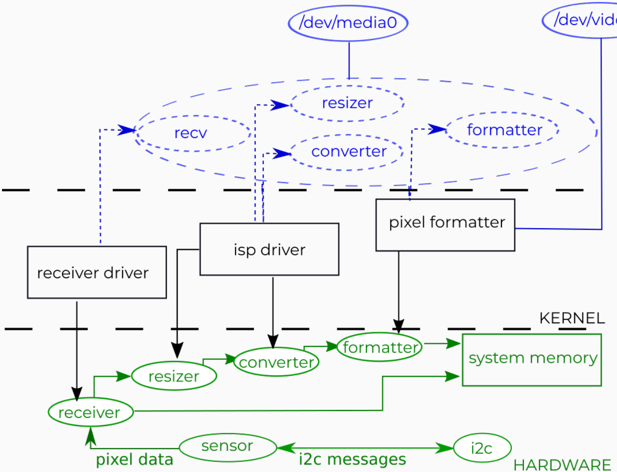
```
static int platform_probe(struct i2c_client *client) {
        priv = devm_kzalloc(sizeof(*priv));

        devm_ioremap(resources);
        devm_request_irq(irq)

        pm_runtime_enable();

        platform_parse_dt(dev->of_node);
        v4l2_async_notifier_register(priv->notifier);

        return 0;
}
```

Video4Linux2: the media controller

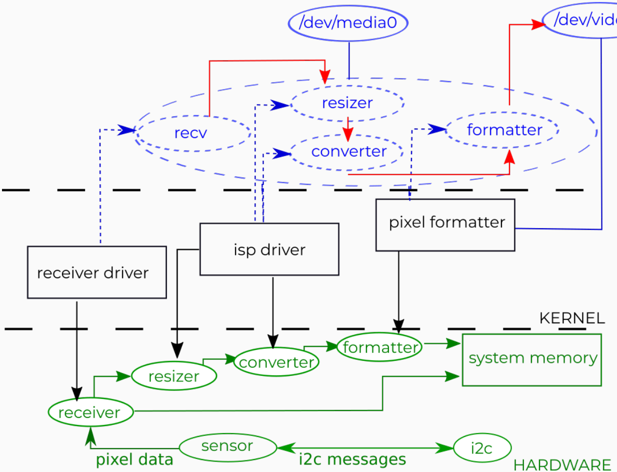
The V4L2 video device APIs are not enough to represent the complexity in modern SoCs

- Dedicated IP blocks on the SoC for image transformation
- media-controller: graph of media *entities* with *sink* and *source* pads
- Each *entity* can be linked to another entity
- media-controller allows the creation of image manipulation pipelines

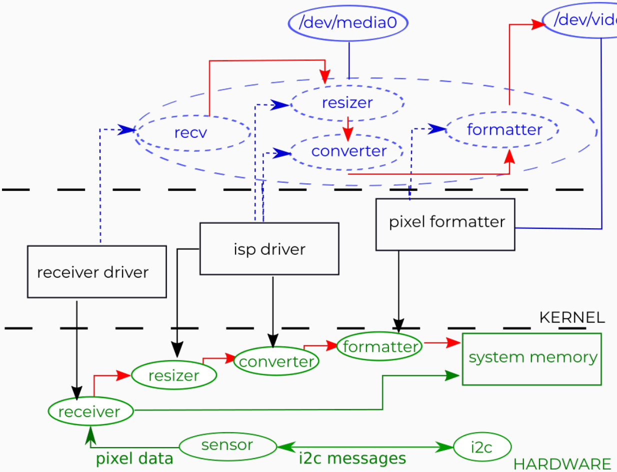
Video4Linux2: the media controller



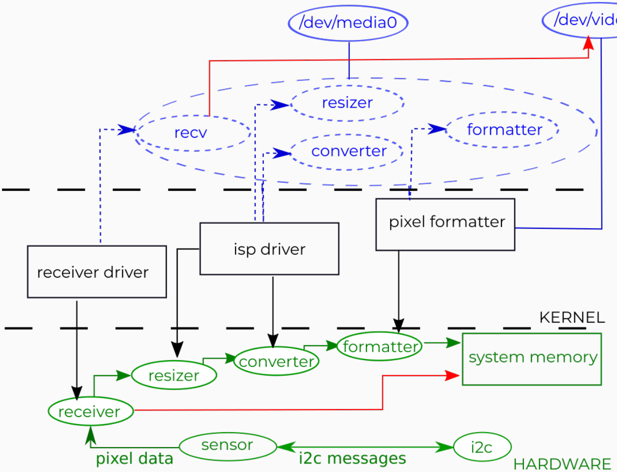
Video4Linux2: the media controller



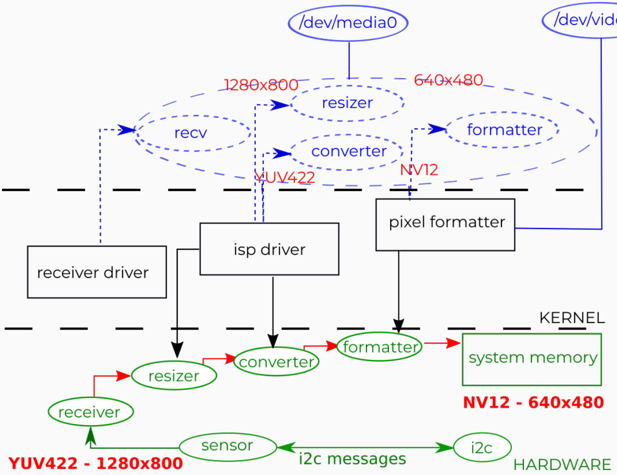
Video4Linux2: the media controller



Video4Linux2: the media controller



Video4Linux2: the media controller



Video4Linux2: the media controller

The media controller drawbacks

- Pipeline creation and management all in userspace
- System boots in a non-usable way
- Video devices vs video sub-devices APIs
- Complexity some times not justified for simple use cases

Thank you!

Questions?