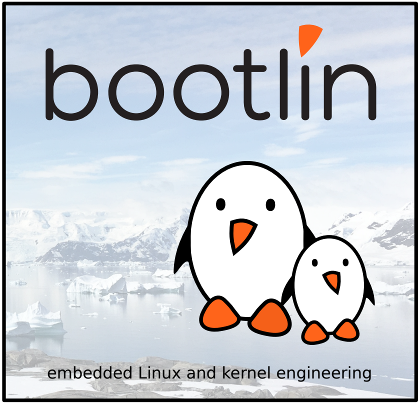
From the Camera Sensor to the User

the Journey of a Video Frame

Maxime Chevallier

maxime.chevallier@bootlin.com



Maxime Chevallier

- Linux kernel engineer at Bootlin.
  - Linux kernel and driver development, system integration, boot time optimization, consulting...
  - Embedded Linux, Linux driver development, Yocto Project & OpenEmbedded and Buildroot training, with materials freely available under a Creative Commons license.
  - `https://bootlin.com`
- Contributions:
  - Worked on network (MAC, PHY, switch) engines.
  - Contributed to the Marvell EBU SoCs upstream support.
  - Worked on Rockchip's Camera interface and Techwell's TW9900 decoder.

Preamble - goals

- Discover the hardware components and protocols involved in video cameras
- Understand how all these components chain together and how to configure them
- See various real-life hardware designs and use-cases

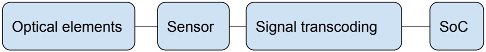
Video Acquisition Hardware

Maxime Chevallier
maxime.chevallier@bootlin.com

Corrections, suggestions, contributions and translations are welcome!

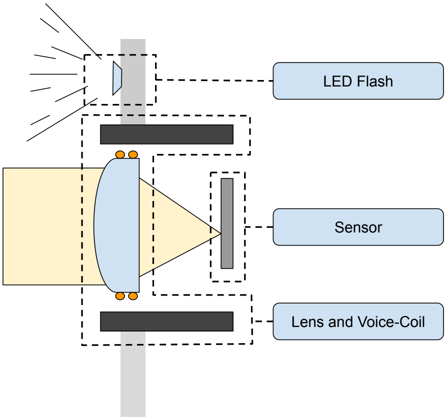
embedded Linux and kernel engineering

Necessary components



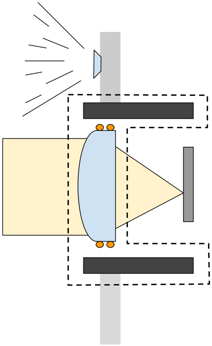
- Optical to Electrical conversion : Lenses and Sensors
- Signal transmission : Digital and Analog protocols
- Signal handling : Controllers and Signal Transcoders
- Image transformation : ISPs, Image Encoders/Decoders

Image acquisition setup



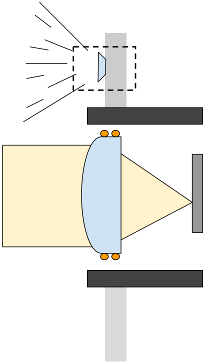
Lens

- Lens :
  - Controls the focus of the incoming light rays
  - Can be adjusted for manual or auto-focus
- Voice Coil Actuator :
  - Lens position is adjusted by a Voice Coil actuator
  - A wire coil is attached to the Lens
  - The Lens assembly sits in a static magnetic field
  - The coil is drived by a DAC providing adjustable current
  - Mostly controlled through I^{2}C
  - `MEDIA_ENT_F_LENS`

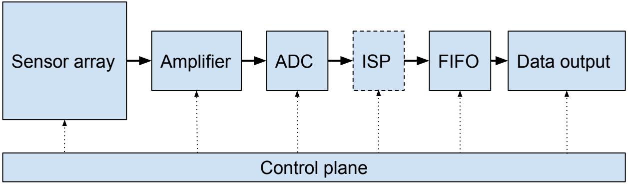


Flash

- High-power LED driver
- Sometimes include a *privacy indicator*
- Controlled through I^2C
- `MEDIA_ENT_F_FLASH`

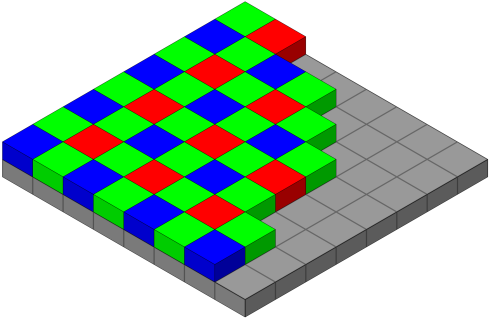


Sensor



- Converts an optical signal to an analog electrical signal
- Uses CCD or CMOS technologies
- Include internal ADCs and amplifiers
- Sometimes include an Image Signal Processor
- Data plane uses a dedicated protocol
- Control plane mostly uses I^2C

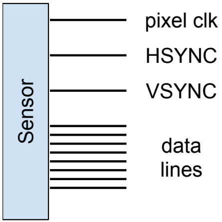
Digital Sensors



- Sensors acquire 3 color components, arranged in a grid
- These colors are sometimes sent raw : `RGGB`, `RGBG`, etc.
- Conversion from raw sampling to pixel value : `debayering`
- Also called `demosaicing`, can be done in-situ
- On basic sensors, this need to be done in the Host
- Possibly costly operation, depending on the algorithm

Digital transmission protocols : Raw Parallel

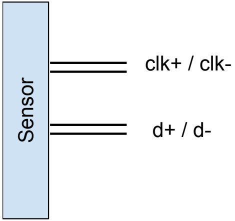
- Simple approach : Transmit data and sync signals
- Consists of the following lines :
  - Parallel data lines : 8 to 12 bits
  - Pixel Clock : Ticks every pixel sent
  - HSYNC : Toggles on line end
  - VSYNC : Toggles on frame end
- Often needs post-processing, such as `demosaicing`



Digital transmission protocols : CCP2

Compact **C**amera **P**ort **2**

- Serialized interface
- Sync signals embedded in data
- Uses differential pairs to transmit data and clocks
- PHY Layer is based on `subLVDS`
- 4 pins needed : 2 for clk and 2 for data
- Up to 650 mbps



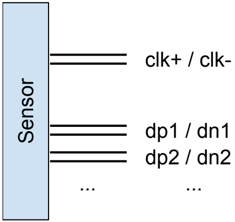
Transmission protocols : MIPI CSI

Camera **S**erial **I**nterface

- Standard from the MIPI Alliance
- Multiple layers defined :
  1. **PHY** : Physical transmission
  2. **Lane Management** : Lane distribution and merging
  3. **Low Level Protocol** : Checksuming, Error Correction
  4. **Application** : Pixel to Byte conversion
- CSI-2 is the most used
- CSI-3 : Bi-directional protocol

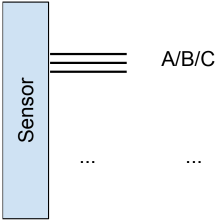
MIPI D-PHY Layer

- Serialized interface
- Sync signals embedded in data
- Uses differential pairs to transmit data and clocks
- Minimum 4 pins : 2 for clk and 2 per lane
- Up to 4 lanes
- Up to 6 Gbps

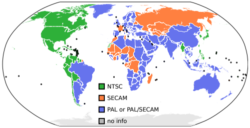


MIPI C-PHY Layer

- Serialized interface
- Sync signals and clock embedded in data
- 3-levels signals : High, Med and Low
- 3 lines per lane
- 16 bits transmitted over 7 symbols (in quinary)
- Up to 3 lanes
- Up to 41 Gbps



Analog transmission protocols



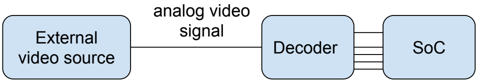
- Video Broadcasting protocols
- Designed for transmission over an analog media
- Decomposes video into components : Y, Cr, Cb
- Y : Luminance (Black and White image)
- U, V : Chrominance (Color information)
- **PAL** : Europe
- **NTSC** : USA, Japan
- **SECAM** : France, Eastern Europe, Russia

Interlacing

- Increase the perceived framerate with the same bandwidth
- Transmit the odd lines, then the even lines
- Each part is called a `field`
- Requires a compatible display
- Else the video need to be `deinterlaced`
- Requires some signal processing to get correct results

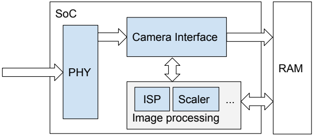
Interlacing artifacts

Analog to Digital Transcoders



- Converts an analog video signal into a digital signal
- Can support multiple input standards
- Can embed a small `ISP` for simple cropping/scaling
- Converts into a digital video standard :
  - BT.656 for `PAL` and `NTSC` standards
  - BT.1120 for higher resolution standards
- Parallel or Serial interfaces supported by BT.* standards

Host Interface



- Hardware blocks located in the SoC
- Has lots of different features depending on the Hardware :
  - PHY layer support
  - Image processing (simple or advanced), including :
    - Scaling, Cropping
    - Deinterlacing
    - Pixel format conversion
- Stores the frame into buffers using DMA
- The V4L2 framework and the media controller API supports these blocks

Image processing

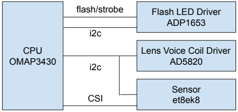
- Cropping : Remove areas from the image
  - Easy to perform
- Scaling : Resize the image
  - Can require complex algorithms
- Deinterlacing : Recompose an interlaced stream
  - Joining fields is easy
  - Removing artifacts can be complex
- Pixel format conversion
  - Debayering : Convert raw sensor data to a usable image format
  - Colorspace conversion : `RGB` to/from `YUV`
- 3A Processing
  - Auto exposure : Adjust the brightness, control the sensor **gain**
  - Auto focus : Adjust the focal point, control the **lens position**
  - Auto white balance : Correct the colors

See the talk on ISP drivers by Helen Koike earlier today !

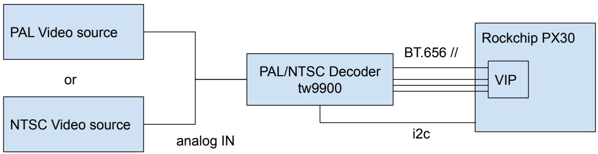
Example : Nokia N900

- Smartphone with full Linux support
- Based on TI OMAP3 SoC
- Has a Flash LED driver
- Voice Coil controlled through a DAC
- CSI Sensor controlled through I2C
- DTS found in `arch/arm/boot/dts/omap3-n900.dts`

Nokia N900



Example : Custom Product



- Based on Rockchip PX30 SoC
- Remote video source through an analog signal : `PAL` or `NTSC`
- Decoded through a `tw9900`, which auto-detects the standard
- BT.656 Parallel bus to transmit the digital signal
- Uses the `VIP` Camera Interface (Upstreaming in progress)
- Fields are simply joint, not a full de-interlacing

Linux Support

- All the above components and more are supported by Linux
- Wide variety of topologies, challenging to have a full-featured infrastructure
- `V4L2` Gives the infrastructure to support Video Cameras
- `V4L2` Also deals with buffer management and interaction with userspace
- The `Media Controller` API allows configuring each block through `subdevs`
- Complex devices can be handled in userspace with `libcamera`
- Welcoming community :)

Conclusion

- The number of technologies involved can be overwhelming
- Old analog terminologies and technologies still apply today
- However, Linux support is pretty good
- `V4L2` and the `media controller` API allow complex use-cases

Thank you!
Questions? Comments?

Maxime Chevallier — maxime.chevallier@bootlin.com

https://bootlin.com/pub/conferences/2019/elce/chevallier-network-classification-offload/