Giving Linux a Camera Stack: libcamera's 3 Years Journey and Exciting Future

Embedded Linux Conference 2021
Seattle, 2021-09-28

Laurent Pinchart
laurent.pinchart@ideasonboard.com
Once Upon A Time
In the beginning...
... were simple devices
A monolithic API for TV grabbers and webcams alike.

Enables development of universal applications.

High-level controls, the TV signal provides a good image already.
Hi everybody,

after more than two years of development the Linux UVC driver is mostly ready to jump the fence and get included in the mainline kernel.

This driver aims to support video input devices compliant with the USB Video Class specification. This means lots of currently manufactured webcams, and probably most of the future ones.

I plan to submit the driver through the V4L subsystem, but I'd like it to get a proper review on both the linux-usb and video4linux mailing lists first.

Given the size of the patch I'm open to any suggestion that would make the review process easier.

Laurent Pinchart

(my humble 25 August 1991 moment)
Embedded Cameras

Palm Treo 650 (PXA270, 0.3MP Camera)
From: Guennadi Liakhovetski <g.liakhovetski@pengutronix.de>
To: video4linux-list@redhat.com
Subject: [PATCH 1/6] soc_camera V4L2 driver for directly-connected
        SoC-based cameras
Date: Tue, 5 Feb 2008 18:46:13 +0100 (CET)

This driver provides an interface between platform-specific camera
busses and camera devices. It should be used if the camera is connected
not over a "proper" bus like PCI or USB, but over a special bus, like,
for example, the Quick Capture interface on PXA270 SoCs. Later it should
also be used for i.MX31 SoCs from Freescale. It can handle multiple
cameras and / or multiple busses, which can be used, e.g., in
stereo-vision applications.

Signed-off-by: Guennadi Liakhovetski <g.liakhovetski@pengutronix.de>
What is a camera?
CMOS Sensor
CMOS Sensor
CMOS Sensor

Photon to electron conversion

Charge to voltage conversion

Analogue amplifier (gain)

Analogue to digital converter
Colour Filter Array

source: https://en.wikipedia.org/wiki/Bayer_filter
CFA Interpolation

source: https://en.wikipedia.org/wiki/Bayer_filter
CFA Interpolation

source: https://en.wikipedia.org/wiki/Bayer_filter
Lens Shading
IQ Tuning

source: https://www.flickr.com/photos/davedugdale/15043975135
Back To V4L2
Smart Sensor (a.k.a. YUV Sensor)
So Small, Yet So Smart
In the beginning were simple pipelines...

V4L2 Goes Embedded
... and they were simple to control, with a single API.

V4L2 Goes Embedded
Then the world became complex
The libcamera Journey
OMAP3 Camera in Nokia N900

Then the world became complex...
... and application developers were left suffering.
Solutions were proposed...
... but never implemented.
The world turned dark...
...then hope came back.
libcamera provides a complete userspace camera stack.

The ‘Mesa’ of the camera world.
Camera Devices & Enumeration

Userspace

Kernel

\[\text{libcamera enumerates cameras...}\]
Streams

It supports multiple concurrent streams for the same camera...
Per-Frame Controls

... and per-frame controls.
Image Processing Algorithms (3A)

Image Processing Algorithm are loaded as external modules.
Adaptation layers offer backward compatibility with existing APIs...
Adaptation

... and integrate libcamera with other operating systems.
The libcamera Journey
Our Initial Targets

Intel IPU3 (Kaby Lake) on HP Chromebook x2

USB Video Class (UVC)
Tests

Unit testing

Test, Test, Test
Acer Chromebook Tab 10

ROCK PI 4

And ARM Too (RK3399)
Raspberry Pi 4 + IMX477
An open source camera stack for Raspberry Pi using libcamera

Since we released the first Raspberry Pi camera module back in 2013, users have been clamouring for better access to the internals of the camera system, and even to be able to attach camera sensors of their own to the Raspberry Pi board. Today we’re releasing our first version of a new open source camera stack which makes these wishes a reality.

(Note: in what follows, you may wish to refer to the glossary at the end of this post.)

We’ve had the building blocks for connecting other sensors and providing lower-level access to the image processing for a while, but Linux has been missing a convenient way for applications to take advantage of this. In late 2018 a group of Linux developers started a project called libcamera to address that. We’ve been working with them since then, and we’re pleased now to announce a camera stack that operates within this new framework.
Microsoft Surface Go 2
“I found libcamera based on this bug report, it seems to have the required userspace code to have ipu3 working on 5.0.”

(archseer)
Initial support status with work on kernel drivers from djrscally, kitakar5525 and qzed
It seems that it tends to be purple-ish when it's dark and tends to be green-ish when it's bright.

The sky after rain with leaf taken by surface-ipv3-cameras/ov8865_from_0v8856
I am currently finalizing a first work with libcamera and right now, I can get this kind of output (still some work to do as you can see :))).

Basically, this is AWB in a simple form and auto exposure almost ready for a first shot...
Camera support

linux-surface

Posted by jrevillard

I see that there is some activity here: https://github.com/jakeday/linux-surface/issues/145

I think it's better to trace it here isn't it?

Best,
Jerome

Maintainer note: There is a BountySource bounty available for this issue. If you'd like to see a camera driver consider donating to incentivise the work.

Hide

View in GitHub

SOLVE ISSUE

Overview  Backers (27)  Updates

TOTAL BOUNTY VALUE
$810.00USD

$10  $50  $500

USD

Select payment method

PayPal

CHECKOUT

Are you a developer who can solve this issue?

Got Started

Budget Expectations
libcamera
Today
The Camera Stack
The Camera Manager
The Camera Manager enumerates media devices and instantiates corresponding pipeline handlers.
Each pipeline handlers create and register one or more cameras.

The Camera Manager
The Pipeline
The pipeline handler interfaces with all kernel devices. It abstracts them and exposes video streams to upper layers.
Image Processing Algorithms (IPA) receive statistics from the hardware and compute optimal image parameters.
The Image Processing Algorithms

IPAs are separate modules that don’t access kernel devices directly. They only have access to their pipeline handler through the IPA API.
Pipeline Handler

3A

API

Out-of-tree (including closed-source) IPAs are sandboxed in a separate process. They communicate with the pipeline handler through IPC.

The Image Processing Algorithms
The IPC is handled in core components, transparently for both the pipeline handler and the IPA.
Many helper classes ease the implementation of pipeline handlers for device vendors.

The Helpers and Support Classes
The adaptation layer interface libcamera with other APIs and frameworks, with a single implementation for all supported devices.
A single Android camera HAL module implementation for all devices supported by libcamera.
“libcamerarsrc” offers a multi-stream source element for GStreamer applications.
Native V4L2 Compatibility

Native V4L2 applications are supported through a transparent compatibility layer.
<table>
<thead>
<tr>
<th>Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Multi-camera, multi-stream, per-frame control, hotplug</td>
</tr>
<tr>
<td>Supported platforms</td>
<td>Raspberry Pi 3&amp;4, Intel IPU3 (Kaby Lake), Rockchip RK3399, UVC, NXP i.MX7, Allwinner A31</td>
</tr>
<tr>
<td>IPA modules</td>
<td>Raspberry Pi, Intel IPU3 IPA module isolation with IPC (based on mojom IDL)</td>
</tr>
<tr>
<td>Adaptation layers</td>
<td>GStreamer source element (with multi-stream), Android camera HAL v3.3, V4L2 emulation</td>
</tr>
<tr>
<td>Tooling</td>
<td>Camera Tuning Tool (Raspberry Pi), tracing infrastructure and analysis script</td>
</tr>
<tr>
<td>Applications</td>
<td>cam (command line Swiss army knife), qcam (GUI), simple-cam (tutorial)</td>
</tr>
<tr>
<td>Documentation</td>
<td>Extensive API documentation and high-level tutorials and guides available</td>
</tr>
</tbody>
</table>
Guides

Pipeline Handler Writers Guide

Pipeline handlers are the abstraction layer for device-specific hardware configuration. They access and control hardware through the V4L2 and Media Controller kernel interfaces, and implement an internal API to control the ISP and capture components of a pipeline directly.

Prerequisite knowledge: system architecture

A pipeline handler configures and manages the image acquisition and transportation pipeline realized by specialized system peripherals combined with an image source connected to the system through a data and control bus. The presence, number and characteristics of these vary depending on the system design and the product integration of the target platform.

System components can be classified in three macro-categories:

- **Input ports:** Interfaces to external devices, usually image sensors, which transfer data from the physical bus to locations accessible by other system peripherals. An input port needs to be configured according to the input image format and size and could optionally apply basic transformations on the received images, most typically cropping/slicing and some format conversion. The industry standard for the system typically targeted by Bicameras is to have cameras compliant with the MIPI CSI-2 specifications, implemented on a compatible physical layer such as MIPI D.PHY or MIPI C.PHY. Other options are possible but less common, such as LVDS or the legacy BT.601 and BT.656 parallel protocols.

- **Image Signal Processor (ISP):** A specialized media processor which applies digital transformations on image streams. ISPs can be integrated as part of the SoC as a memory interfaced system peripheral or as standalone chips connected to the application processor through a bus. Most hardware used by Bicameras makes use of in-system ISP designs but pipelines can equally support external ISP chips or be instrumented to use other system resources such as a GPU or an FPGA IP block. ISPs expose a software programming interface that allows the configuration of multiple processing blocks which form an “Image Transformation Pipeline”. An ISP usually produces “processed” image streams along with the metadata describing the processing steps which have been applied to generate the output frames.

- **Camera Sensor:** Digital components that integrate an image sensor with control electronics and usually a lens. Interfaces to the SoC image receiver ports and is programmed to produce images in a format and size suitable for the current system configuration. Complex camera modules can integrate on-board ISP or DSP chips and process images before delivering them to the system. Most systems with a dedicated ISP processor will usually integrate camera sensors which produce images in Raw Layer format and further processing to it.

It is the responsibility of the pipeline handler to interface with these (and possibly other) components of the system and implement the following functionalities:

- Detect and register camera devices available in the system with an associated set of image streams.
- Configure the image acquisition and processing pipeline by assigning the system resources (memory, shared components, etc.) to satisfy the configuration requested by the application.
- Start and stop the image acquisition and processing sessions.

Documentation compiled from source tree, will be integrated in the website.
New Platforms

MediaTek Pumpkin i500 (MT8385)
New Platforms
New Platforms

Purism’s Librem products are the only modern high-end devices where you are in control and have complete visibility into the operating system, all bundled software, and the deeper levels of your computer. Every Librem comes with free international shipping, a one-year hardware warranty, and lifetime software updates with PureOS. And our software updates are intended to make your hardware run better over time, not worse.

Purism Librem 5
<table>
<thead>
<tr>
<th>Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>New platforms</td>
<td>MediaTek MT8385 (with YUV sensors), NXP i.MX8M Plus, Librem 5</td>
</tr>
<tr>
<td>Open-source IPA modules</td>
<td>Intel IPU3, Rockchip ISP, I.MX8M Plus</td>
</tr>
<tr>
<td>Reprocessing API</td>
<td>Work in progress in the libcamera core, Android HAL support will follow</td>
</tr>
<tr>
<td>Controls and properties</td>
<td>New controls and properties are continuously added on a per-need basis</td>
</tr>
<tr>
<td>API cleanups</td>
<td>Moving toward the API freeze for a 1.0 release, API changes will remain backward-compatible (extensible API design, d-pointer design pattern, …)</td>
</tr>
<tr>
<td>Language bindings</td>
<td>Python bindings in progress</td>
</tr>
<tr>
<td>Android LIMITED and FULL CTS compliance</td>
<td>Core infrastructure ready, controls and properties (static, control and dynamic metadata) being developed incrementally</td>
</tr>
</tbody>
</table>
Chromium (on MS Surface Go 2)
Embedded Camera

Exploratory Group for Embedded Camera and Sensors

HOSTED BY:

European Machine Vision Association

Kronos Group

Participation In Industry Initiatives
For Camera Vendors
++ Standard Android Camera HAL Implementation.

++ GStreamer, V4L2, ...

// Custom API for IPA module <-> pipeline handler communication.

++ libcamera wrapper classes reduce custom code.

++ No change on the kernel side architecture.

// Pipeline handler is ACME-specific development.

++ Development support available.

// Implementation changes may be required to mainline drivers.

Platform Enablement
We drive MC and V4L2 standardization and extensions development according to our needs.
Kernel APIs

We drive MC and V4L2 standardization and extensions development according to our needs.

libcamera is a userspace framework, not a hostile takeover of kernel development.
++ Standard Android Camera HAL Implementation.

++ GStreamer, V4L2, ...

!/ Custom API for IPA module <-> pipeline handler communication.

++ libcamera wrapper classes reduce custom code.

++ Pipeline handler is ACME-specific development.

++ Development support available.

!/ Implementation changes may be required to mainline drivers.

Platform Enablement
The libcamera core is licensed under the LGPL v2.1 or later.
The libcamera core is licensed under the LGPL v2.1 or later.

Changes need to be published according to the license. This includes pipeline handlers.
The libcamera core is licensed under the LGPL v2.1 or later.

Changes need to be published according to the license. This includes pipeline handlers.

Upstreaming is not mandatory but highly recommended (forks are costly to maintain).
Closed-source IPA modules are fully supported.

Pipeline handlers and IPA modules can link to third-party libraries.

The libcamera core, pipeline handlers and adaptation are licensed under LGPL v2.1 or later.

Kernel code is licensed under GPL v2.0.
An Exciting Future
<table>
<thead>
<tr>
<th>Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-stream controls</td>
<td>Concept approved, will be scheduled in the future.</td>
</tr>
<tr>
<td>Zero shutter lag</td>
<td>Will be possible through the reprocessing API. We are considering a high-level “use cases” library on top of libcamera for ZSL and similar features.</td>
</tr>
<tr>
<td>Exposure bracketing HDR</td>
<td>Similarly to ZSL, could be implemented in a “use cases” library. A solution for device-assisted HDR (hardware merging or software merging based on hardware-generated metadata) is needed.</td>
</tr>
<tr>
<td>Logical camera devices (W+T zoom, power saving,...)</td>
<td>Not planned yet, missing development and test platform.</td>
</tr>
<tr>
<td>Still image trigger sequence (focus &amp; flash)</td>
<td>Not planned yet, missing development and test platform.</td>
</tr>
</tbody>
</table>
## Future Features – Devices

<table>
<thead>
<tr>
<th>Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source IPA modules</td>
<td>Cross-platform core library, long term work to convince device vendors</td>
</tr>
<tr>
<td>GPU-based processing</td>
<td>Proof of concept shader code in qcam test application, should be leveraged to create GPU-based ISP for platforms without a hardware ISP (Librem 5).</td>
</tr>
<tr>
<td>New devices support</td>
<td>Ongoing discussions with SoC/system vendors, community-driven effort on legacy devices (any volunteer for the N900/N9 ?)</td>
</tr>
</tbody>
</table>
Capturing pixels with a camera involves a number of steps, from the ADC reading the photostates in the image sensor to the final pixel values that are ready for encode/display, with various processing and transmission taking place along the way. While simple cases put most of the heavy lifting on the image sensor's side (through its embedded processor) and use a simple parallel bus for transmission, advanced cases require more work to be done outside of the sensor. In addition, modern high-speed transmission buses also bring in more complexity. This talk will present how support for such an advanced use case was integrated into the mainline Linux kernel, using the Media and V4L2 APIs. It involves supporting a sensor using the raw Bayer RGB format, transmission over the MIPI CSI-2 bus as well as support for the Image Signal Processor (ISP) found on Allwinner platforms. A specific focus will be set on this ISP, with details about the features it implements as well as the internal and userspace APIs that are used to support it. The integration between all of the involved components will also be highlighted.

Speakers

Paul Kociatkowski
Embedded Linux Engineer, Bootlin

Paul joined Bootlin in 2018 and started with bringing support for the Allwinner VPU to mainline Linux. He went on to cover more topics related to graphics and multimedia, with various contributions to the DRM and V4L2 Linux subsystems as well as various related projects. Before that... Read More →
<table>
<thead>
<tr>
<th>Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAL v3.5(+)</td>
<td>On the roadmap, on hold due to lack of development and test platform. Future Android camera HAL API extensions will be implemented (including extensions to the libcamera core if needed).</td>
</tr>
<tr>
<td>Zero Shutter Lag</td>
<td>HAL-based ZSL implemented using the libcamera reprocessing API.</td>
</tr>
<tr>
<td>Features</td>
<td>Status</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Frameworks</td>
<td>PipeWire, OpenCV, Qt Multimedia, Electron, &lt;insert your framework here&gt;, ...</td>
</tr>
<tr>
<td>Applications</td>
<td>Firefox, OBS, &lt;insert your application here&gt;, ...</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>Chrome OS, Android, Linux distributions, Buildroot, OpenEmbedded, ...</td>
</tr>
</tbody>
</table>
libcamera-devel@lists.libcamera.org
irc://chat.freenode.net/#libcamera
laurent.pinchart@ideasonboard.com
Thank you.