Embedded Linux Conference Europe

Getting a Time of Flight Camera Working in Linux, the Full Story from Kernel to User Space

Bogdan Togorean
Analog Devices Inc.

#lfelc @twitterhandle

Agenda

- About me
- Time of Flight (ToF)
- Analog Devices 3D ToF
- User Space SDK
- ADI ToF special considerations
- Linux Kernel Driver – implementation details
- ADI ToF SDK – implementation details
- Supported Platforms Peculiarities
  - Qualcomm® APQ8016e (Dragonboard 410C)
  - Raspberry Pi
  - NXP I.MX8M Mini (Variscite DART-MX8M-Mini)
  - Nvidia Jetson Nano
  - Nvidia Xavier AGX
  - Rockchip RK3399pro

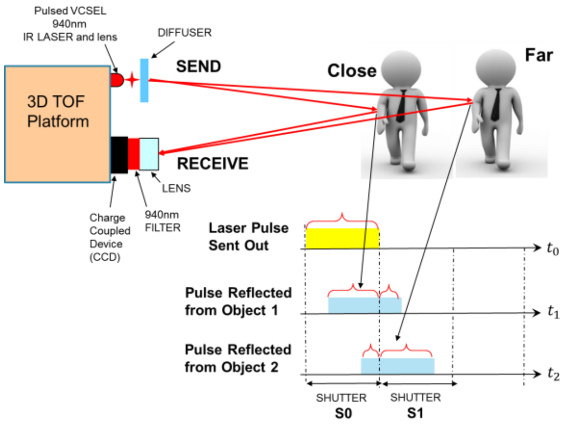
About me

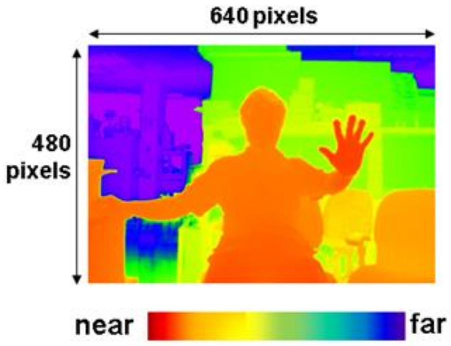
- Embedded Software Engineer at Analog Devices since 2019
- Member of ADI Systems Development Group (SDG)
- Developer of drivers for ADI parts:
  - Audio codecs (ALSA)
  - ToF sensor driver (V4L2)
  - HDMI Video transmitters (DRM)
  - High speed converters (IIO)
- Focus on open source products
- Previous experience in automotive embedded software

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Time Of Flight

- 3D Time of Flight is an industry term used to describe a type of scanner less (aka 'Flash') LIDAR (Light Detection and Ranging) used for depth sensing
- Used for short ranges typically < 10m from the source
- How Does It Work?
  - It uses light, continuous or pulsed to illuminate objects within a field of view
  - Light hits the objects and is reflected onto a sensor
  - The time it takes for the light to reflect off the objects and return to the source is measured
  - Using the known value for the speed of light the distance of the object from the source is determined
  - This information can then be used to create a depth or distance map of the scene in 3D





Time of Flight (ToF) calculation

$$distance = \frac{(speed\ of\ light)(Time)}{2}$$

where: speed of light = 3.0 x 10^8 m/s

Example:

t= 20ns

$$distance = \frac{(3.0\ \text{x}108\text{m/s})*(20\text{ns})}{2} = 3.0 \text{ meters}$$

- **# of Photons = Charge**
- **Ratio of Charge in S0 and S1**

$$\frac{S0}{S0+S1}$$

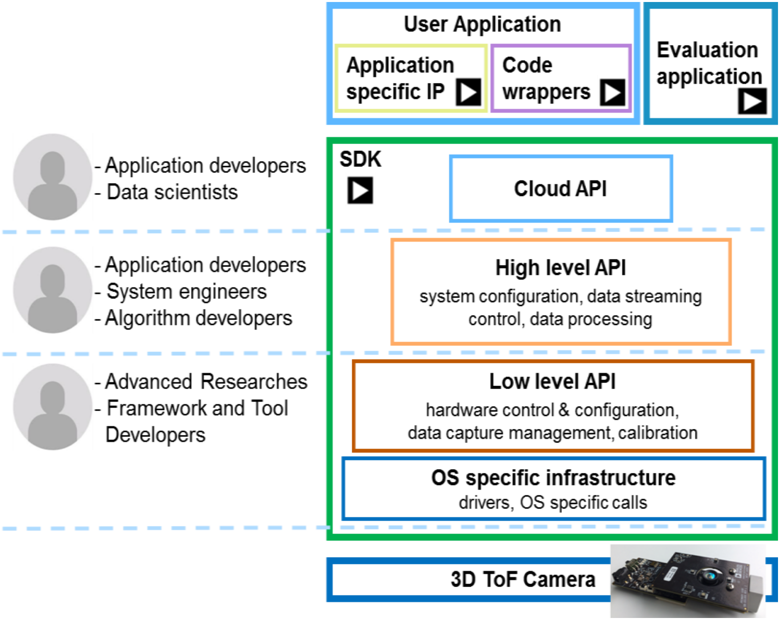
used to calculate Distance

3D ToF Development Kit

- 3D ToF Development Platform AD-96TOF1-EBZ
- Based on ADI’s ADDI9036 ToF processor
- Output: Depth & IR images @ 640 x 480 pixels (VGA), 30fps
- Distance sensing range: 20cm up to 6m
- Measurement accuracy: typical < 2% of the measured distance
- Connectivity: 96Boards compatible connector, 15 pin FPC connector (PI, Nvidia)
- Supported platforms:
  - 96Boards: DragonBoard410c, Thor96
  - Raspberry Pi 3 & 4
  - Nvidia: Jetson Nano, Xavier AGX, Xavier NX

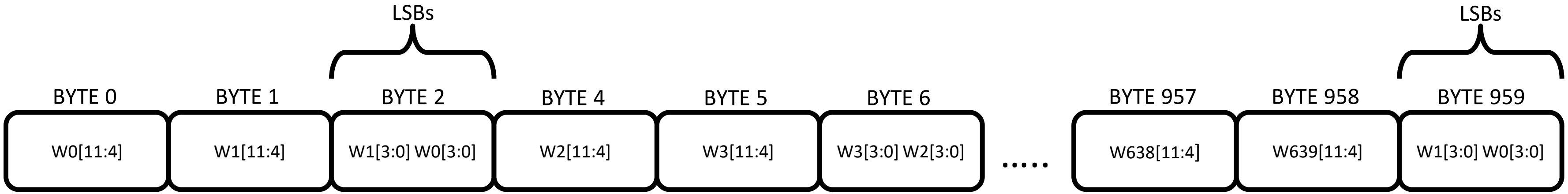
User Space SDK

- Open source SDK
- Runs both on PCs and embedded systems
- Support for multiple OSs
  - Linux, Windows, (Mac OS)
- Multiple connectivity options
  - MIPI, USB, Ethernet
- Wrappers for popular software frameworks & languages
  - C++, OpenCV, Python, Open3D
  - MATLAB, Robot OS
- Supported development platforms
  - Dragonboard 410c, Thor96
  - Raspberry Pi 3,4
  - Nvidia Jetson Nano, Xavier AGX, Xavier NX
  - Rockchip RK3399pro



ADI ToF special considerations (1/3)

- Output data type: MIPI RAW12 format LSB first



- This type requires unpacking
  – Performed by ISP on Raspberry Pi, Nvidia Jetson Nano, Nvidia Xavier AGX and NXP I.MX8
  – Implemented in SDK for Dragonboard, RK3399

ADI ToF special considerations (2/3)

- 3 possible MIPI output modes:

DEPTH only

FS VC = DEPTH	DEPTH DATA LINE = 1	DEPTH DATA LINE = 2	DEPTH DATA LINE = 3	...	DEPTH DATA LINE = 480	FE VC = DEPTH

IR only

FS VC = IR	IR DATA LINE = 1	IR DATA LINE = 2	IR DATA LINE = 3	...	IR DATA LINE = 480	FE VC = IR

DEPTH & IR

FS VC = DEPTH	FS VC = IR	DEPTH DATA LINE = 1	IR DATA LINE = 1	......	DEPTH DATA LINE = 480	IR DATA LINE = 480	FE VC = DEPTH	FE VC = IR

ADI ToF special considerations (3/3)

- Multiple operating ranges:
  - near (25 – 80 cm)
  - medium (30 cm – 3 m)
  - far (3 – 6 m)
- To each operating range correspond its own program instructions block (Firmware) + module specific data (Calibration)
- ADDI9036 chip has only volatile memory
- Calibration data is module specific written in production phase
- Some camera module implementations store FW + Calibration in EEPROM, others only Calibration
- Calibration data is stored in EEPROM using IEE754 floating point format

Linux Kernel Driver – implementation details (1/6)

- Sensor driver integrated in V4L2 Framework
- Located in /drivers/media/i2c/addi9036.c
- OV5640 Camera Driver as starting point
- Register v4l2-subdev with a source pad
- Expose standard & custom IOCTLs
- 2 major versions until final form

```
pi@raspberrypi:~ $ media-ctl -d /dev/media1 -p
Media controller API version 4.19.86

Media device information
------------------------
driver          unicam
model           unicam
serial
bus info        platform:3f801000.csi
hw revision     0x1
driver version  4.19.86

Device topology
- entity 1: addi9036 0-0064 (1 pad, 1 link)
            type V4L2 subdev subtype Sensor flags 0
            device node name /dev/v4l-subdev0
        pad0: Source
                [fmt:SBGGR12_1X12/640x960 field:none colorspace:srgb
                 crop:(0,0)/640x960]
                -> "unicam":0 [ENABLED,IMMUTABLE]

- entity 3: unicam (1 pad, 1 link)
            type Node subtype V4L flags 1
            device node name /dev/video0
        pad0: Unknown
                <- "addi9036 0-0064":0 [ENABLED,IMMUTABLE]
```

THE LINUX FOUNDATION

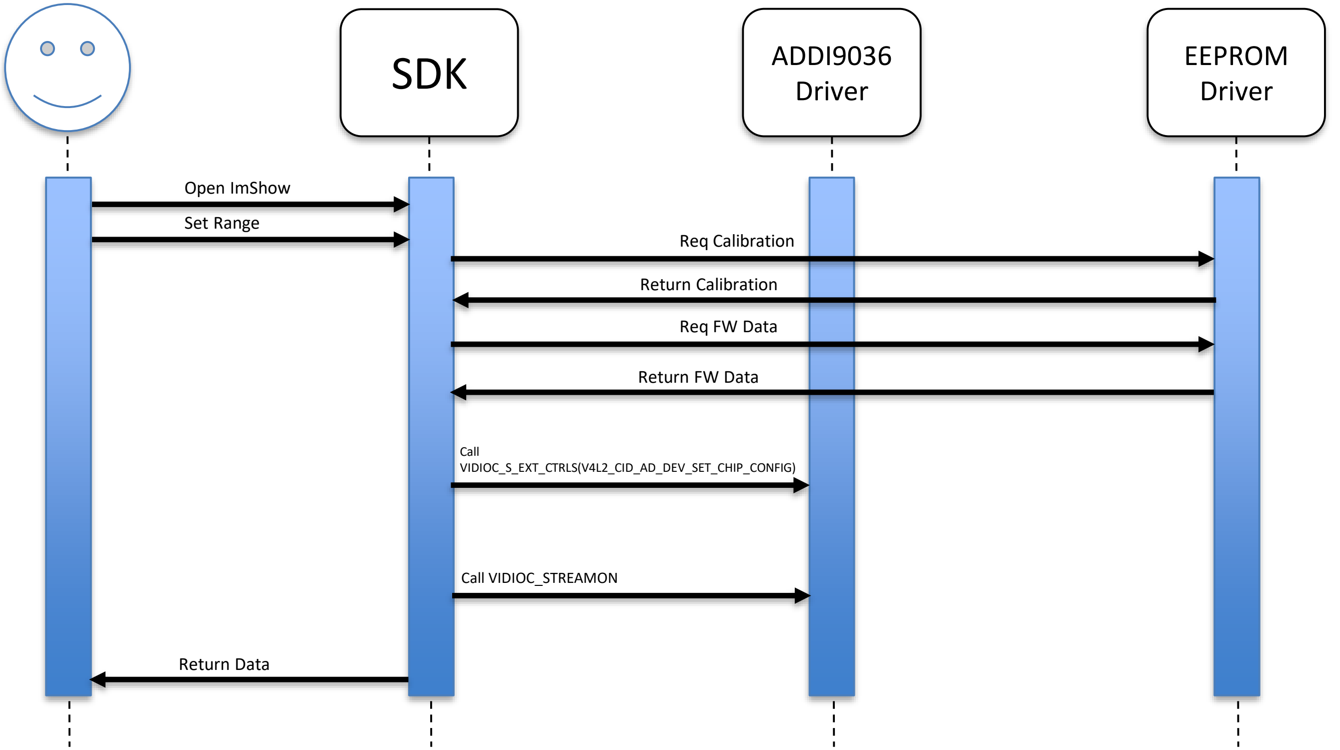
Linux Kernel Driver – implementation details (2/6)

- Initial version – handy for Camera/SDK development
- No clock handling (controlled by loading FW)
- Relied on old v4l2_subdev_core_ops->s_power()
- Custom controls for FW & calibration loading from Userspace

```
static const struct v4l2_ctrl_config addi9036_ctrl_chip_config = {
        .ops            = &addi9036_ctrl_ops,
        .id             = V4L2_CID_AD_DEV_SET_CHIP_CONFIG_QUERY,
        .name           = "chip_config",
        .type           = V4L2_CTRL_TYPE_U16,
        .def            = 0xFF,
        .min            = 0x00,
        .max            = 0xFFFF,
        .step           = 1,
        .dims           = { 2048 },
        .elem_size      = 2
};
```

- This was not acceptable for upstreaming

Linux Kernel Driver – implementation details (3/6)

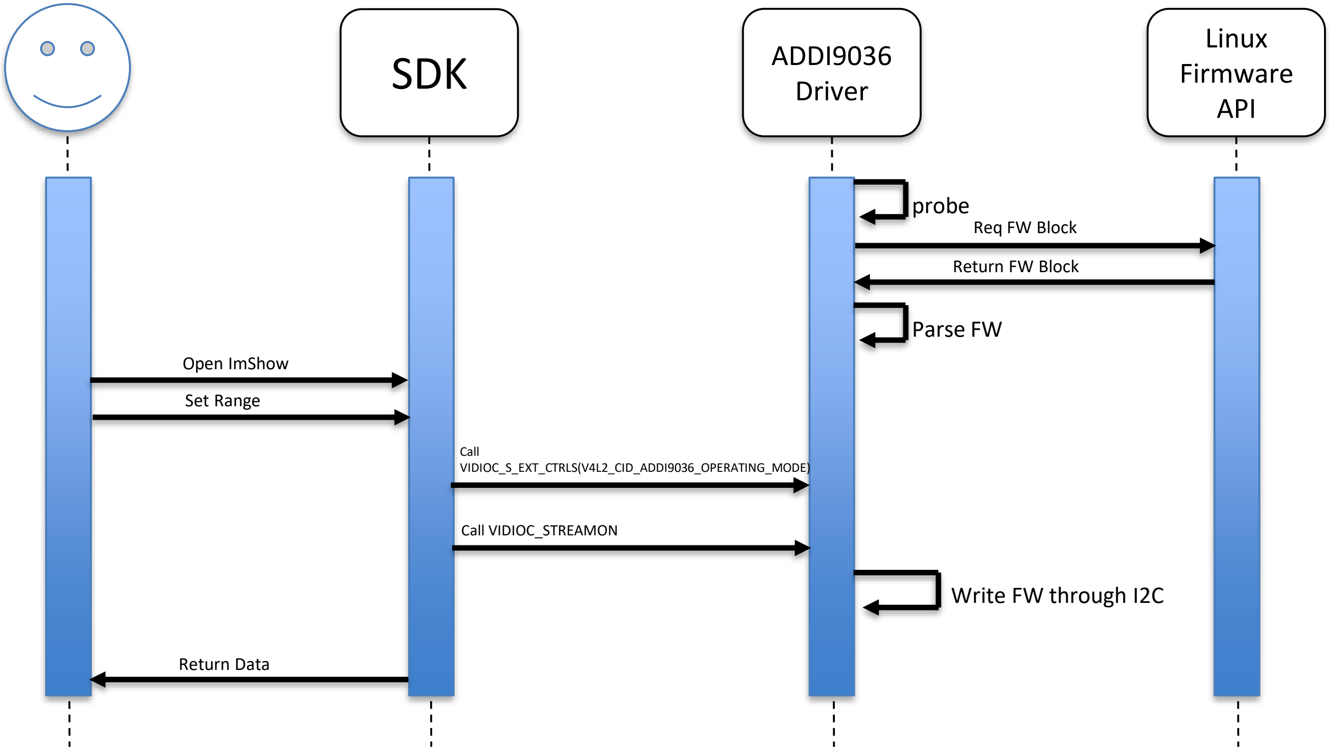


Linux Kernel Driver – implementation details (4/6)

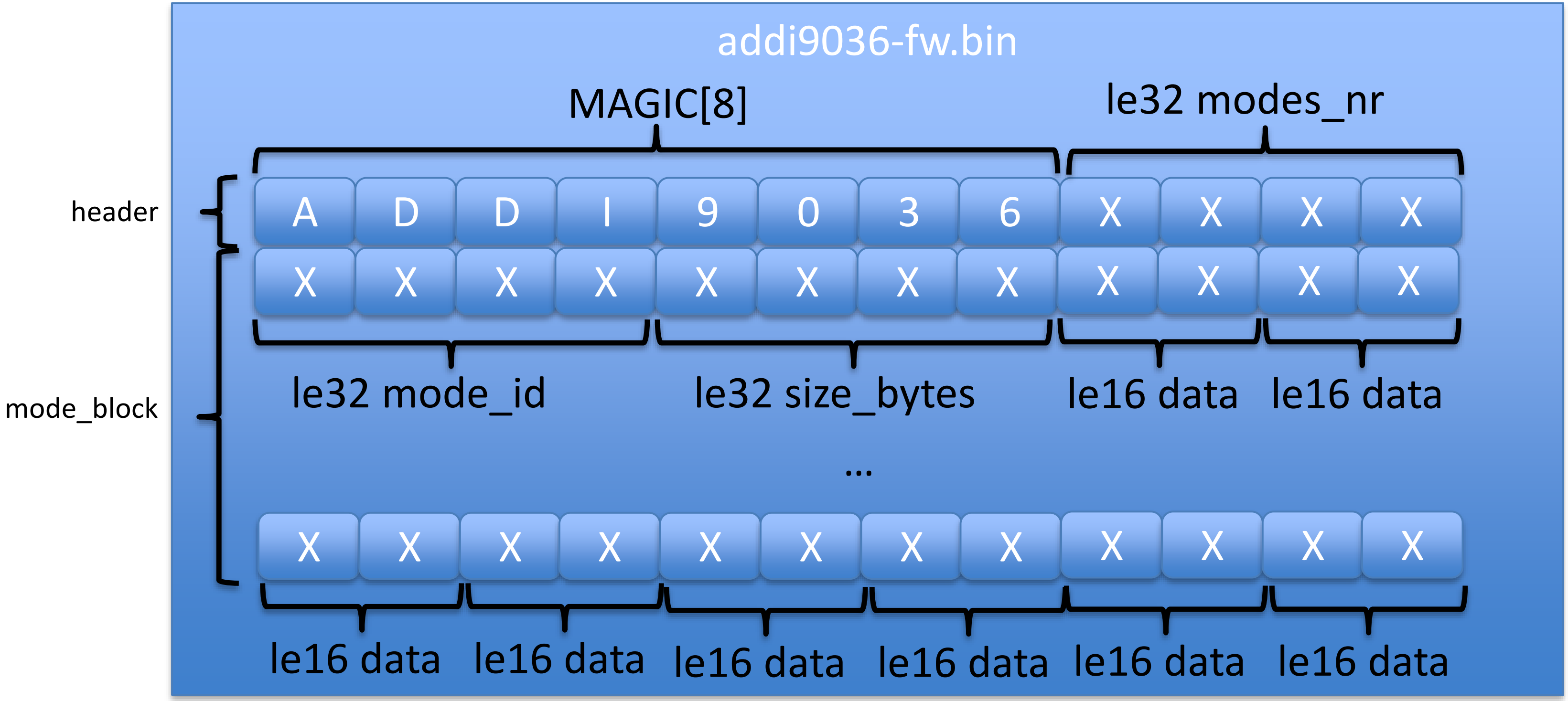
- Version two of driver moved FW loading from SDK to driver
- Added handling of optional reset GPIO using GPIO consumer framework
- Implement runtime PM for handling sensor power state
- Get FW and calibration from rootfs using Linux Firmware Framework
- Implement custom Integer type IOCTL for selecting operating range
- Chip is programmed at stream_on with corresponding FW based on selected operating range

THE LINUX FOUNDATION

Linux Kernel Driver – implementation details (5/6)

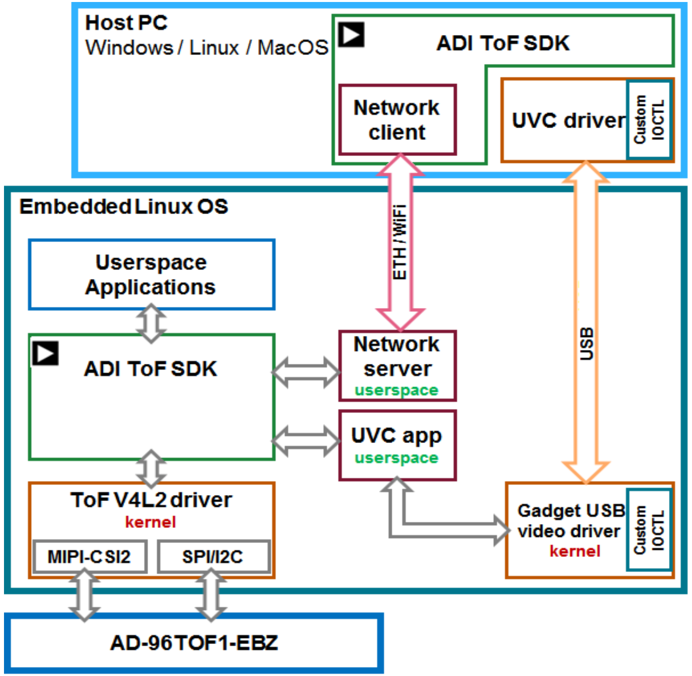


Linux Kernel Driver – implementation details (6/6)



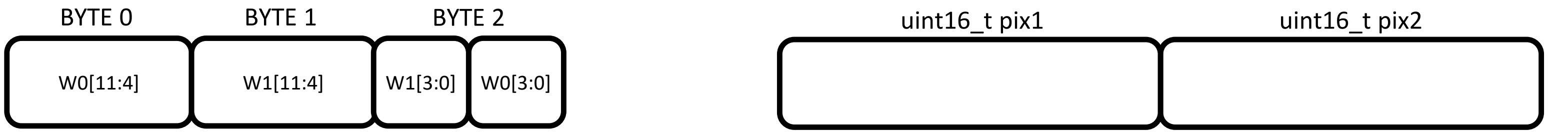
ADI ToF SDK – implementation details (1/4)

- SDK changed in parallel with the two main versions of the driver
- Scripts for easy compilation and dependencies installation
- Raw data processing (unpacking, reordering, shifting)
- Access and acquire data from temperature sensors installed on camera module
- Examples for frame acquisition, remote frame acquisition and display of data using OpenCV
- Some advanced image processing examples like detection of hand signs to play Rock, Paper and Scissors
- Tools for EEPROM read / write and performing calibration



ADI ToF SDK – implementation details (2/4)

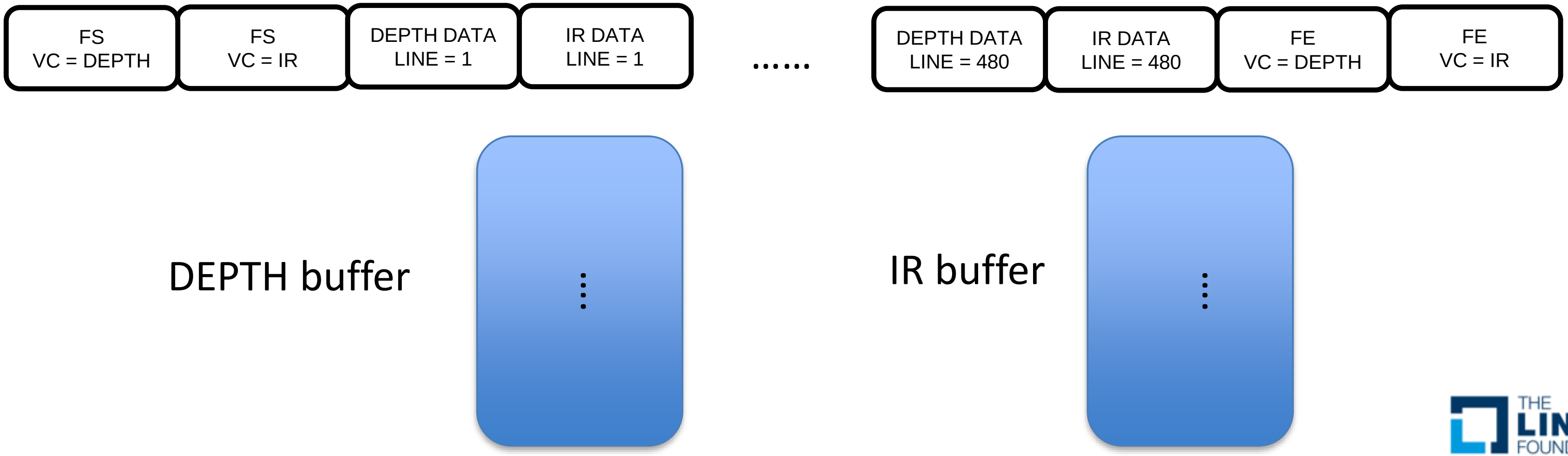
- Unpacking required for Dragonboard and is implemented in SDK
- The frame is read from the device as an array of uint8_t
- Every 3 uint8_t can produce 2 uint16_t that have only 12 bits in use.



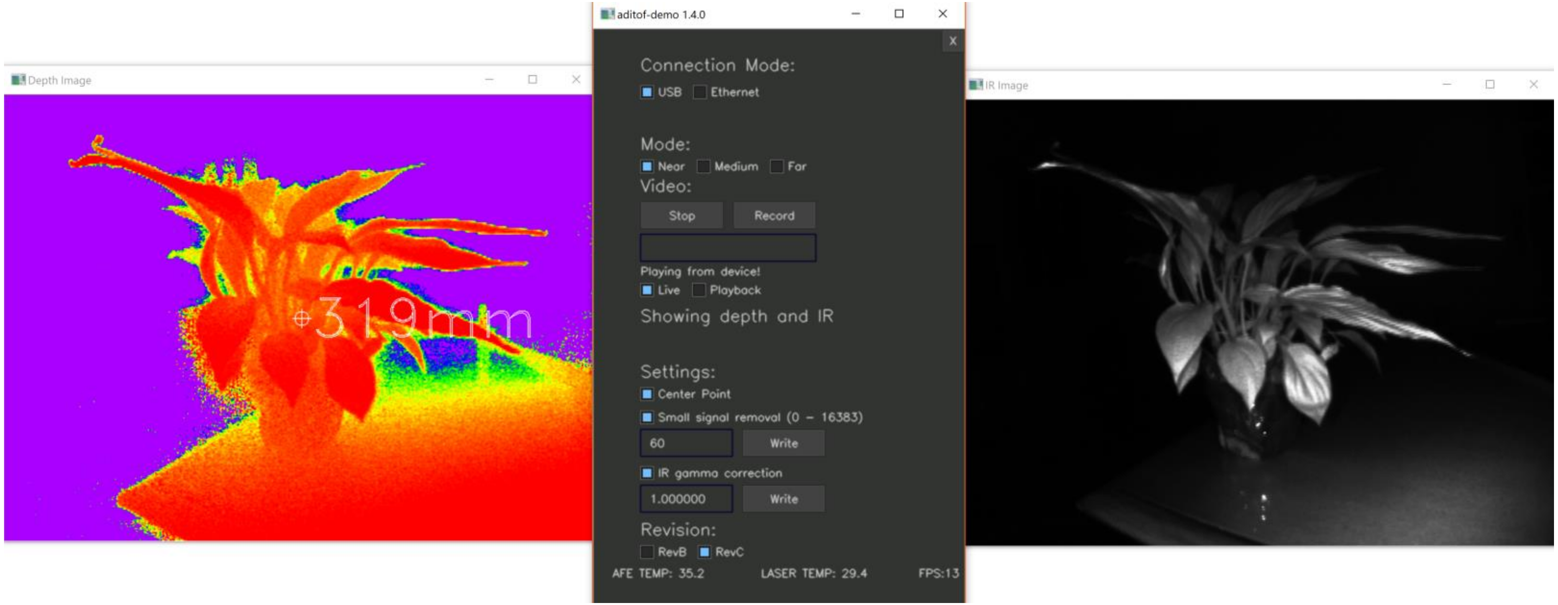
- Performed for both DEPTH and IR frames on each sample
- Implemented with loop unroll and making use of ARM NEON has no major load impact on CPU.

ADI ToF SDK – implementation details (3/4)

- For platforms where IR and DEPTH frames come concatenated on same buffer a deinterleaving operation of data is required.
- Makes sense only if both DEPTH and IR streams are enabled



ADI ToF SDK – implementation details (4/4)



- Select local/remote context
- Select Range
- Live run or playback of prerecorded data
- Compute distance of center point

Supported Platforms Peculiarities – Dragonboard 410C

- First platform supported
- Does not perform unpacking in ISP
- DEPTH and IR streams should be set on same VC. This means that will end up in same buffer and deinterleaving must be performed.
- Video pipeline should be configured using media-ctl
- The CAMSS video capture device does not take over and export the connected sensor IOCTLs

Supported Platforms Peculiarities – Raspberry PI

- Use of **bcm2835-unicam.c**
- Capable of performing RAW12 unpacking in ISP
- DEPTH and IR configured on same VC
- Video pipeline configuration is not explicitly required
- Unicam driver add controls from the subdevice

Supported Platforms Peculiarities - NXP I.MX8M Mini

- Use of **mx6s_capture.c**
- RAW12 format supported by HW but not implemented in driver
- Support unpacking in ISP
- I.MX8M Mini does not support Virtual Channels
- Discard frames if both DEPTH and IR streams are enabled
- v4l2_device_register_subdev_nodes was not called at complete notifier callback

Supported Platforms Peculiarities - Nvidia Jetson Nano

- Perform unpacking of RAW12 to uint16
- Does not support Virtual Channels
- Can be captured only IR or DEPTH but not both in the same time
- Video pipeline is configured automatically based on DT bindings
- Driver required adaptations for integration in NVIDIA Camera Common framework

Supported Platforms Peculiarities - Nvidia Xavier AGX

- Same driver as on Jetson can be used
- Support Virtual Channels
- An individual /dev/video device is created for each VC
- SDK should handle this specific case and open two video devices.
- Two instances of addi9036 driver are instantiated but one is dummy
- Subdev IOCTLs should be accessed through /dev/v4l-subdev

Supported Platforms Peculiarities – Rockchip RK3399pro

- Custom v4l driver stack
- Drivers located in separate folder
  - kernel/drivers/media/i2c/soc_camera/rockchip
- A customized driver was created to handle
  - specific calls for ioctl handling
  - setting the image format
  - initializing the driver
- Custom devicetree including additional information such as camera name, camera FoV, image size and orientation
- Does not perform unpacking in ISP

Thank you for your attention

- Questions? Comments?

- Bogdan Togorean – bogdan.togorean@analog.com

Embedded Linux Conference
Europe