EBV Elektronik
An Avnet Company

Technology. Passion. EBV.

Astrial

industrial SoM for ai@edge

Gianluca Filippini

December, 2024

ASTRIAL
SYSTEM ELECTRONICS for AI Innovation

an “ai journey”: March 2023 to October 2024

SYSTEM Electronics
a coesia brand

https://www.systemelectronics.com/

ai@EBV sub-10W vision systems

Decision factors for hw acceleration on ai: vision, high data throughput, low power, small form factor

smart cameras
quality inspection
people monitoring
object detection
object sorting/counting
License Plate Reading

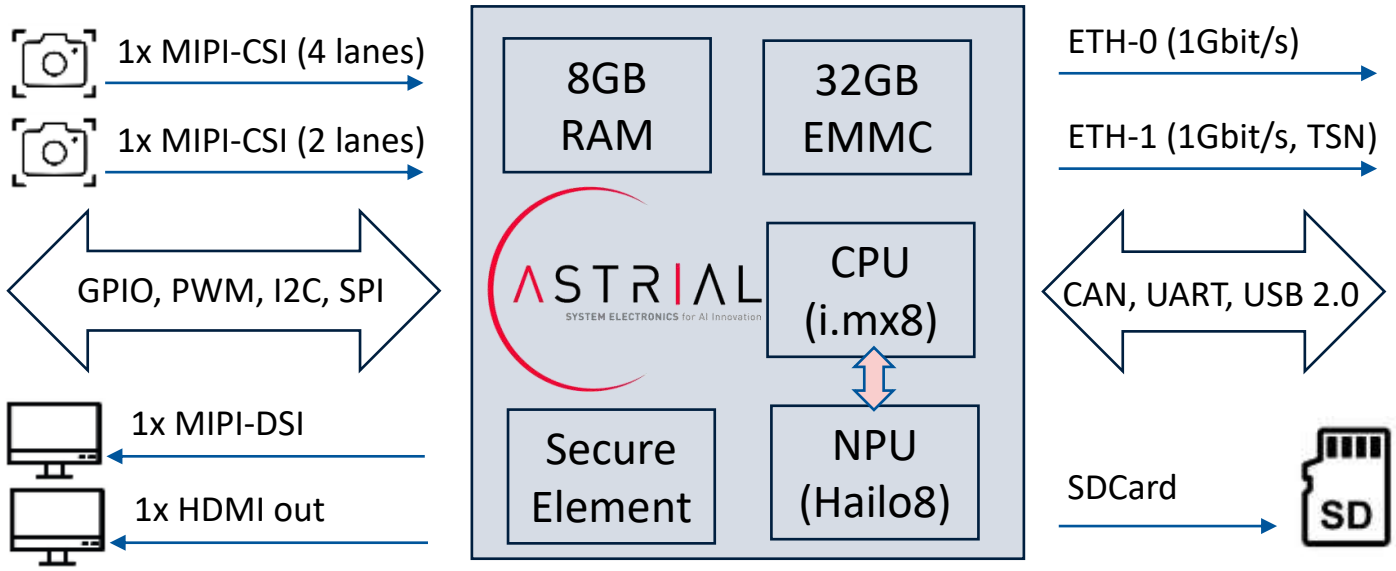
robotics
storage sorting
selective picking
warehouse automation

drones
search & rescue
traffic monitoring
parking control
agriculture
door delivery

industrial vehicles
blind spot monitoring
load inspection

compute module
… any CM4 product
is a fit for Astrial …

Astrial: software design first



Note: NO wireless connectivity, NO analog I/O
(must be added externally on the carrier board)

NXP

clone of NXP hw reference design im8m-plus-evk for best software compatibility and longevity

HAILO

aligned with official code from Hailo Driver, HailoRT, ModelZoo.

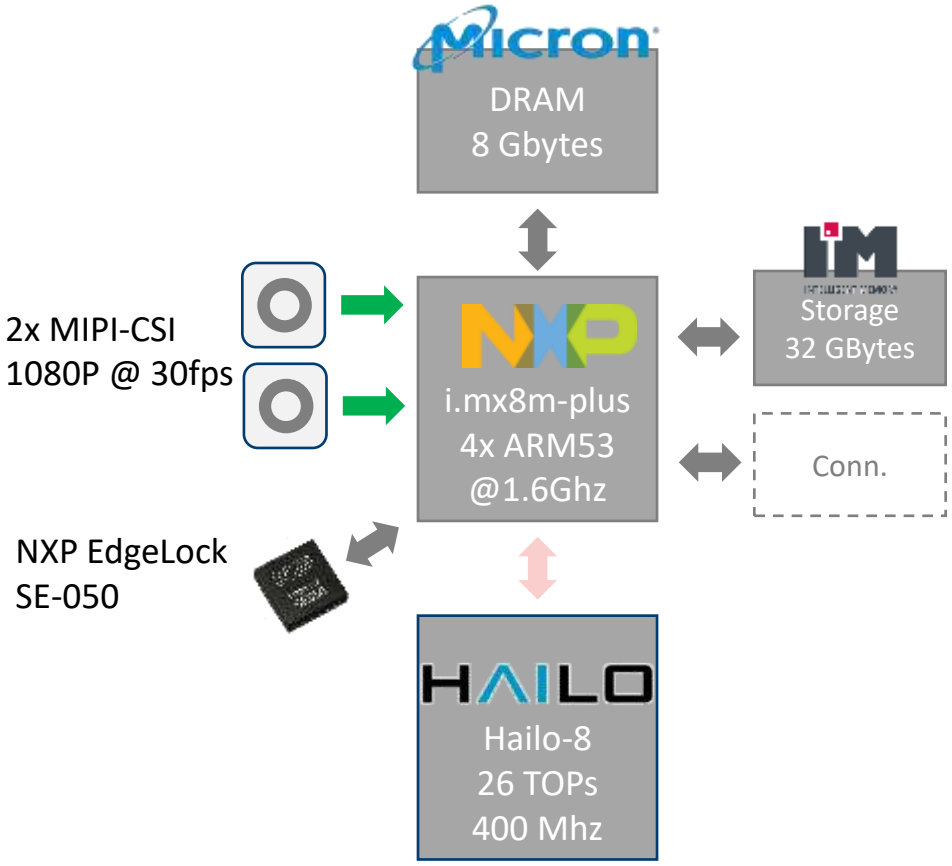
Yocto/Linux LTS and Security

Software releases aligned on YOCTO LongTermSupport (currently Kirkstone)
Extra Secure Element (NXP 050) for enhanced user-level software security.

Astrial: NXP i.mx8m-plus & Hailo-8

LOADED

Hardware



Software

NXP + yocto PROJECT

i.mx8

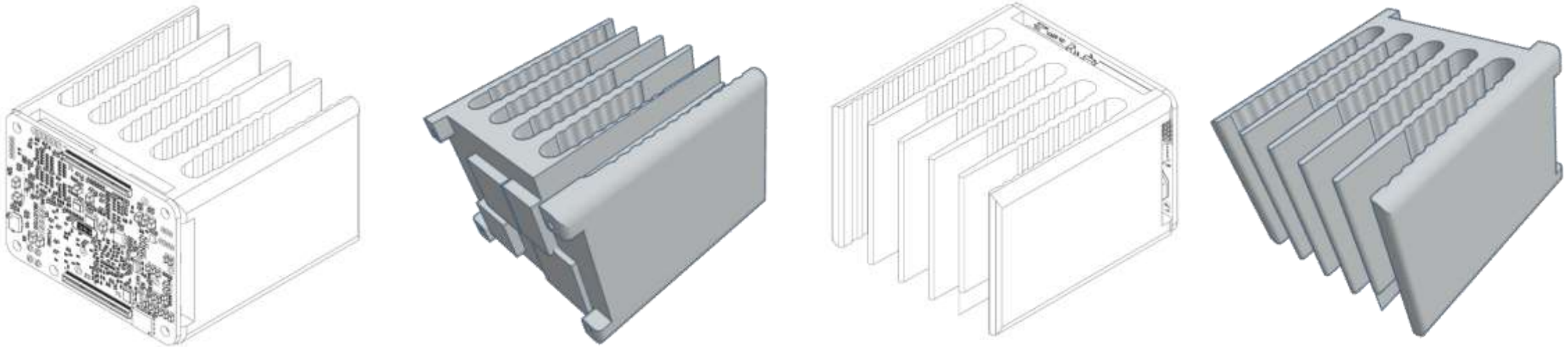
Processing	I.MX8 plus (4 x Cortex A53 + cortex M7 2,3 TOPS)
AI	HAILO-8 (26 TOPS @3W)
Memory	RAM 8 GB LBDDR4 / eMMC 32 GB
Video / HMI	Hardware H264/H265 encoder/decoder
	Dual ISP for camera
	GPU for HMI
Security	Enhanced IoT edgeLock SE050
Peripherals	Gigabit Ethernet PHY
	26 × GPIO HEADER
	1 × USB 2.0, 2 × UART, 3 × I2C, 2 × SPI, 2 × SDIO, 1 × PCM
	1 × PWM, 1 × HDMI, 1 × CANbus
	1 × 4-lane MIPI DSI, 1 × 2-lane MIPI CSI, 1 × 4-lane MIPI CSI
consumption	From 3W up to 10W
BSP	Linux YOCTO Kirkstone / LTS and GitHub support
mechanical	CM4 footprint (40 x 55 mm)

ASTRIAL
SYSTEM ELECTRONICS for AI Innovation

Astrial is Ind(a/u)strial

24/7 test in thermal chamber with 35C ambient temperature, fan-less (custom heatsink)

All components are -40 C / +85 C



ASTRIAL
SYSTEM ELECTRONICS for AI Innovation

RPi-CM4 compatible

differences and trade-offs between NXP and Broadcom chip

- Astrial does not expose 1x pcie lane (dedicated to Hailo8).
- MIPI-CSI supported cameras: only RPi camera v.2.0 and similar (sony imx219)
- MIPI-DSI: work in progress to enable Rpi display (not in current sw release)
- I2S audio I/O: not yet available on Astrial
- HDMI output: only one output available (due to NXP imx8 single HDMI output)

(details on the Astrial GPIO mapping available in the User Manual on Astrial website)

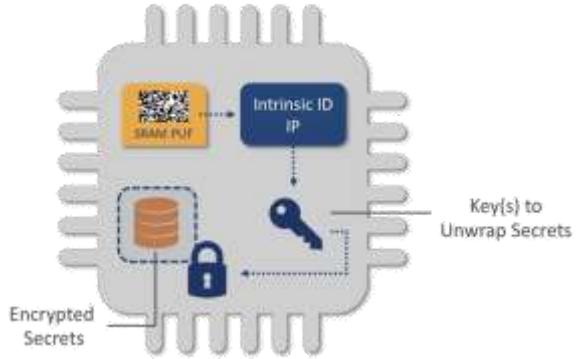
ASTRIAL
SYSTEM ELECTRONICS for AI Innovation

Software Security

NXP SE050 crypto chip onboard

Use case #1
encryption of sensitive data / software

Use case #2
Root of trust for 3RD party software



Software Security

Technology. Passion. EBV.

https://www.cyberresilienceact.eu/
https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act
https://www.linuxfoundation.org/blog/understanding-the-cyber-resilience-act

https://yogosha.com/blog/cra-cyber-resilience-act-guide/

ASTRIAL
SYSTEM ELECTRONICS for AI Innovation

Ecosystem

because hardware and software are not enough

Tutorials, hands-on and videos

ASTRIAL
hackster.io
YouTube
in
HAILO | Community

Open-Source code

GitHub

Technical support

EBV
SYSTEM Electronics
a coesia brand
HAILO
NXP

https://community.hailo.ai/

https://github.com/System-Electronics/astrial-howto

EBV events 2024

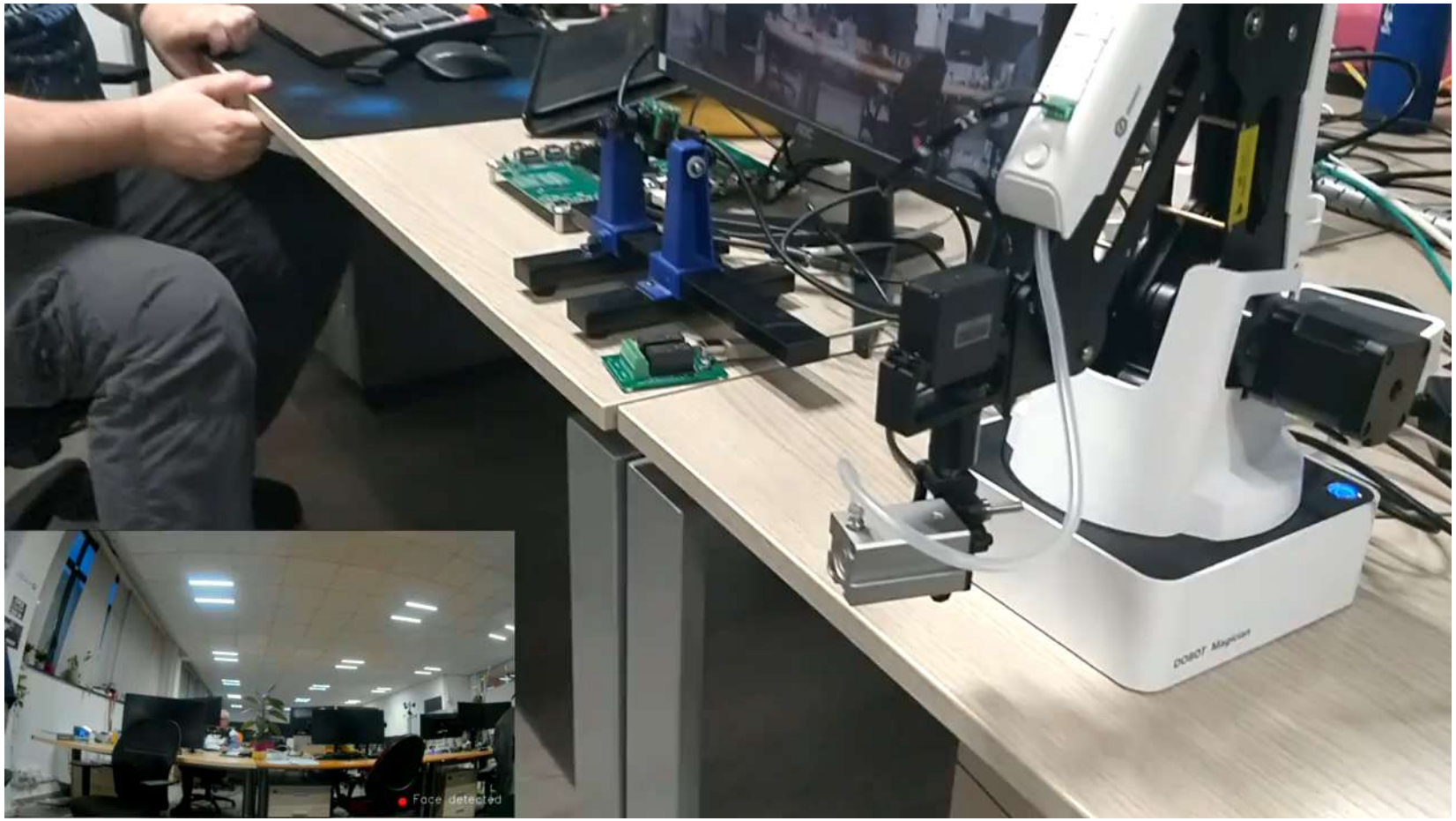
Technology. Passion. EBV.

embeddedworld
Exhibition&Conference

System Electronics
1,239 followers
5mo •

KALPA

embedded world Exhibition&Conference got off to a great start. And today we are ready for day two.



EBV events 2024

EBV Elektronik | An Avnet Company |

Technology. Passion. EBV.

sps Fiere di Parma ITALIA smart production solutions

KALPA

Smart Refrigerator

sps Fiere di Parma ITALIA smart production solutions

HIPERT

6D Pose Estimation Robot

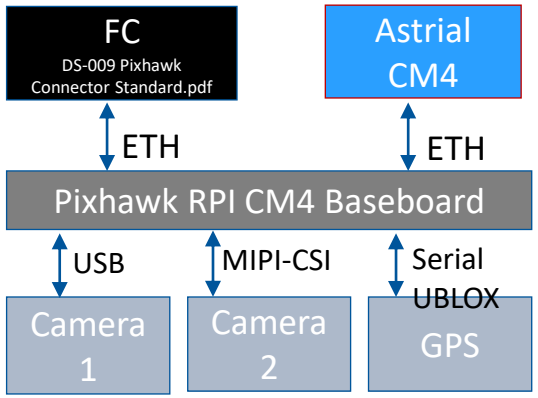
EBV events 2024

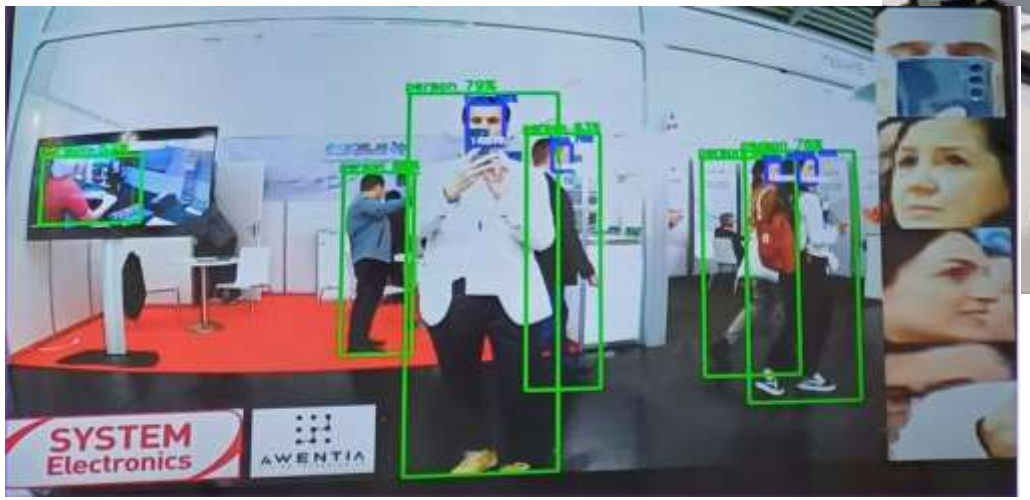
EBV Elektronik | An Avnet Company |

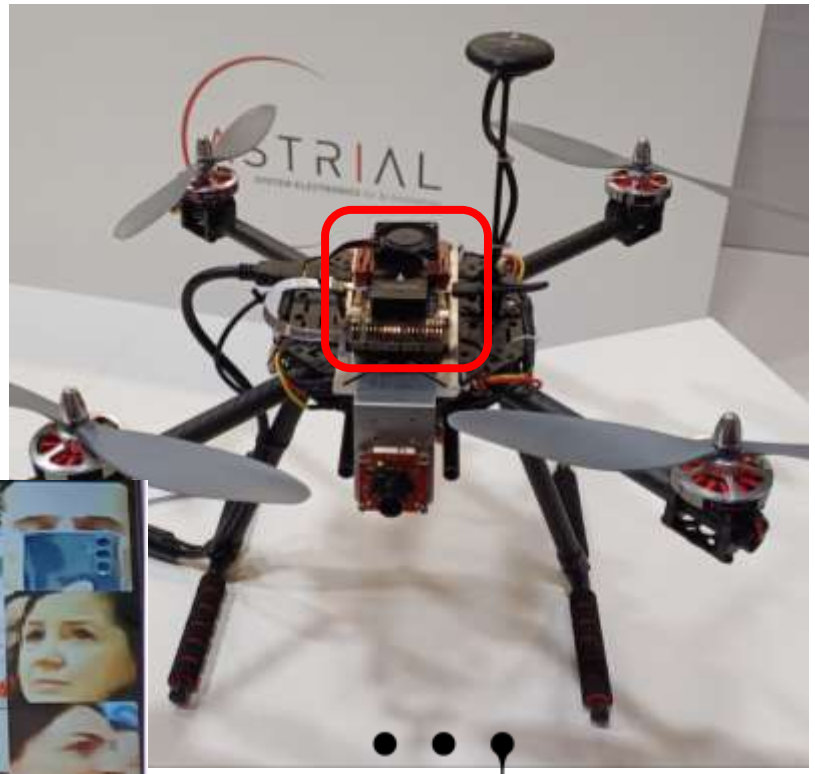
Technology. Passion. EBV.

electronica

MESSE MÜNCHEN







AWENTIA VISION · TECHNOLOGIES

People detection, Face cropping and snapshot of the last 4 detected faces

Industrial use case: blind spot monitoring

KALPA

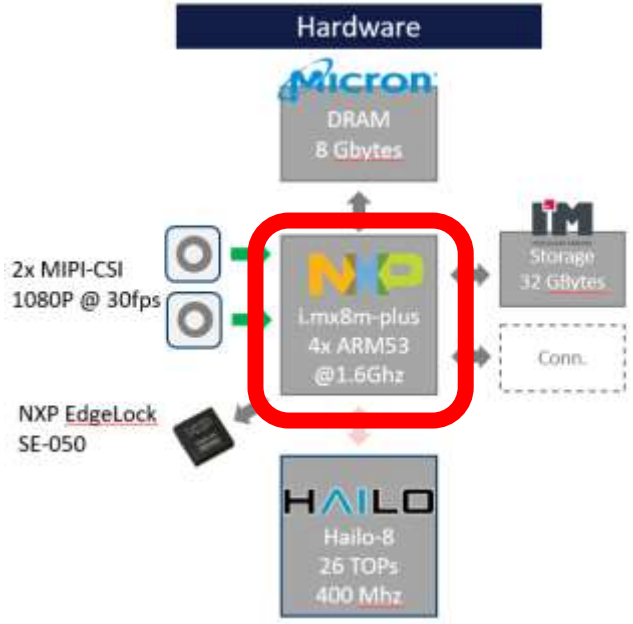
Industrial use case: blind spot monitoring

KALPA

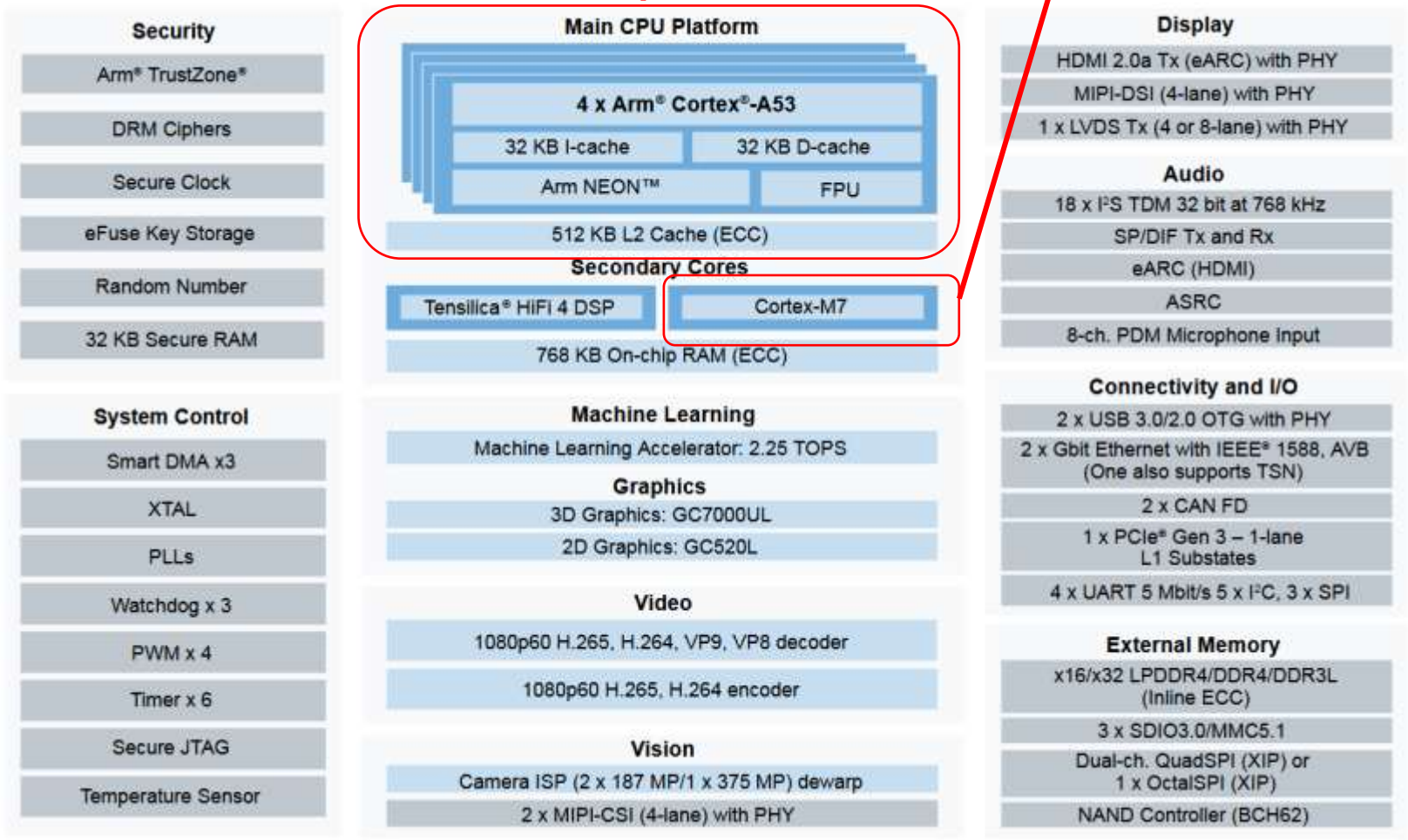
Industrial use case: crane payload safety

KALPA

CPU: i.mx8m-plus



https://www.yoctoproject.org/

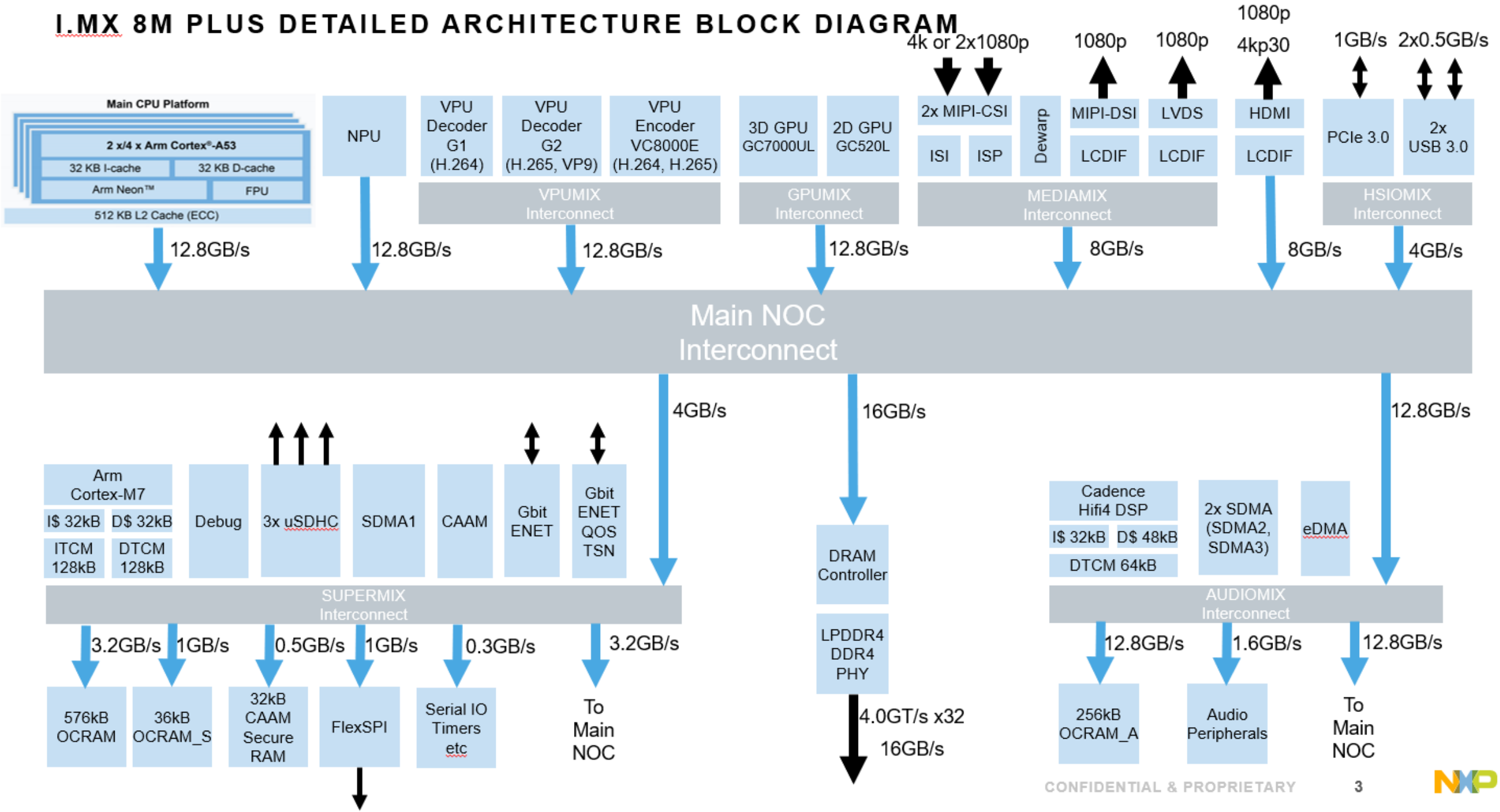


NXP Linux BSP via YOCTO build:

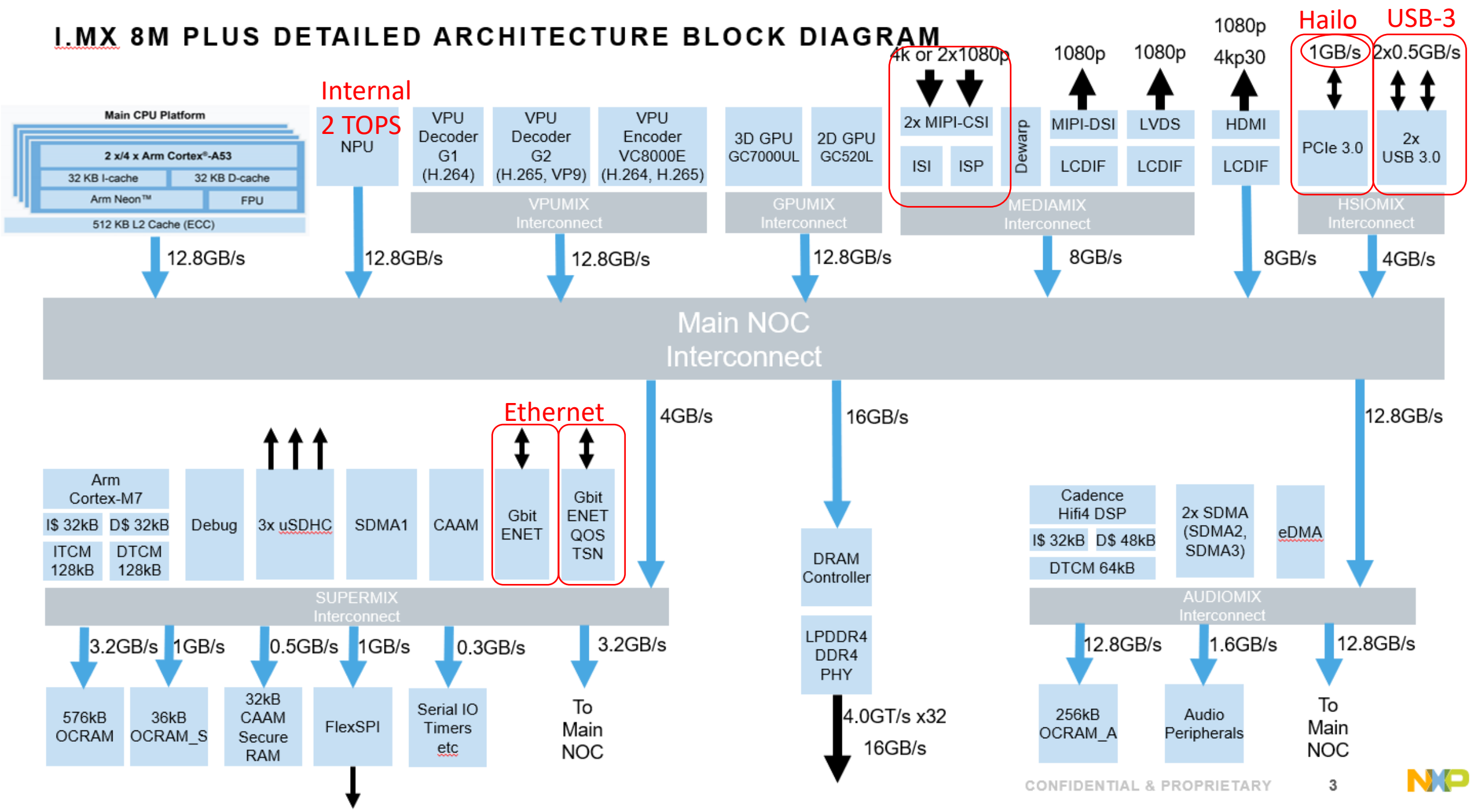
https://www.nxp.com/design/design-center/software/embedded-software/i-mx-software/embedded-linux-for-i-mx-applications-processors:IMXLINUX

- Quad Arm Cortex®-A53 processor up to 1.6/1.8 Ghz (with a Neural Processing Unit 2.3 TOPS)
- Dual image signal processors (ISP) and two camera inputs for an effective advanced vision system.
- Multimedia capabilities including video encode and decode (H.264/H.265)
- 3D/2D graphic acceleration for HMI functionalities.
- Real-time control with Cortex-M7
- Dual Gigabit Ethernet with Time Sensitive Networking (TSN).
- High industrial reliability with DRAM inline ECC

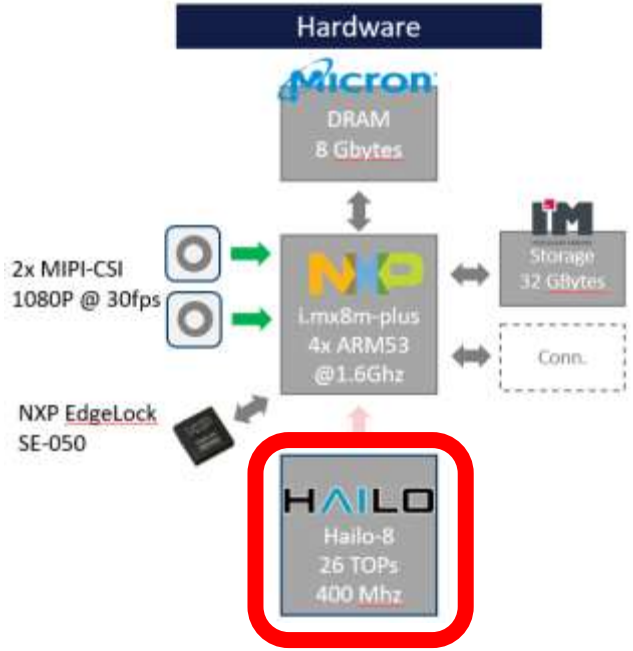
I.MX 8M PLUS DETAILED ARCHITECTURE BLOCK DIAGRAM



I.MX 8M PLUS DETAILED ARCHITECTURE BLOCK DIAGRAM



CPU: Hailo H8


Hardware
Micron
DRAM
8 Gbytes
2x MIPI-CSI
1080P @ 30fps
NXP
i.mx8m-plus
4x ARM53
@1.6Ghz
iM
Storage
32 Gbytes
Conn.
NXP EdgeLock
SE-050
HAILO
Hailo-8
26 TOPs
400 Mhz

Hailo-8™ : co-processor | accelerator

High Performance
26 TOPS

Automotive Grade
ASIL-B (D)
AEC-Q100 Grade-2

Flexible
Fully Programmable
Comprehensive SDK

High Efficiency
Typical Power
Consumption: 2.5W

Single Chip Solution
No External DRAM
required

Configuration
- Standalone
- Co-processor

Hailo-8 Products (*available at EBV Elektronik*)

Hailo-8L Entry-Level AI Accelerator

CoB –
13 TOPS

M.2 Entry-Level AI Acceleration Modules

13 TOPS

Key B+M
2 lanes

Key A+E
2 lanes

Hailo-8 AI Accelerator

CoB –
26 TOPS

M.2 AI Acceleration Module

26 TOPS

Key M
4 lanes

Key B+M
2 lanes

Key A+E
2 lanes

Hailo-8R™ mPCIe AI Acceleration Module

13 TOPS

Hailo8-R (reduced) 200Mhz Clock to fit power budget on mpcie bus.

Hailo-8 Century High Performance PCIe Cards

Small Form Factor
52-104 TOPS

0802, 0803, 0804

Wide Host Compatibility
104-208 TOPS

0804S, 0805S, 0806S, 0807S, 0808S

multi-chip configuration delivering up to 104 TOPS, typical power consumption of 35W

https://lannerinc.com/products/edge-ai-appliance/deep-learning-accelerators/falcon-h8

Neural Networks - Properties

Control

- Flexibility during compile time
- Fully deterministic in runtime

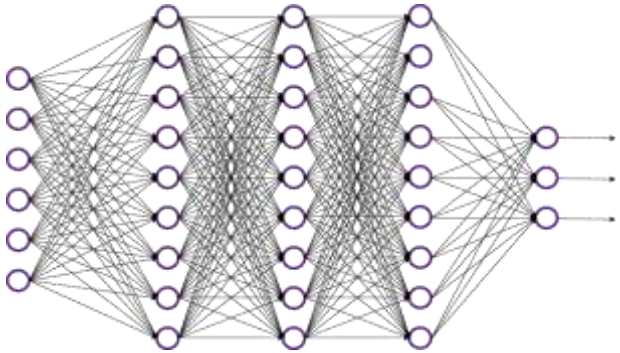
Memory

- Parameters and partial sums are localized
- Layer outputs move around (but not too far)

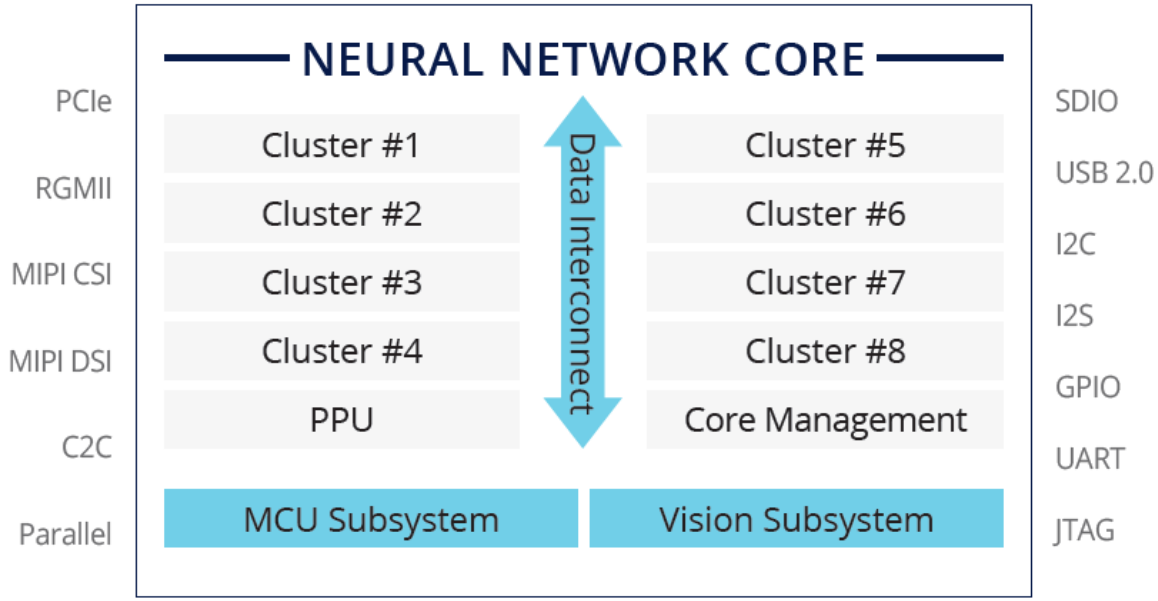
Compute

- Recurring operations (MACs >> Activations)
- Mostly low precision

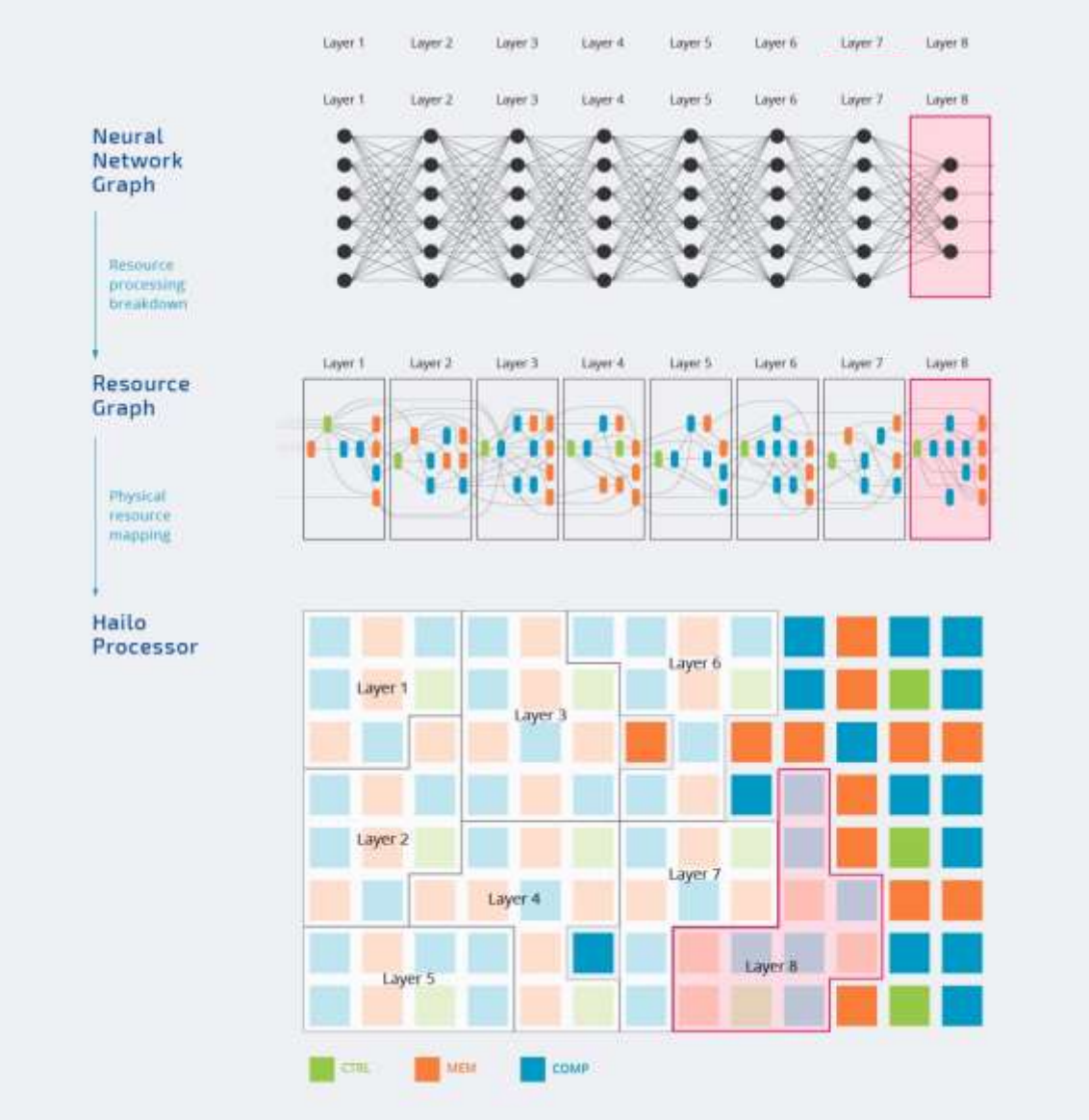
Interconnect



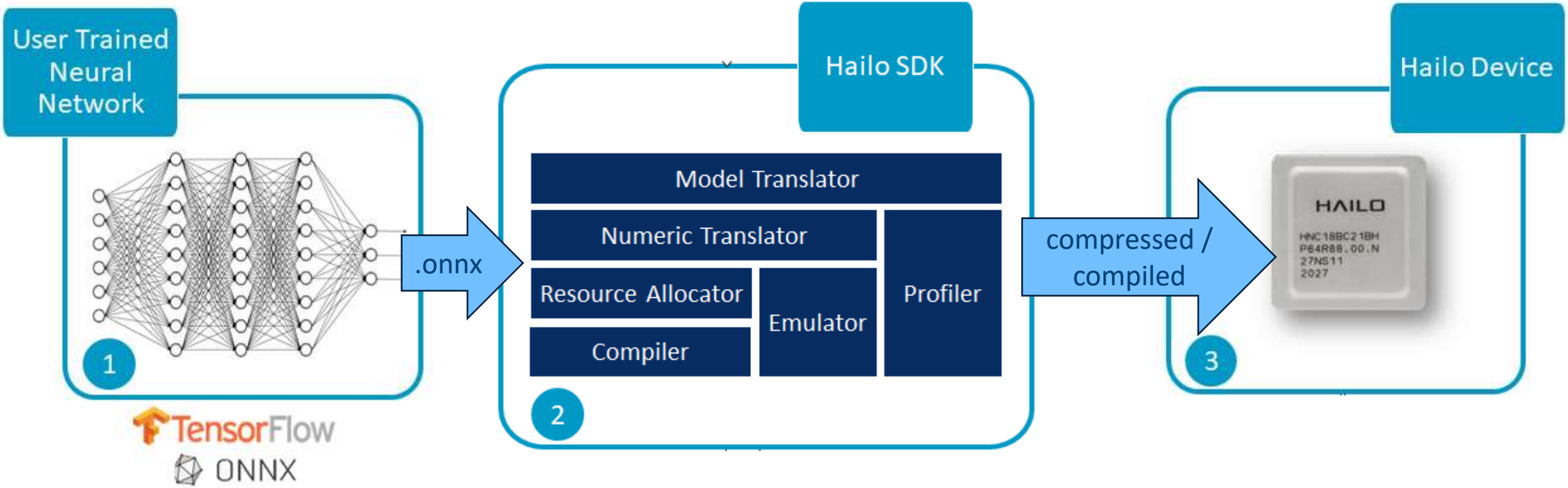
Hailo-8™



Neural Networks - Structure Defined Dataflow Architecture



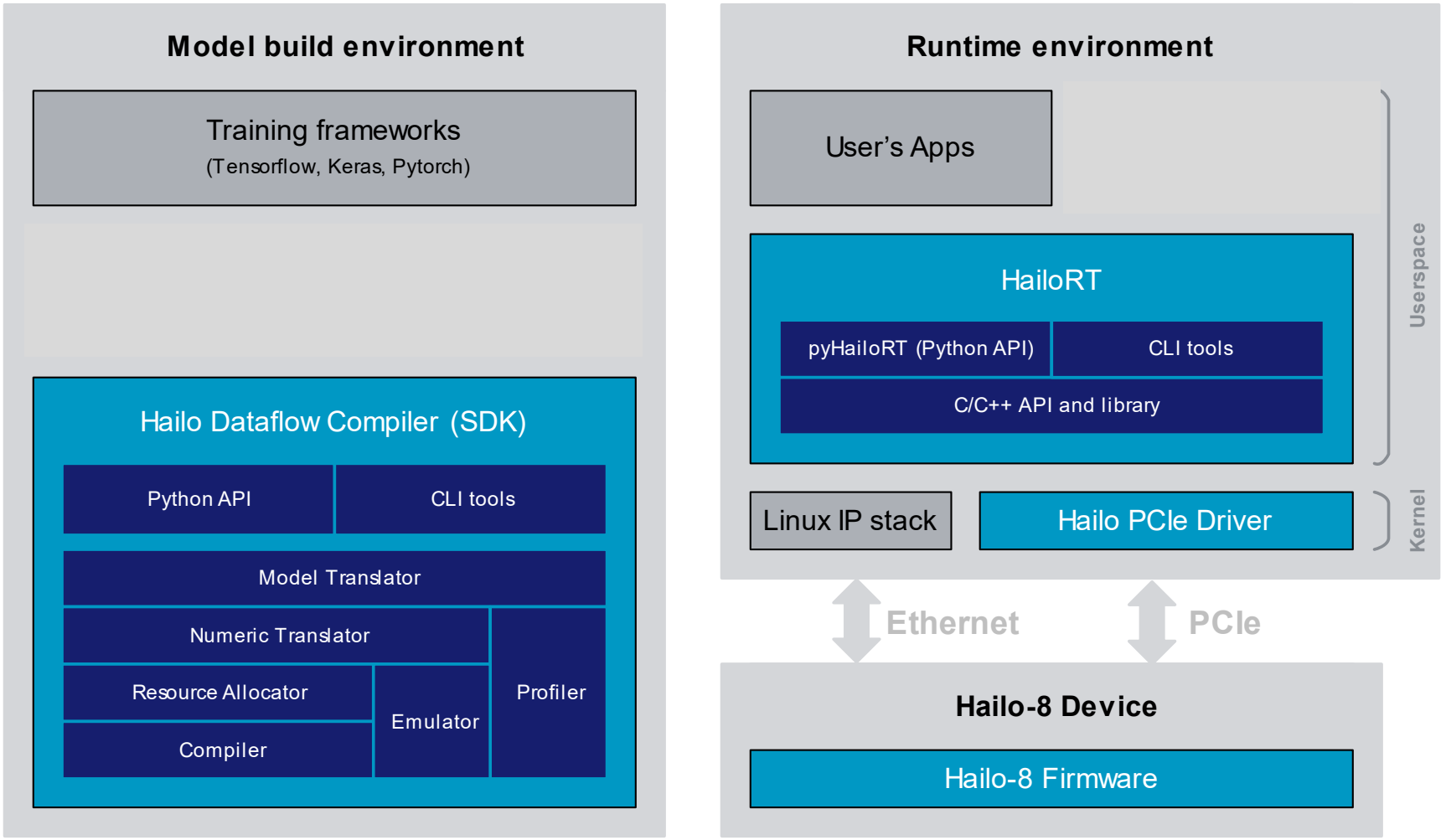
Hailo Dataflow Compiler (DFC)



IMPORTANT!

See DFC tutorials/notebooks with “jupyter notebook”

Hailo Software Toolchain and Developer Tools



Hailo software component

Other software component

IMPORTANT!

Read Documentation First (!)
- HailoSDK for DFC (*pdf)*
- HailoRT_Guide for Linux (*pdf)*
- Tappas User Guide (*github)*

https://hailo.ai/developer-zone/documentation/

https://github.com/hailo-ai/tappas

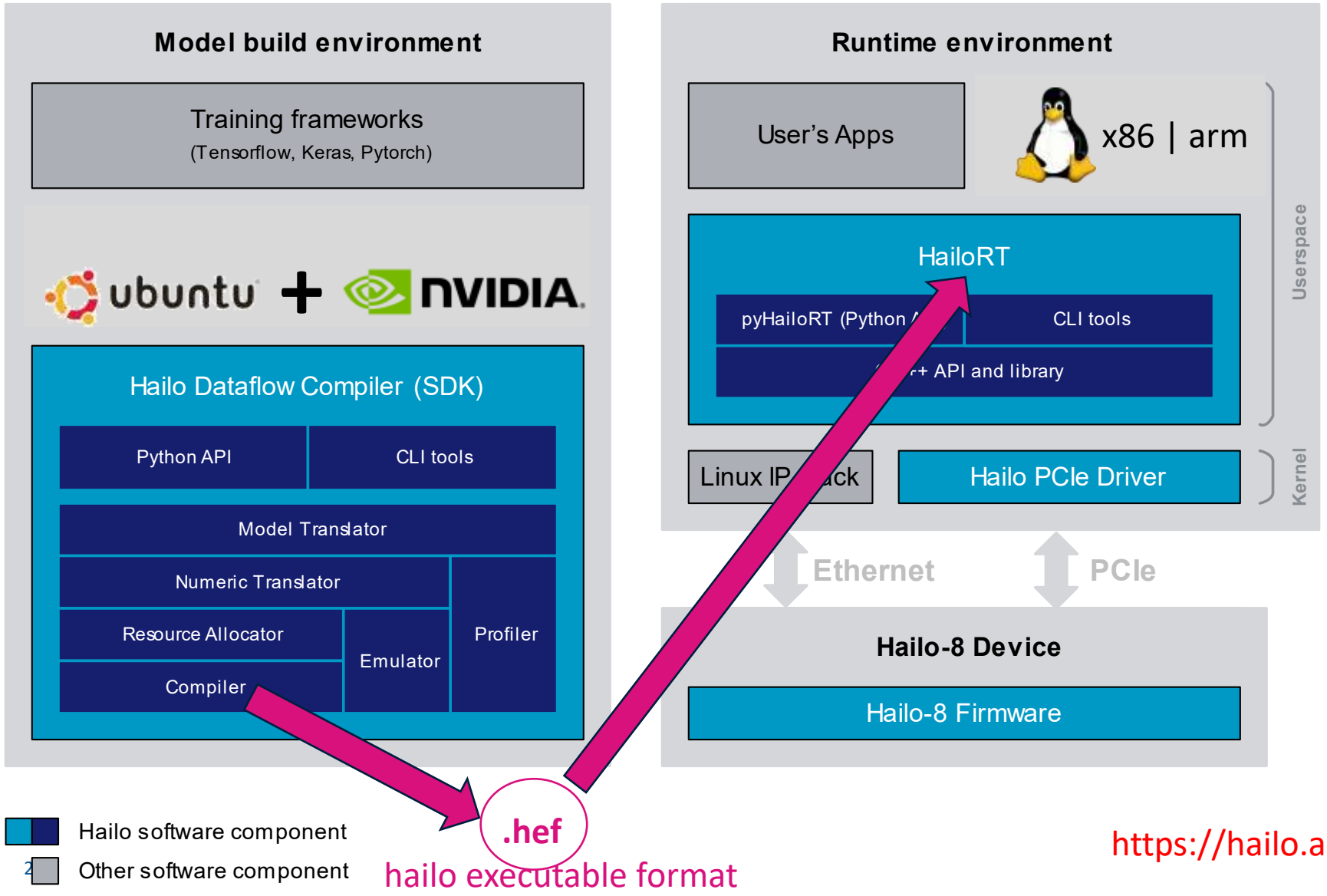
IMPORTANT!

Note: subscribe to the "DevZone" on Hailo.ai for SDK and Documents above

IMPORTANT!

https://hailo.ai/developer-zone/sw-downloads/

Hailo Software Toolchain and Developer Tools



IMPORTANT!

Read Documentation First (!)
- HailoSDK for DFC (*pdf)*
- HailoRT_Guide for Linux (*pdf)*
- Tappas User Guide (*github)*

https://hailo.ai/developer-zone/documentation/

https://github.com/hailo-ai/tappas

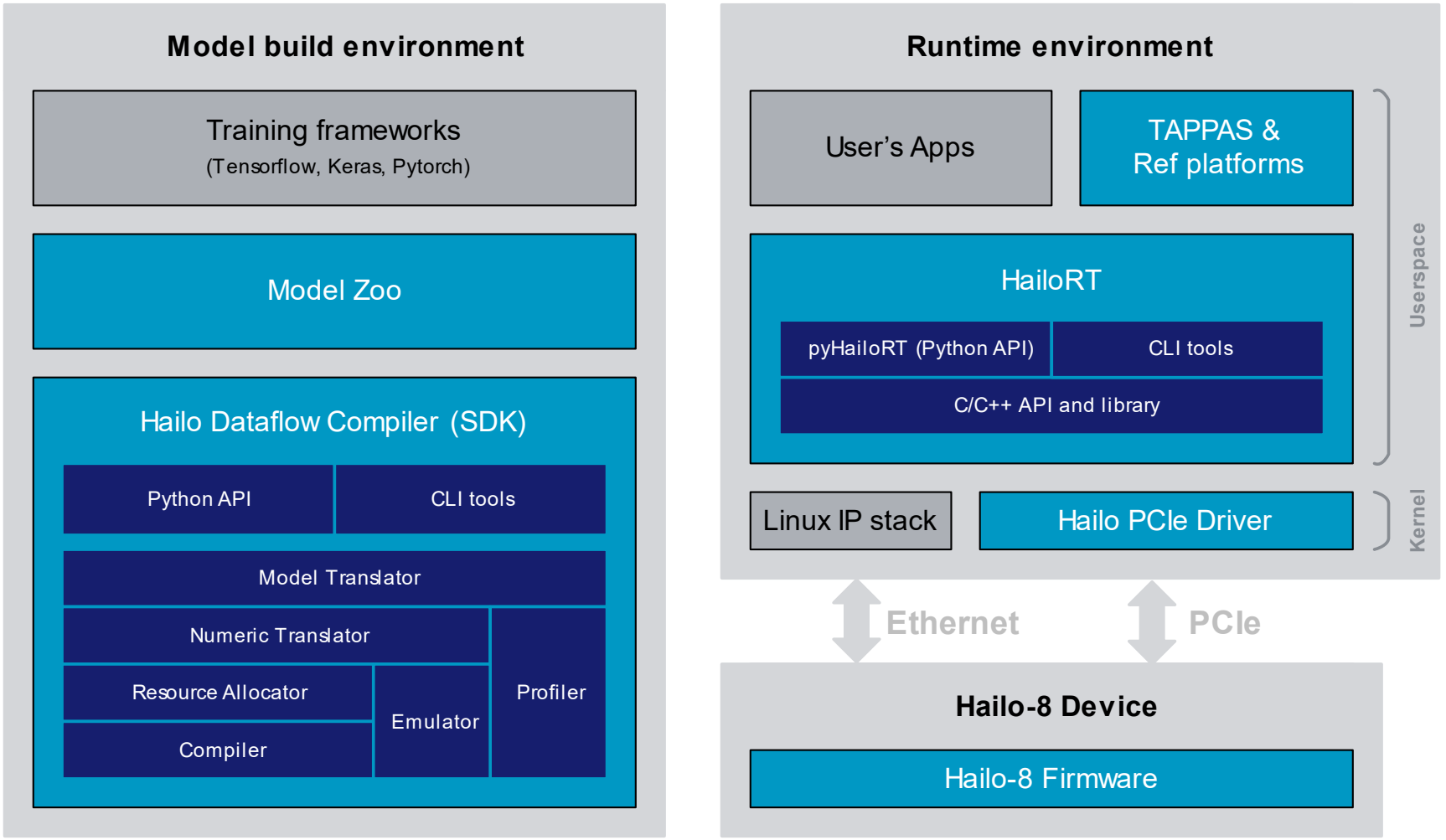
IMPORTANT!

Note: subscribe to the "DevZone" on Hailo.ai for SDK and Documents above

IMPORTANT!

https://hailo.ai/developer-zone/sw-downloads/

Hailo Software Toolchain and Developer Tools



IMPORTANT!

Read Documentation First (!)
- HailoSDK for DFC (*pdf)*
- HailoRT_Guide for Linux (*pdf)*
- Tappas User Guide (*github)*

https://hailo.ai/developer-zone/documentation/

https://github.com/hailo-ai/tappas

IMPORTANT!

Note: subscribe to the "DevZone" on Hailo.ai for SDK and Documents above

IMPORTANT!

https://hailo.ai/developer-zone/sw-downloads/

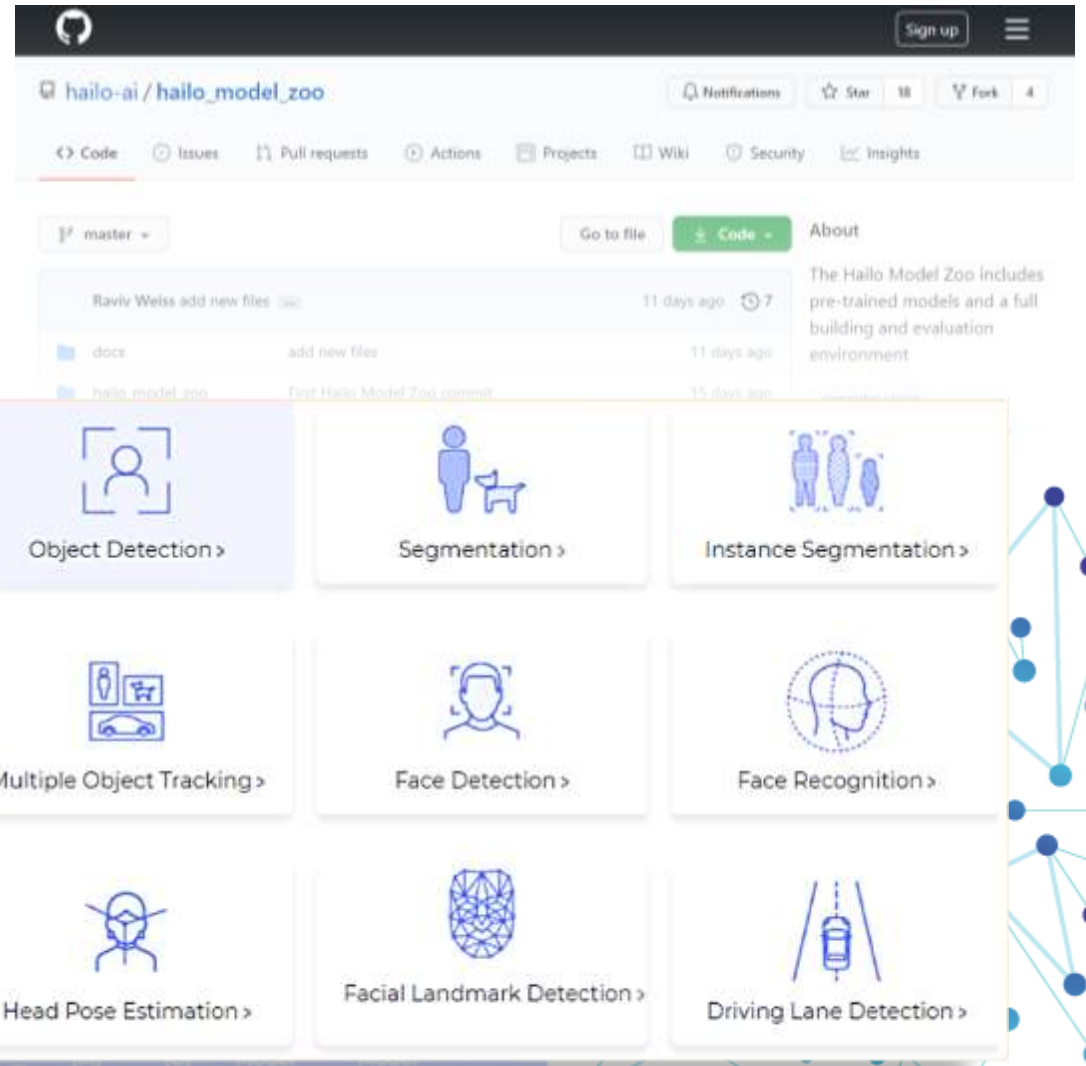
Hailo software component

Other software component

EBV Elektronik
An Avnet Company

The Hailo Model Zoo

- Opensource repository (https://github.com/hailo-ai/hailo_model_zoo/)
- A variety of common and state-of-the-art pre-trained models and tasks in TensorFlow and ONNX to re-train networks (https://github.com/hailo-ai/hailo_model_zoo/blob/master/docs/RETRAIN_ON_CUSTOM_DATASET.md)
- Quickly and easily reproduce Hailo-8 performance for evaluation and development



Model	mAP		(HxWxC)					
centernet_resnet_v1_18	26.37	24.57	512x512x3	14.23	15.73	pretrained	compiled	link
centernet_resnet_v1_50	31.79	30.09	512x512x3	30.08	28.56	pretrained	compiled	link
ssd_mobiledet_dsp	28.9	28.47	320x320x3	7.07	2.83	pretrained	compiled	link
ssd_mobilenet_v1	23.18	22.2	300x300x3	6.79	1.25	pretrained	compiled	link
ssd_mobilenet_v1_hd	17.67	16.43	720x1280x3	6.81	12.26	pretrained	compiled	link
ssd_mobilenet_v2	24.27	23.26	300x300x3	4.46	0.76	pretrained	compiled	link

Hailo-8 Measured Benchmarks

https://hailo.ai/products/ai-accelerators/hailo-8-ai-accelerator/#hailo8-benchmarks

NN Model	Input Resolution	FPS	Power (W)	FPS/W
Classification				
ResNet-50 v1	224×224	1357	3.7	364
MobileNet_v2_1.0	224×224	2434	2.1	1176
EfficientNet_M	240×240	890	4.0	221
ViT_base	224×224	138	2.6	52
Object Detection				
SSD_MobileNet_v1	300×300	1016	2.2	455
YOLOv5m	640×640	242	5.3	46
Semantic Segmentation				
stdc1	1024×1920	58	3.0	19

Notes:
1. Based on SDK version 3.9.0 (June 2021)
2. Measurements are done in room temperature through PCIe interface on Hailo-8 evaluation board
3. System host: Intel(R) Core(TM) i5-9400 CPU @ 2.90GHz
4. Performance figures are gives for processing 8 simultaneous streams

i.MX8m-plus + Hailo measured max performance

Note: idle power ~0.7W

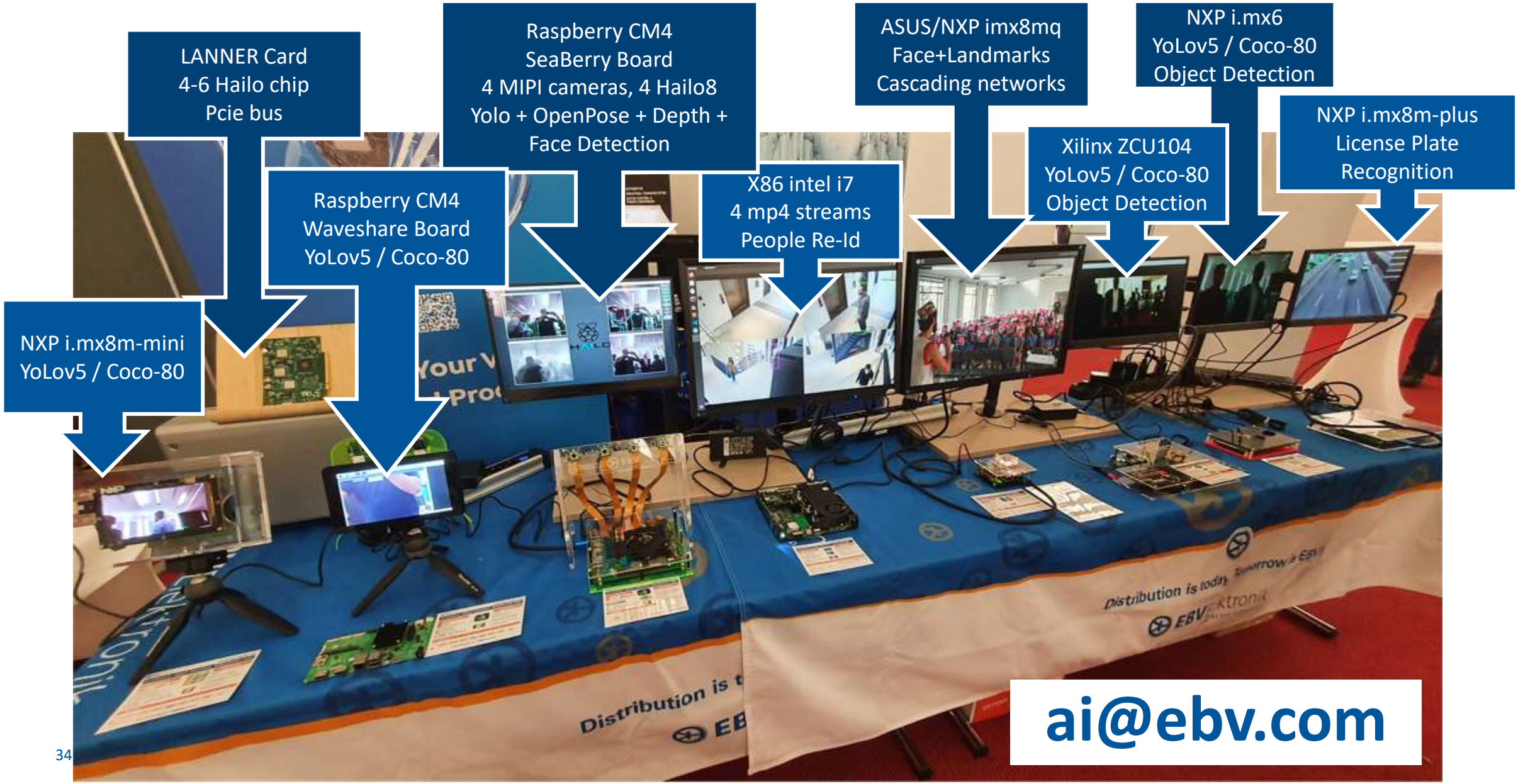
[190 fps]
- Using the large heatsink from the Hailo box (20x20mm size, 25mm tall, no fan, ambient temp is 23C)
- Running the yolov5m.hef at “full speed” (190fps) for a while I see an average power consumption of 5.8W.
- Internal Temperature goes up to 90C
- Heatsink temperature goes up to 70C (you cannot touch it..)

[120 fps]
- Power: 3.180 W
- Internal temp: 70 C
- Heatsink temp: 60 C

[60 fps]
- Power: 2.0 W
- Internal temp: 60 C
- Heatsink temp: 53 C

[30 fps]
- Power: 1.49 W
- Internal temp: 51 C
- Heatsink temp: 45 C

https://learnopencv.com/custom-object-detection-training-using-yolov5/


LANNER Card
4-6 Hailo chip
Pcie bus
Raspberry CM4
SeaBerry Board
4 MIPI cameras, 4 Hailo8
Yolo + OpenPose + Depth +
Face Detection
ASUS/NXP imx8mq
Face+Landmarks
Cascading networks
NXP i.mx6
YoLov5 / Coco-80
Object Detection
NXP i.mx8m-plus
License Plate
Recognition
Raspberry CM4
Waveshare Board
YoLov5 / Coco-80
X86 intel i7
4 mp4 streams
People Re-Id
Xilinx ZCU104
YoLov5 / Coco-80
Object Detection
NXP i.mx8m-mini
YoLov5 / Coco-80
Distribution is today
EBVElektronik

ai@ebv.com

ai@ebv.com

DEEP VISION CONSULTING

www.3dplus.ai

embeddedworld
Exhibition&Conference

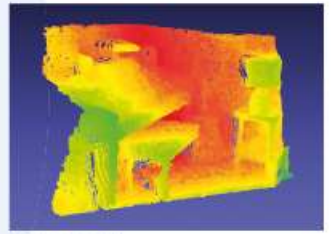
Software Library for 3D Reconstruction at the Edge

3D metric point clouds can be created from stereo cameras by correlating the differences between left and right video streams: this task is commonly known as stereo matching. State of the art **Deep Learning techniques for stereo matching** are computationally very demanding and the neural architectures are quite complex too: for all these reasons, running them on Hailo-8™ hosted on an embedded platform is not just a piece of cake.

Leveraging the flexibility of Hailo's neural core, Deep Vision Consulting created a software library for 3D reconstruction at the edge with stereo vision, by **custom-ly designing** a stereo matching Deep Learning model containing 3D operations, like 3D convolutions, and porting it to Hailo-8™ hosted on **NXP i.MX 8M Plus** or **Raspberry Pi CM4**. See performance in table.

LEFT RIGHT

DISPARITY



POINT CLOUD

Images are taken from Middelbury Stereo Dataset, D. Scharstein, et al. High-resolution stereo datasets with subpixel-accurate ground truth. GCPR 2014.

Input Resolution	Max Disparity	Point Cloud Size	FPS with Hailo-8™ and		
			Intel® i7	Raspi CM4	NXP i.MX 8
512x384	128	197.000	42	14	14
768x576		442.000	27	10	10

EBV Elektronik
An Avnet Company

Technology. Passion. EBV.

opsero

EW2024: quad camera Xilinx+Hailo

Multi-camera YOLOv5 on Zynq UltraScale+ with Hailo-8 AI Acceleration

Posted on February 15, 2024 | Jeff Johnson

See it live: The demo described in this post will be displayed live at the EBV Elektronik booth at Embedded World 2024 on April 9-11. I'll be attending too, so get in touch if you're keen to meet up.

Over the last few months I've been lucky to work with two very talented people on an interesting project for multi-camera machine vision applications: Gianluca Filippini and Mario Bergeron. Back in 2022, I was contacted by Gianluca, an engineer from EBV Elektronik. He was using our FPGA Drive FMC, an adapter for connecting Solid-state

embeddedworld
Exhibition&Conference

- G.Filippini & Jeff Johnson/Opsero (Canada)
- Full code available on github

https://www.fpgadeveloper.com/multi-camera-yolov5-on-zynq-ultrascale-with-hailo-8-ai-acceleration/

Multi-camera YOLOv5 on Zynq UltraScale+ with Hailo-8 AI Acceleration

https://github.com/fpgadeveloper/zynqmp-hailo-ai

hands-on

EBV Elektronik
An Avnet Company

ASTRIAL
SYSTEM ELECTRONICS for AI Innovation

www.astrial.ai

Technology. Passion. EBV.

Github code
Yocto build

Astrial hw
documents

Examples
full app

pre-built
yocto images
(full build)

```
├── 01_gpio_ctrl
│   ├── 00_deprecated
│   ├── 01_simple
│   ├── 02_timer
│   ├── 03_speedtest
│   ├── 04_pwm
│   ├── 05_extra
│   ├── README.txt
│   └── astrial pinout.xlsx
└── 02_image_ai
    ├── 01_camera_capture
    ├── 02_hailortcli
    ├── 03_object_detection
    ├── README.txt
    └── docs
```

ASTRIAL
SYSTEM ELECTRONICS for AI Innovation

SSID: ASTRIAL
SSID: ASTRIAL-5G
pass: astrial2024

astrial00	astrial10	astrial20	astrial30	astrial40
IP: 192.168.8.140	IP: 192.168.8.132	IP: 192.168.8.151	IP: 192.168.8.153	IP: 192.168.8.158
==========	==========	==========	==========	==========
username:	board user list:	board user list:	board user list:	board user list:
user00	user10	user20	user30	user40
user01	user11	user21	user31	user41
user02	user12	user22	user32	user42
…	…	…	…	…
user09	user19	user29	user39	user49
==========	==========	==========	==========	==========

Before we begin: 

#1: login on your assigned board using ssh:
`ssh user(xx)@192.168.8.XX`
try “htop” to see CPU resources …

#2: choose & change name for your user:
`change_my_name.sh nickname`

#3: lock your board hardware:
`astrial-lock.sh`
(try to lock again and see result)

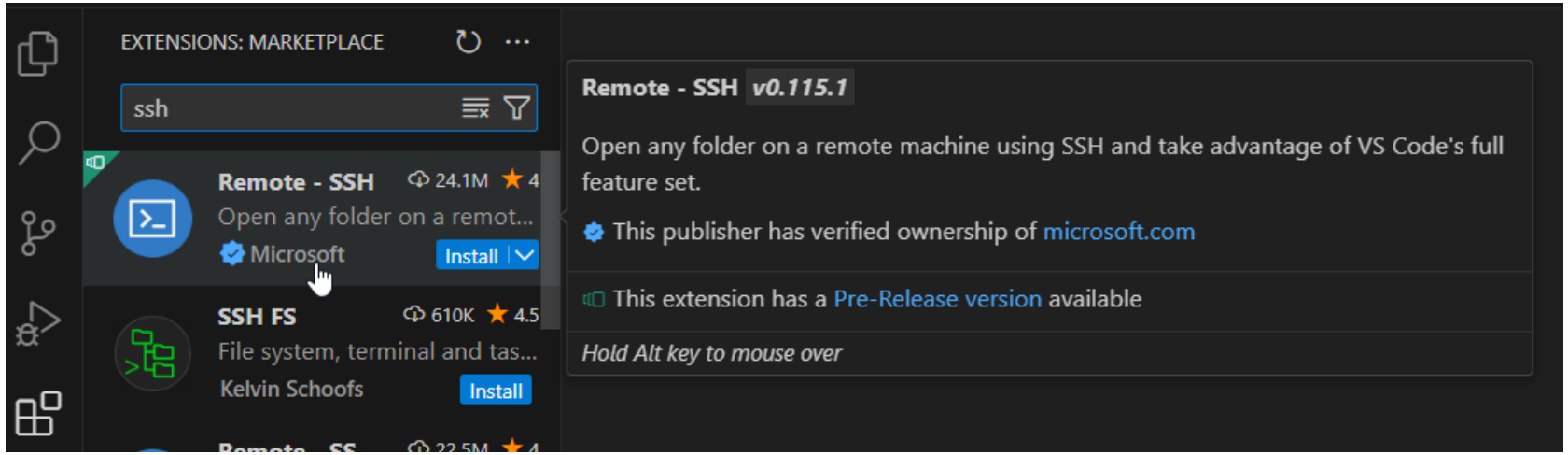
#4: unlock your board hardware:
`astrial-unlock.sh`
(try to unlock again and see result)

#4: folder “astrial_primer”, check content

Modify files locally & remotely: “vstudio + ssh” or WinSCP

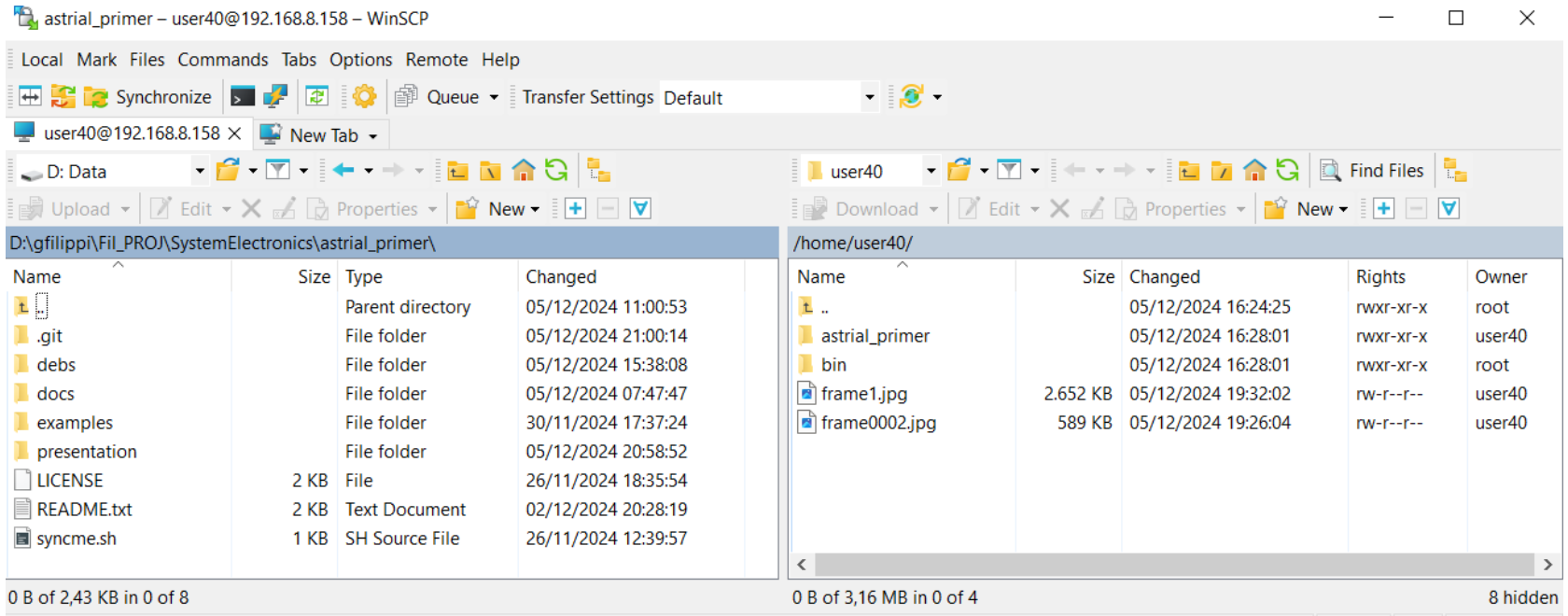
https://code.visualstudio.com/docs/remote/ssh

Microsoft Visual Studio

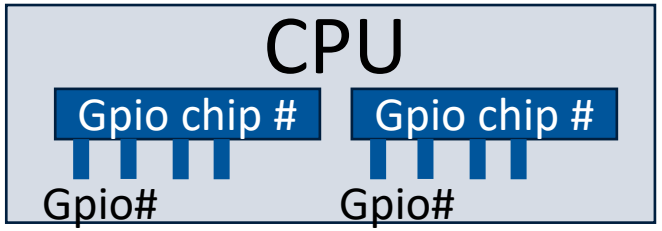


https://winscp.net/eng/download.php

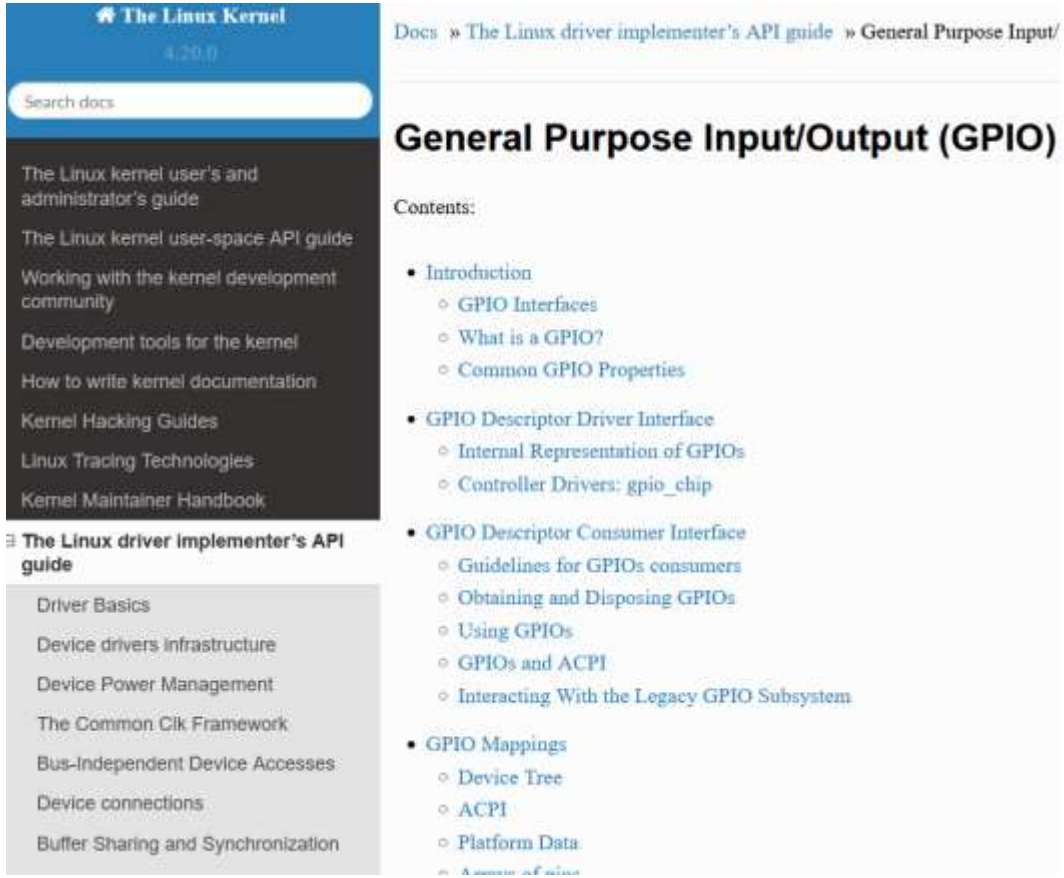
WinSCP



General Purpose IO (GPIO)



https://www.kernel.org/doc/html/v4.20/driver-api/gpio/index.html



Common GPIO Properties:

Active-High and Active-Low
It is natural to assume that a GPIO is "active" when its output signal is 1 ("high"), and inactive when it is 0 ("low"). It is possible to define a GPIO as being either active-high ("1" means "active", the default) or active-low ("0" means "active") so that drivers only need to worry about the logical signal and not about what happens at the line level.

Open Drain and Open Source
Sometimes shared signals need to use "open drain" (where only the low signal level is actually driven), or "open source" (where only the high signal level is driven) signaling. That term applies to CMOS transistors; "open collector" is used for TTL. A pullup or pulldown resistor causes the high or low signal level. This is sometimes called a "wire-AND"; or more practically, from the negative logic (low=true) perspective this is a "wire-OR". Some GPIO controllers directly support open drain and open source outputs; many don't.

General Purpose IO (GPIO)

In Linux user space “everything is a file”
(https://en.wikipedia.org/wiki/Everything_is_a_file)

From Linux Kernel documentation:
(https://www.kernel.org/doc/Documentation/gpio/sysfs.txt)

```
There are three kinds of entries in /sys/class/gpio:
   -        Control interfaces used to get userspace control over GPIOs;
   -        GPIOs themselves; and
   -        GPIO controllers ("gpio_chip" instances).

The control interfaces are write-only:
    /sys/class/gpio/
            "export" ... Userspace may ask the kernel to export control of
                         a GPIO to userspace by writing its number to this file.

            "unexport" ... Reverses the effect of exporting to userspace.
                         Example:  "echo 19 > unexport" will remove a "gpio19"
                         node exported using the "export" file.
```

Only one problem: it is deprecated (but still working..)

https://www.thegoodpenguin.co.uk/blog/stop-using-sys-class-gpio-its-deprecated/

```
./examples/
├── 01_gpio_ctrl
│   ├── 00_deprecated
│   │   ├── README.txt
│   │   └── trigger.sh
```

ARDUINO

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

```
#!/usr/bin/python3

import RPi.GPIO as GPIO
import time

led = 18
switch = 31

GPIO.setmode(GPIO.BOARD)
GPIO.setup(led, GPIO.OUT)
GPIO.setup(switch, GPIO.IN)

for i in range(10):
    GPIO.output(led, GPIO.HIGH)
    time.sleep(0.2)
    GPIO.output(led, GPIO.LOW)
    time.sleep(0.2)
    print('Switch status = ',
    GPIO.input(switch))

GPIO.cleanup()
```

Everything is a file

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

"Everything is a file" is an idea that Unix, and its derivatives, handle input/output to and from resources such as documents, hard-drives, modems, keyboards, printers and even some inter-process and network communications as simple streams of bytes exposed through the filesystem name space.[1] Exceptions include semaphores, processes and threads.

The advantage of this approach is that the same set of tools, utilities and APIs can be used on a wide range of resources and a number of file types. When a file is opened, a file descriptor is created, using the file path as an addressing system. The file descriptor is then a byte stream I/O interface on which file operations are performed. File descriptors are also created for objects such as anonymous pipes and network sockets – and therefore a more accurate description of this feature is *Everything is a file descriptor*.[2][3]

Additionally, a range of pseudo and virtual filesystems exists which exposes internal kernel data, such as information about processes, to user space in a hierarchical file-like structure.[4] These are mounted into the single file hierarchy.

An example of this purely virtual filesystem is under /proc that exposes many system properties as files. All of these files, in the broader sense of the word, have standard Unix file attributes such as an owner and access permissions, and can be queried by the same classic Unix tools and filters. However, this is not universally considered a fast or portable approach. Some operating systems

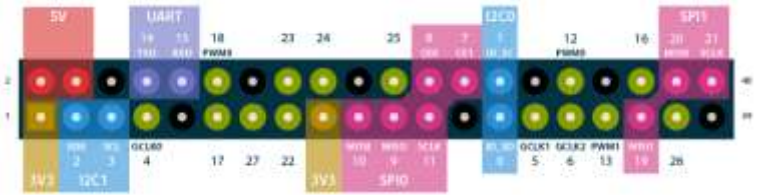
GPIO mapping and where to find it

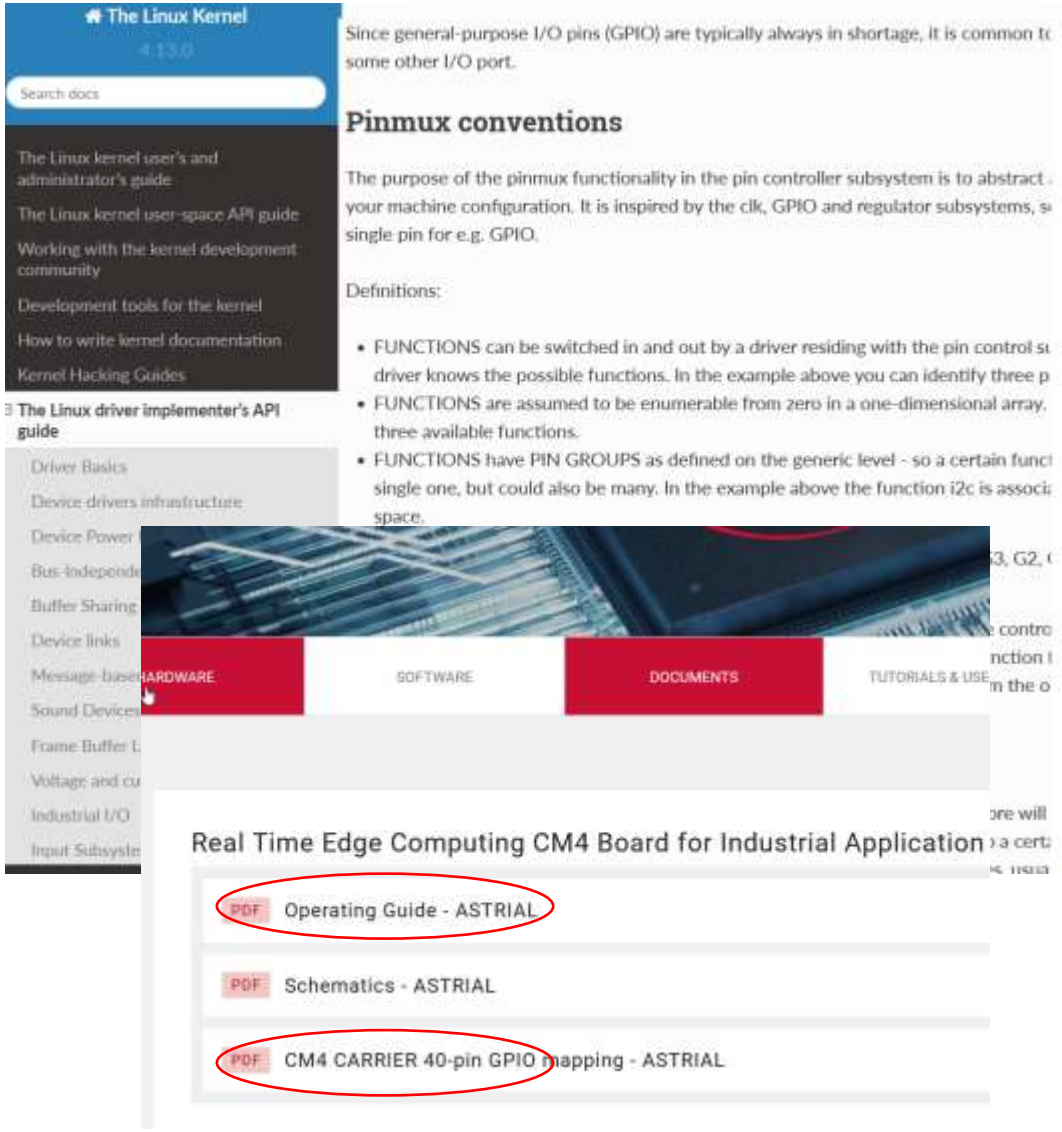
https://www.kernel.org/doc/html/v4.13/driver-api/pinctl.html

Since general-purpose I/O pins (GPIO) are typically always in shortage, it is common to be able to use almost any pin as a GPIO pin if it is not currently in use by some other I/O port. The purpose of the pinmux functionality in the pin controller subsystem is to abstract and provide pinmux settings to the devices you choose to instantiate in your machine configuration.

The user manual of the CPU vendor is the starting point for GPIO map.
IMPORTANT: see Astrial documents for Rpi 40pin carrier GPIO map

Default Functions	i.Mx8MP Name	Name	CM4 CARRIER 40-pin GPIO mapping		Name	i.Mx8MP Name	Default Functions
		3.3V	1	2	5V		
I2C5 SDA	GPIO3.IO[25]	GPIO2	3	4	5V		
I2C5 SCL	GPIO3.IO[21]	GPIO3	5	6	GND		
GPIO	GPIO2.IO[8]	GPIO4	7	8	GPIO14	GPIO4.IO[21]	UART1 TX
		GND	9	10	GPIO15	GPIO4.IO[22]	UART1 RX
GPIO	GPIO4.IO[23]	GPIO17	11	12	GPIO18	GPIO5.IO[13]	GPIO
GPIO	GPIO2.IO[5]	GPIO27	13	14	GND		
GPIO	GPIO2.IO[0]	GPIO22	15	16	GPIO23	GPIO2.IO[1]	GPIO
		3.3V	17	18	GPIO24	GPIO2.IO[2]	GPIO
SPI1 MOSI	GPIO5.IO[7]	GPIO10	19	20	GND		
SPI1 MISO	GPIO5.IO[8]	GPIO9	21	22	GPIO25	GPIO2.IO[3]	GPIO
SPI1 SCLK	GPIO5.IO[6]	GPIO11	23	24	GPIO8	GPIO5.IO[9]	SPI1 CS
		GND	25	26	GPIO7	GPIO4.IO[26]	GPIO
I2C6 SDA	GPIO3.IO[20]	GPIO0	27	28	GPIO1	GPIO3.IO[19]	I2C6 SCL
GPIO	GPIO2.IO[9]	GPIO5	29	30	GND		
GPIO	GPIO4.IO[27]	GPIO6	31	32	GPIO12	GPIO5.IO[2]	PWM4
PWM1	GPIO5.IO[5]	GPIO13	33	34	GND		
GPIO	GPIO5.IO[12]	GPIO19	35	36	GPIO16	GPIO4.IO[24]	GPIO
GPIO	GPIO2.IO[4]	GPIO26	37	38	GPIO20	GPIO5.IO[11]	GPIO
		GND	39	40	GPIO21	GPIO5.IO[10]	GPIO



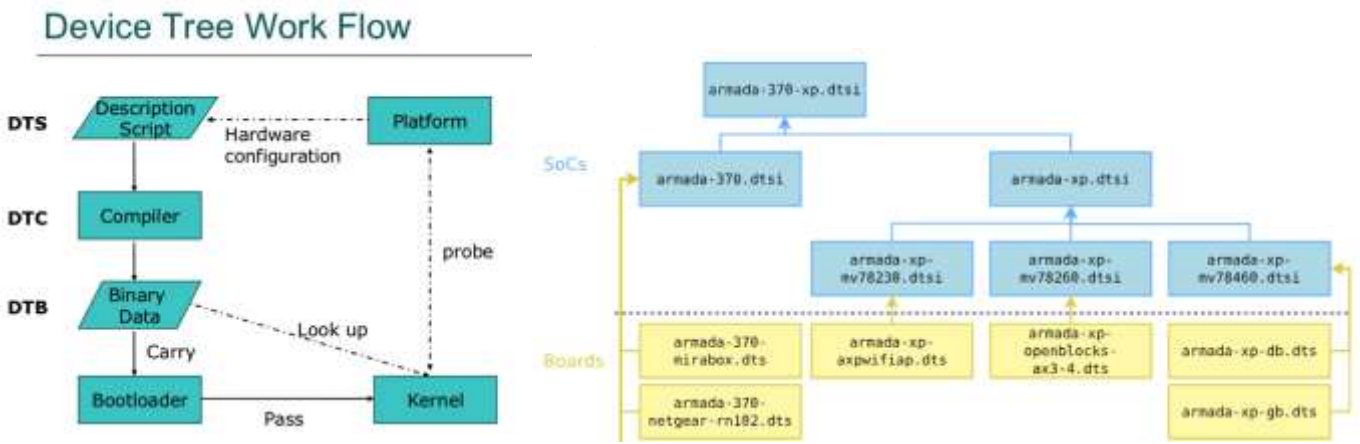


Linux Device Tree to export ALT GPIO

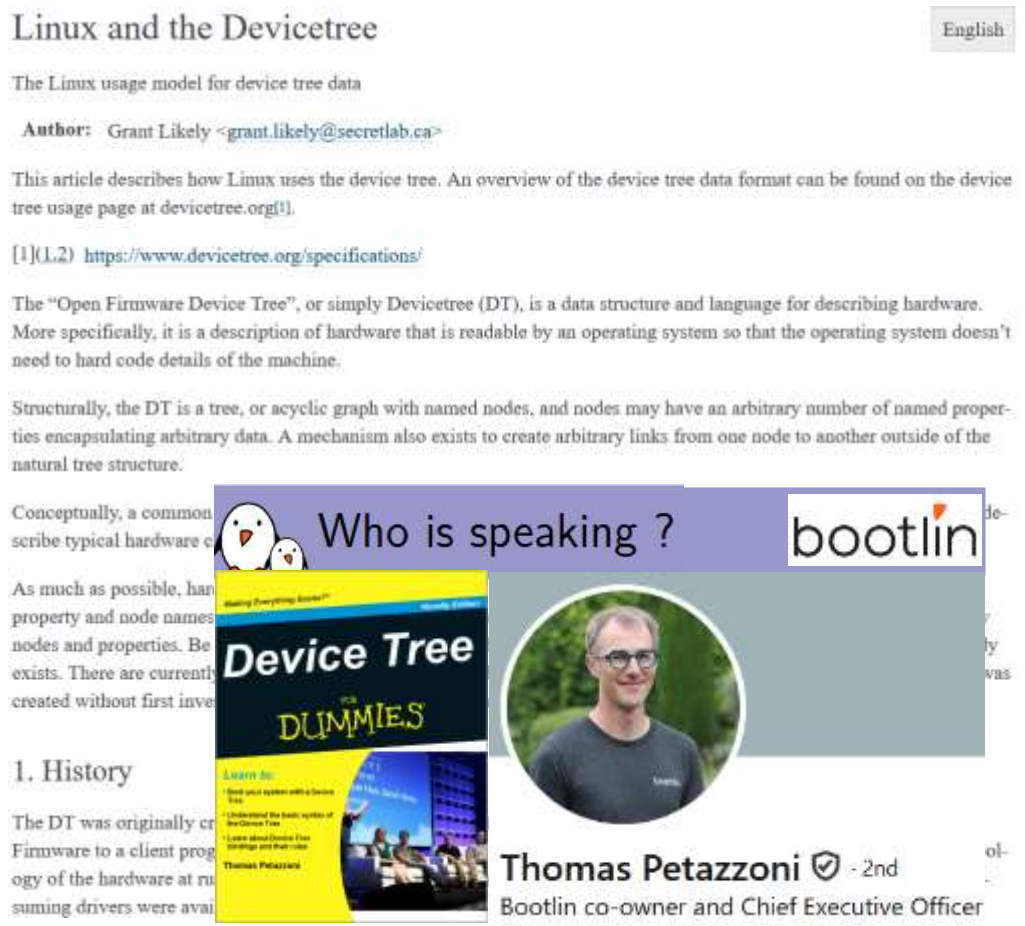
The Device Tree is a Hardware Description Language that can be used to describe the system hardware in a tree data structure. In this structure, each tree node describes a device. The source code of the Device Tree is compiled by the Device Tree Compiler (DTC) to form the Device Tree Blob (DTB), readable by the kernel upon startup.

https://www.develer.com/en/blog/linux-kernel-the-device-tree/

https://robbie-cao.github.io/2016/09/device-tree



https://docs.kernel.org/devicetree/usage-model.html


Linux and the Devicetree

English

The Linux usage model for device tree data

Author: Grant Likely <grant.likely@secretlab.ca>

This article describes how Linux uses the device tree. An overview of the device tree data format can be found on the device tree usage page at devicetree.org[1].

[1](1,2) https://www.devicetree.org/specifications/

The "Open Firmware Device Tree", or simply Devicetree (DT), is a data structure and language for describing hardware. More specifically, it is a description of hardware that is readable by an operating system so that the operating system doesn't need to hard code details of the machine.

Structurally, the DT is a tree, or acyclic graph with named nodes, and nodes may have an arbitrary number of named properties encapsulating arbitrary data. A mechanism also exists to create arbitrary links from one node to another outside of the natural tree structure.

Conceptually, a common

As much as possible, har
property and node names
nodes and properties. Be
exists. There are currentl
created without first inve

1. History

The DT was originally cr
Firmware to a client prog
ogy of the hardware at ru
suming drivers were avai

Astrial multiple dts definition:
https://github.com/System-Electronics/linux-imx-lf-5.15.71/tree/main/arch/arm64/boot/dts/system-electronics
https://github.com/System-Electronics/meta-sysele-nxp-5.15.71/blob/main/conf/machine/astrial-imx8mp.conf

Astrial default .dtb (binary loaded at boot):
`/run/media/boot-mmcblk1p1/imx8mp-astrial.dtb`

SysFs is deprecated: welcome “libgpiod”

The new library provides a set of command line tools and programming API to properly handle GPIO

https://www.ics.com/blog/gpio-programming-exploring-libgpiod-library

- **gpiodetect** - List all GPIO chips present on the system, their names, labels and number of GPIO lines.
- **gpioinfo** - List all lines of specified GPIO chips, their names, consumers, direction, active state and additional flags.
- **gpioget** - Read values of specified GPIO lines.
- **gpioset** - Set values of specified GPIO lines, and potentially keep the lines exported and wait until timeout, user input or signal.
- **gpiofind** - Find the GPIO chip name and line offset given the line name.
- **gpiomon** - Wait for events on GPIO lines, specifying which events to watch, how many events to process before exiting or if the events should be reported to the console.

Use the “01_simple” example to trigger an LED. Choose the “C” code or “Python” code.

```
sudo python ./simple_trigger.py
```

Important: LOCK your Astrial board.

```
├── examples
│   ├── 01_gpio_ctrl
│   │   ├── 01_simple
│   │   │   ├── c
│   │   │   │   ├── Makefile
│   │   │   │   ├── buildme.sh
│   │   │   │   ├── simple_trigger.c
│   │   │   └── python
│   │   │       ├── led_off.sh
│   │   │       └── simple_trigger.py
```

The “02_timer” is the same code but the call to trigger the led is delegated to a linux hw timer.

```
sudo ./timer_trigger -f 4 -s
```

```
├── examples
│   ├── 01_gpio_ctrl
│   │   ├── 02_timer
│   │   │   └── c
│   │   │       ├── Makefile
│   │   │       ├── buildme.sh
│   │   │       ├── logger.h
│   │   │       ├── timer_trigger.c
```

Kernel.org git repositories

Git repositories hosted at kernel.org

index

Name	Description
pub/scm/libs	
libgpiod/libgpiod.git	C library and tools for interacting with the linux GPIO character device

README License

```
# SPDX-License-Identifier: CC-BY-SA-4.0
# SPDX-FileCopyrightText: 2017-2023 Bartosz Golaszewski <brgl@bgdev.pl>

libgpiod
========

  libgpiod - C library and tools for interacting with the linux GPIO
             character device (gpiod stands for GPIO device)

Since linux 4.8 the GPIO sysfs interface is deprecated. User space should use
the character device instead. Version 2 of libgpiod requires GPIO character
device uAPI v2 which was first released in linux 5.10. This library
encapsulates the ioctl calls and data structures behind a straightforward API.

RATIONALE
---------

The new character device interface guarantees all allocated resources are
freed after closing the device file descriptor and adds several new features
that are not present in the obsolete sysfs interface (like event polling,
setting/reading multiple values at once or open-source and open-drain GPIOs).

Unfortunately interacting with the linux device file can no longer be done
using only standard command-line tools. This is the reason for creating a
library encapsulating the cumbersome, ioctl-based kernel-userspace interaction
in a set of convenient functions and opaque data structures.

Additionally this project contains a set of command-line tools that should
allow an easy conversion of user scripts to using the character device.
```

Understanding GPU governor

The state of the CPU clock will have an impact on the system performance. Use “cpufrequtils” to check your status if needed.

```
> sudo cpufreq-info
```

Performance

The CPUfreq governor "performance" sets the CPU statically to the highest frequency.

Powersave

The CPUfreq governor "powersave" sets the CPU statically to the lowest frequency.

Userspace

The CPUfreq governor "userspace" allows the user, or any userspace program running with UID "root", to set the CPU to a specific frequency by making a sysfs file "scaling_setspeed" available in the CPU-device directory.

Ondemand

The CPUfreq governor "ondemand" sets the CPU frequency depending on the current system load. Load estimation is triggered by the scheduler through the update_util_data->func hook; when triggered, cpufreq checks the CPU-usage statistics over the last period and the governor sets the CPU accordingly.

https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

CPU frequency and voltage scaling code in the Linux(TM) kernel

L i n u x C P U F r e q

C P U F r e q G o v e r n o r s

- information for users and developers -

Dominik Brodowski <linux@brodo.de>
some additions and corrections by Nico Golde <nico@ngolde.de>
Rafael J. Wysocki <rafael.j.wysocki@intel.com>
Viresh Kumar <viresh.kumar@linaro.org>

Clock scaling allows you to change the clock speed of the CPUs on the fly. This is a nice method to save battery power, because the lower the clock speed, the less power the CPU consumes.

Contents:

==========

Most cpufreq drivers (except the intel_pstate and longrun) or even most cpu frequency scaling algorithms only allow the CPU frequency to be set to predefined fixed values. In order to offer dynamic frequency scaling, the cpufreq core must be able to tell these drivers of a "target frequency". So these specific drivers will be transformed to offer a "->target/target_index/fast_switch()" call instead of the "->setpolicy()" call. For set_policy drivers, all stays the same, though.

Understanding GPU governor

The state of the CPU clock will have an impact on the system performance. Use “cpufrequtils” to check your status if needed.

```
> sudo cpufreq-info

analyzing CPU 0:
  driver: cpufreq-dt
  CPUs which run at the same hardware frequency: 0 1 2 3
  CPUs which need to have their frequency coordinated by software: 0 1 2 3
  maximum transition latency: 182 us.
  hardware limits: 1.20 GHz - 1.60 GHz
  available frequency steps: 1.20 GHz, 1.60 GHz
  available cpufreq governors: conservative, ondemand,
                               userspace, powersave, performance, schedutil
  current policy: frequency should be within 1.20 GHz and 1.60 GHz.
                  The governor "ondemand" may decide which speed to use
                  within this range.
  current CPU frequency is 1.20 GHz (asserted by call to hardware).
  cpufreq stats: 1.20 GHz:99.88%, 1.60 GHz:0.12%
```

```
> sudo cpufreq-set -g performance
> sudo cpufreq-info

current policy: frequency should be within 1.20 GHz and 1.60 GHz.
                The governor "performance" may decide which speed to use
                within this range.
current CPU frequency is 1.60 GHz (asserted by call to hardware).
```

https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

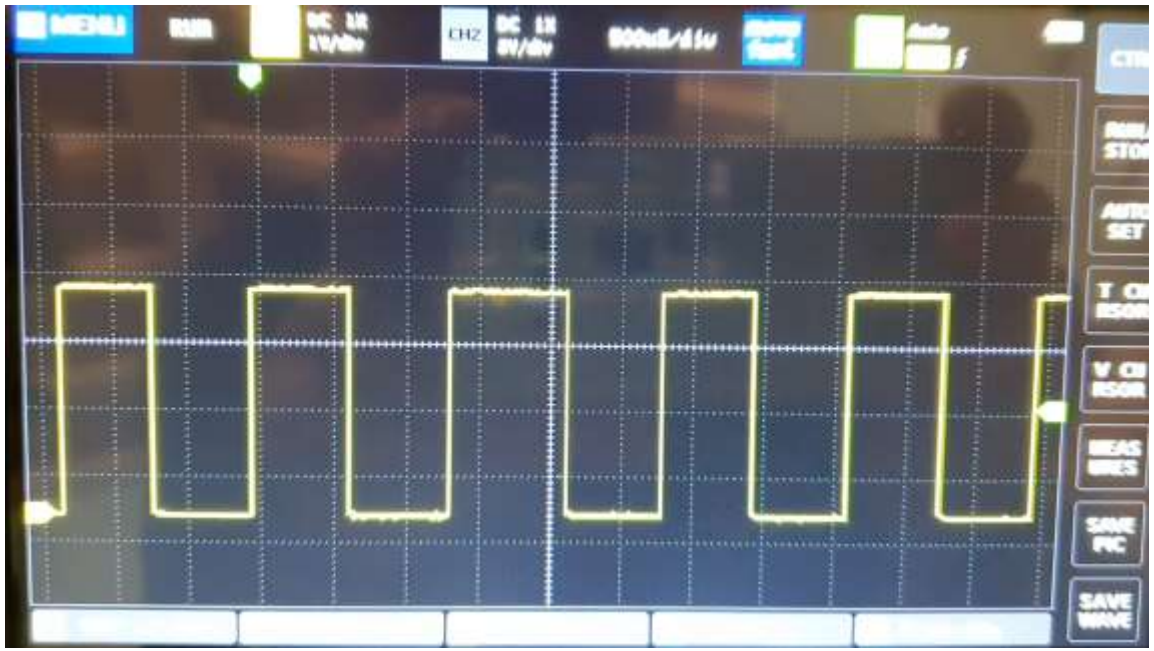
```
CPU frequency and voltage scaling code in the Linux(TM) kernel


                L i n u x    C P U F r e q

                C P U F r e q   G o v e r n o r s

                - information for users and developers -


                Dominik Brodowski  <linux@brodo.de>
            some additions and corrections by Nico Golde <nico@ngolde.de>
                Rafael J. Wysocki <rafael.j.wysocki@intel.com>
                Viresh Kumar <viresh.kumar@linaro.org>



   Clock scaling allows you to change the clock speed of the CPUs on the
    fly. This is a nice method to save battery power, because the lower
            the clock speed, the less power the CPU consumes.


Contents:
---------
1. What Is A CPUFreq Governor?
==============================

Most cpufreq drivers (except the intel_pstate and longrun) or even most
cpu frequency scaling algorithms only allow the CPU frequency to be set
to predefined fixed values.  In order to offer dynamic frequency
scaling, the cpufreq core must be able to tell these drivers of a
"target frequency". So these specific drivers will be transformed to
offer a "->target/target_index/fast_switch()" call instead of the
"->setpolicy()" call. For set_policy drivers, all stays the same,
though.
```

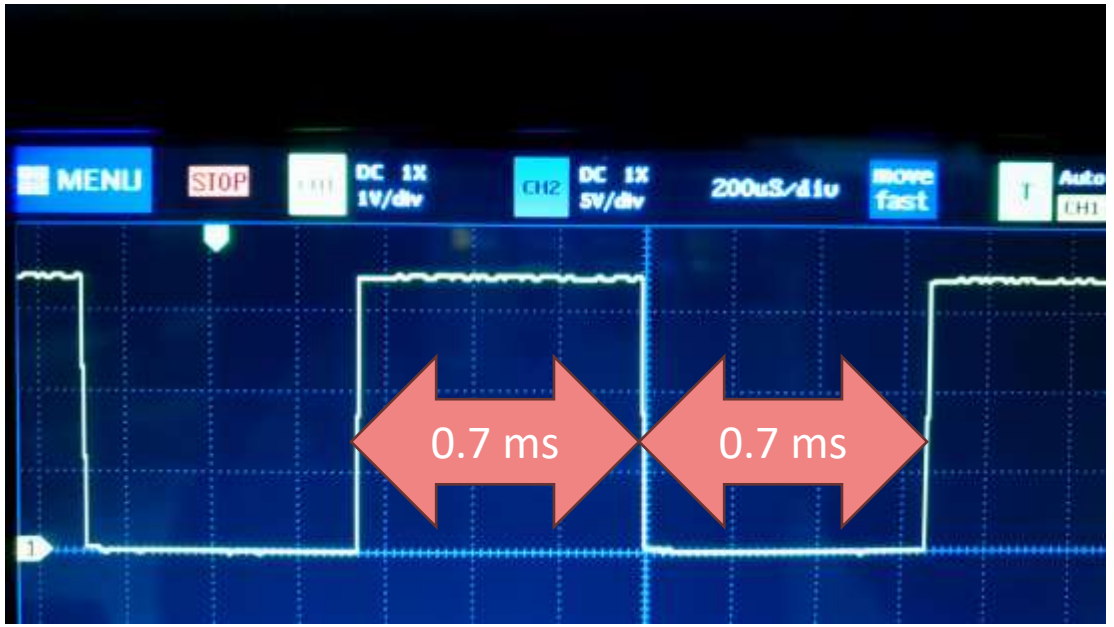
Speed limit for user space GPIO control

```
├── examples
│   ├── 01_gpio_ctrl
│   │   ├── 03_speedtest
│   │   │   ├── 20241203_083057.jpg
│   │   │   ├── README.txt
│   │   │   └── c
│   │   │       ├── Makefile
│   │   │       ├── buildme.sh
│   │   │       ├── cleanup.sh
│   │   │       ├── simple_trigger.c
```

- oscilloscope
- run toggle with no sleep
- see max freq for libgpio toggle & jitter

CPU_governor PERFORMANCE : 740 Hz
CPU_governor ONDEMAND : 720 Hz





What about PWM ?

Astrial default device tree already exposes n.2 “pulse width modulation” hardware. These devices are programmable in frequency, duty cycle and polarity.

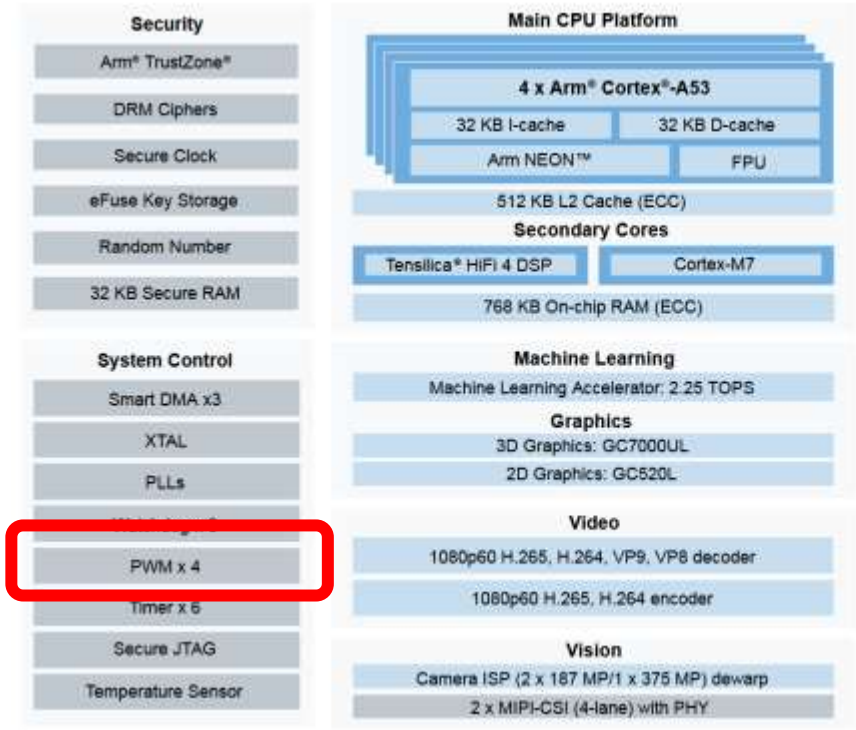
There is no specific library, still using SysFS.

Advantages: very accurate and stable signal, useful for many variable speed controls. Can be used to generate high speed signals, and so it can be used to modulate DC value (see LED brightness)

```
├── examples
│   ├── 01_gpio_ctrl
│   │   ├── 04_pwm
│   │   │   ├── python
│   │   │   │   ├── README.txt
│   │   │   │   ├── pwm.py
│   │   │   │   ├── pwm_off.py
│   │   │   │   └── pwm_on.py
│   │   │   └── script
│   │   │       ├── pwm_export.sh
│   │   │       ├── pwm_run.sh
│   │   │       └── pwm_unexport.sh
```

NOTE: using the pwm hardware will offload the CPU from any type of scheduling and computation (check with “htop”). The Linux scheduler does not affect external pwm generation.

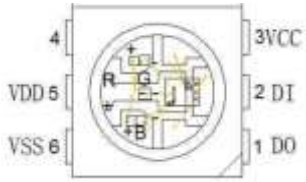
NXP i.mx8




```
04_pwm/python$ sudo python ./pwm_on.py

04_pwm/python$ sudo python ./pwm_off.py
```

Not doable using“libgpio” in user space: WS2812 RGB leds

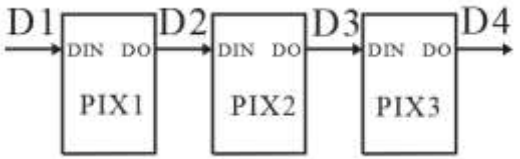
Some devices do require very accurate timing to control them.
This RGB (also RGBW variant) uses different waveforms to signal “0” and “1” over a single wire communication. Each “pixel” will consume 3 bytes (8 bit/color) and forward the remaining bits to the chain of leds.

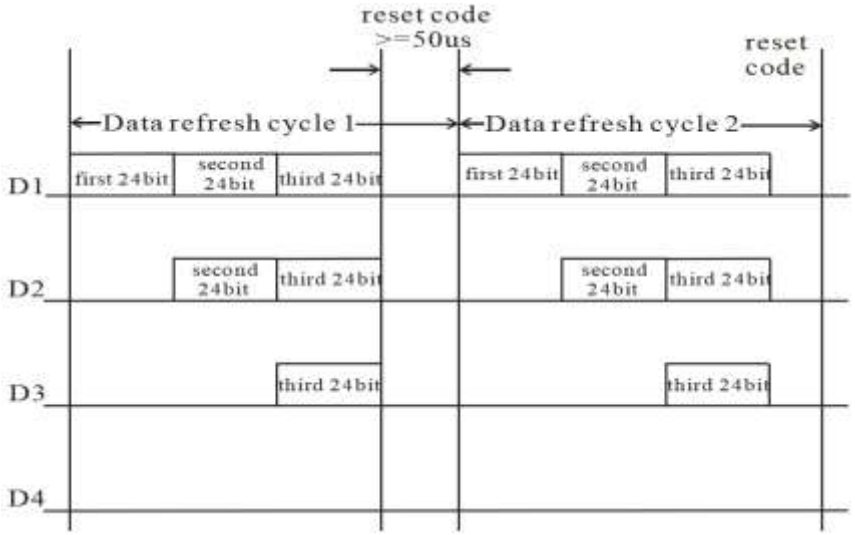


NO.	Symbol	Function description
1	DOUT	Control data signal output
2	DIN	Control data signal input
3	VCC	Power supply control circuit
4	NC	
5	VDD	Power supply LED
6	VSS	Ground

Possible Solutions:

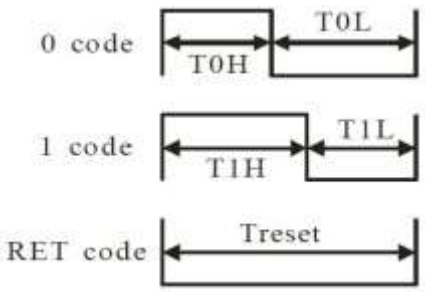
- “bitbang” with low-level code (down to assembly) to guarantee cycle-accurate timing
- use programmable hw (usually SPI) with some “tweaks”
- use companion MCU with RTOS for external phjeripheral management. (this is also common with ROS2 and message passing APIs)
- use a Real-Time Linux kernel version





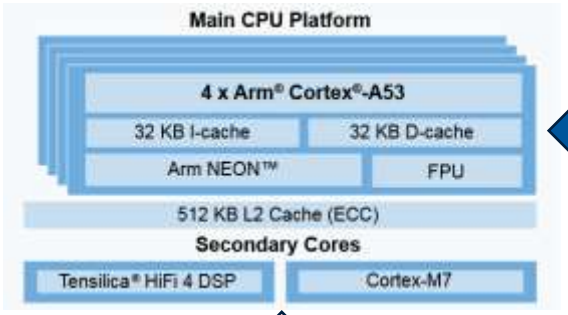
Data transfer time(TH+TL=1.25μs±600ns)

T0H	0 code, high voltage time	0.4μs	±150ns
T1H	1 code, high voltage time	0.8μs	±150ns
T0L	0 code, low voltage time	0.85μs	±150ns
T1L	1 code, low voltage time	0.45 μs	±150ns
RES	The unit of frame, low voltage time	Above 50μs	



NOTE: Linux RT in mainline: https://arstechnica.com/gadgets/2024/09/real-time-linux-is-officially-part-of-the-kernel-after-decades-of-debate/

External peripherals with MCU: pro and cons



UART, SPI, I2C

MCU

ARDUINO

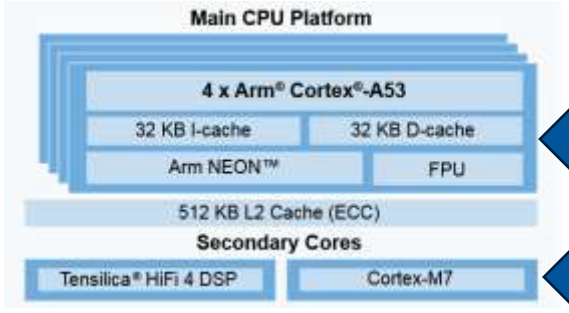
MCUXpresso

STM32 Cube

Using an external MCU allows a more flexible architecture
(it is possible to connect multiple MCU, single functionality)

PRO:
- Wider selection of hardware for specific requirements.
- Easier code management using favorite toolchain
- Simpler communication protocols (I2C / SPI or custom bus)
- I2C allows for multiple (independent) MCU on the same two-wire bus

CONS:
- Requires software security auditing, also on hw design
- Requires «tricks» to debug (beyond printf)
- Remote software update requires extra hardware to guarantee «fallback» for failure protection in case the MCU fails to update (external bus might have data loss)



Using the internal M7 everything must use NXP toolchain.
The MPU will run linux (yocto) and the MCU will run the RTOS (FreeRTOS)

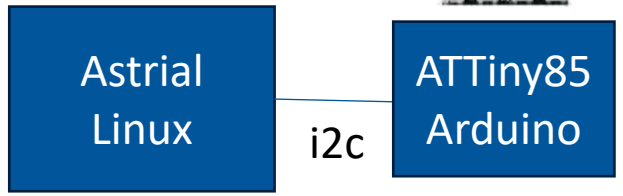
PRO:
- The binary image of the MCU is included in the linux image (the ARM will boot and update the MCU). This is an advantage for CE/CRA certification.
- Secure software update (OTA)
- Debugging with pro-tools (via JTAG) will allow to step-lock heterogeneous cores (clock stops for all cores, including MCU)

CONS:
- Requires custom device tree to expose pins from the MCU
- Documentation is not «easy» since not all the interfaces are standardized (RPMSG)
https://docs.kernel.org/staging/rpmsg.html

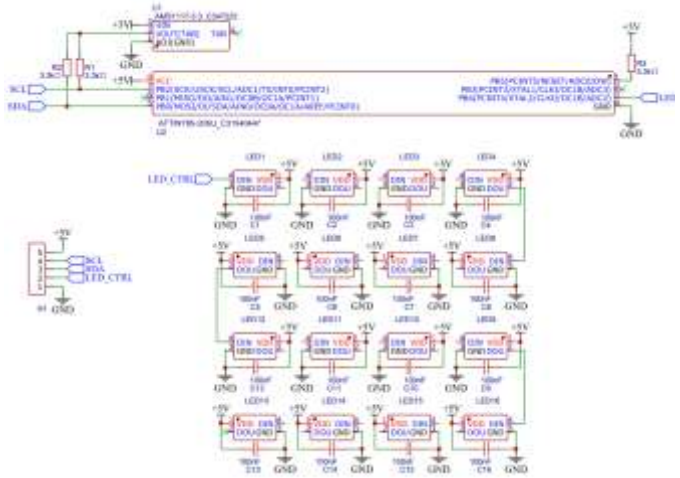
(hdunicorn)

Tic Tac Toe with Arduino and WS2812b

```python
import smbus2, time

usleep = lambda x: time.sleep(x/1000000.0)

#
# the LED are a sequential string, we need
# a mapping function to properly display the
# user matrix "aligned"
#
display_mtx=[   3,  4,  11,  12,
                2,  5,  10,  13,
                1,  6,   9,  14,
                0,  7,   8,  15 ]

# flatten list of list
def flatten(xss):
        return [x for xs in xss for x in xs]

def display_4x4(data):
        bus = smbus2.SMBus(4)
        addr = 0x6a

        # convert list of list
        tmp = list(flatten(data))
        pattern = list(flatten(data))

        # remap data into a pattern
        for i in range(16):
           pattern[display_mtx[i]]=tmp[i]

        # add API_VER and CMD=2
        pattern = [0,0,2] + pattern

        # write to I2C
        rv = False
        while not rv:
           try:
              bus.write_i2c_block_data(addr, 0, pattern)
              rv=True
           except:
              usleep(100)
              pass

        bus.close()
```



The python client/server game is taking care of the user interaction. A separate python routine (smbus) is sending a specific command over I2C and the MCU will refresh the RGB array. The main CPU on Astrial is off-loaded from time-accurate controls.



```
void i2cReceive( uint8_t n )
{
    if ( n > MAX_TRANSMISSION )
    {
        n = MAX_TRANSMISSION;
    }
    for (uint8_t i = 0; i < n; i++)
    {
        if ( TinyWireS.available( ) )
        {
            request[ i ] = TinyWireS.read( );
            request_len = i + 1;
        }
        else
        {
            break;
        }
    }

    while ( TinyWireS.available( ) )
    {
        TinyWireS.read( );
    }

    handleCommand = true;
    rv = request_len;
}


void i2cRequest( )
{
    // just a ping
    TinyWireS.write( rv );
}
```

```
/05_extra
├── docs
│   └── WS2812.pdf
├── rgbled
│   ├── README.txt
│   ├── attinyrgb
│   │   ├── attinyrgb.ino
│   │   ├── uncrustify.cfg
│   │   └── uncrustify.sh
│   ├── docs
│   ├── test_matrix.py
│   └── test_white.py
└── tic-tac-toe
    ├── README.txt
    ├── client_player.py
    ├── rgbled.py
    └── server_tictactoe.py
```

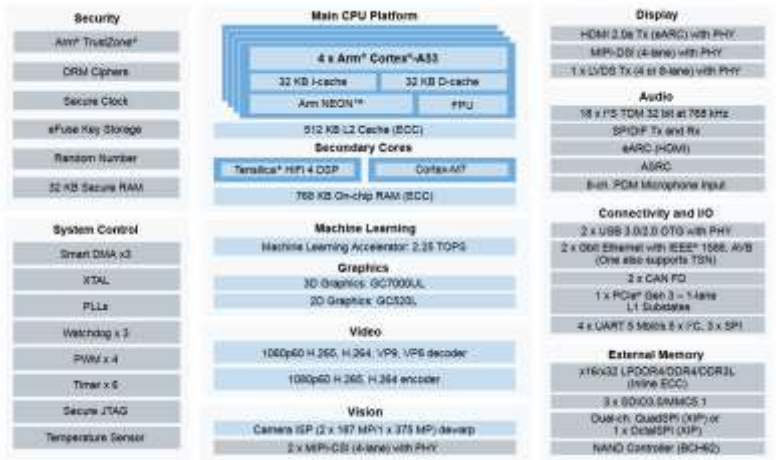
Image Camera: MIPI-CSI and USB devices

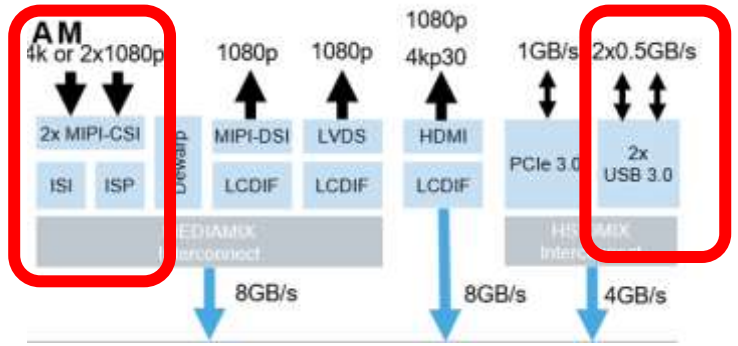
Camera for image acquisition do require an high data transfer rate, especially if there is no data-compression (raw format).

The MIPI alliance defined a hardware specification and a software protocol (Camera Serial Interface) to allow high data transfer with low latency. The improvement in performance requires more software support (ISP calibration and kernel drivers)

https://www.mipi.org/specifications/csi-2

Features	USB 3.0	MIPI CSI-2
Availability on SoC	On high -end SoCs	Many (Typically 6 lanes available)
Bandwidth	400 MB/s	320 MB/s/lane 1280 MB/s (with 4 lanes)*
Cable Length	< 5 meters	<30 cm
Space Requirements	High	Low
Plug-and-play	Supported	Not supported
Development Costs	Low	Medium to High





USB camera allows a simpler and faster integration (Linux UVC video)
https://www.kernel.org/doc/html/v4.9/media/v4l-drivers/uvcvideo.html
Careful considerations on data compression and latency must be taken in account also for high-speed USB protocols (USB 3.0)

Software for vision: Video4Linux, Gstreamer and OpenCV

Video4Linux is a collection of device drivers and an API for supporting realtime video capture on Linux systems.[1] It supports many USB webcams, TV tuners, and related devices, standardizing their output, so programmers can easily add video support to their applications.

Video4Linux is responsible for creating V4L2 device nodes aka a device file (/dev/videoX, /dev/vbiX and /dev/radioX) and tracking data from these nodes. The device node creation is handled by V4L device drivers using the video_device struct (v4l2-dev.h) and it can either be allocated dynamically or embedded in another larger struct.

https://www.kernel.org/doc/html/v4.12/media/v4l-drivers/index.html

GStreamer is a framework for creating streaming media applications. (C code)
The GStreamer framework is designed to make it easy to write applications that handle audio or video or both.

Gstreamer is a good framework for designing even high-end audio applications which put high demands on latency.

Specifically, GStreamer provides

- an API for multimedia applications
- a plugin architecture
- a pipeline architecture
- a mechanism for media type handling/negotiation
- a mechanism for synchronization
- a set of tools

https://gstreamer.freedesktop.org/documentation/tutorials/index.html?gi-language=c

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

https://opencv.org/about/

Software for vision: Video4Linux, Gstreamer and OpenCV

V4L

Video4Linux is a collection of device drivers and an API for supporting realtime video capture on Linux systems.[1] It supports many USB webcams, TV tuners, and related devices, standardizing their output, so programmers can easily add video support to their applications.

Video4Linux is responsible for creating V4L2 device nodes aka a device file (/dev/videoX, /dev/vbiX and /dev/radioX) and tracking data from these nodes. The device node creation is handled by V4L device drivers using the video_device struct (v4l2-dev.h) and it can either be allocated dynamically or embedded in another larger struct.

https://www ml/v4.12/media/v4l-drivers/index

GPLv2 license

GStreamer is a framework for creating streaming media applications. (C code)
The GStreamer framework is designed to make it easy to write applications that handle audio or video or both.

gst-launch-1.0 is a tool that builds and runs basic GStreamer pipelines.

In its simplest form, a PIPELINE-DESCRIPTION is a list of elements separated by exclamation marks (!).

Please note that gst-launch-1.0 is primarily a debugging tool. You should not build applications on top of it.

For applications, write a little python script or Rust application (or use whatever other programming language you prefer) and use the gst_parse_launch() function of the GStreamer API as an easy way to construct pipelines from pipeline descriptions.

L-GPL license

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

https://opencv.org/a

BSD license

Camera Image capture

For this example you can only use MIPI-CSI camera.
Camera will show up as /dev/video3

```
./01_camera_capture/
├── README.txt
├── gstreamer
│   ├── README.txt
│   ├── capture_frame_jpeg.sh
│   ├── capture_frame_raw.sh
│   └── check_formats.sh
└── python
    └── capture_frame.py
```

#1 check formats using V4L

```
> cd gstreamer
> sudo v4l2-ctl --list-formats-ext -d /dev/video3


> sudo ./check_formats.sh
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture

        [0]: 'YUYV' (YUYV 4:2:2)
                Size: Stepwise 176x144 - 4096x3072 with step 16/8
        [1]: 'NV12' (Y/CbCr 4:2:0)
                Size: Stepwise 176x144 - 4096x3072 with step 16/8
        [2]: 'NV16' (Y/CbCr 4:2:2)
                Size: Stepwise 176x144 - 4096x3072 with step 16/8
        [3]: 'RG10' (10-bit Bayer RGRG/GBGB)
                Size: Stepwise 176x144 - 4096x3072 with step 16/8
```

MIPI-CSI camera providing only “RAW” formats (no compression)

```
user90@astrial-90:~$ sudo v4l2-ctl --list-formats-ext -d /dev/video4
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture
        [0]: 'YUYV' (YUYV 4:2:2)
                Size: Discrete 1280x720
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.133s (7.500 fps)
                        Interval: Discrete 0.200s (5.000 fps)
        [1]: 'MJPG' (Motion-JPEG, compressed)
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.042s (24.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.133s (7.500 fps)
                        Interval: Discrete 0.200s (5.000 fps)
```

USB camera providing JPEG (faster fps) compressed images (note: could be also H.264/AVC or H.265/HEVC compression)

Camera Image capture

```
./01_camera_capture/
├── README.txt
├── gstreamer
│   ├── README.txt
│   ├── capture_frame_jpeg.sh
│   ├── capture_frame_raw.sh
│   └── check_formats.sh
└── python
    └── capture_frame.py
```

For this example you can only use MIPI-CSI camera.
Camera will show up as /dev/video3

#2 capture few images in jpeg

```
gst-launch-1.0 v4l2src device=/dev/video3 num-buffers=3 !
'video/x-raw,width=1920,height=1080,format=YUY2,framerate=30/1' !
videoconvert  ! jpegenc ! multifilesink index=0 location=frame%04d.jpg sync=true async=false
```

#3 capture few images in raw
(use https://github.com/IENT/YUView [win] or Vooya [linux] to visualize)

```
gst-launch-1.0 v4l2src device=/dev/video3 num-buffers=3 ! 'video/x-
raw,width=1920,height=1080,format=YUY2,framerate=30/1' ! multifilesink index=0
location=frame%04d.raw sync=true async=false
```

Note: for imx8 we have “hw accelerated plugins” documented in “Gstreamer User Guide”

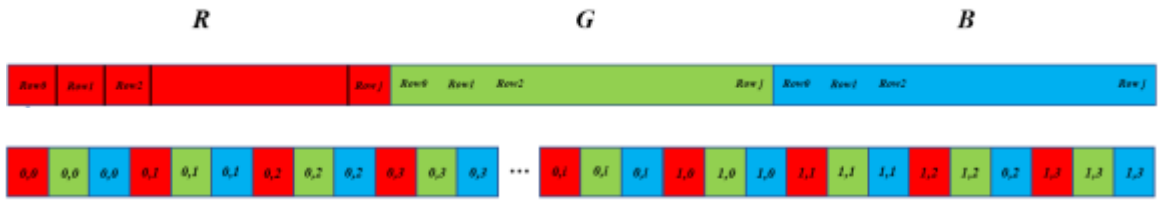
https://community.nxp.com/pwmxy87654/attachments/pwmxy87654/imx-processors%40tkb/15/2/i.MX8GStreamerUserGuide.pdf
https://developer.ridgerun.com/wiki/index.php/IMX8/Multimedia/GStreamer_Support/Hardware-accelerated_plugins

Camera Image capture

```
./01_camera_capture/
├── README.txt
├── gstreamer
│   ├── README.txt
│   ├── capture_frame_jpeg.sh
│   ├── capture_frame_raw.sh
│   └── check_formats.sh
└── python
    └── capture_frame.py
```

#2 capture few images in jpeg

```
from time import time import cv2

# Create a new VideoCapture object, using "/dev/video3"
cam = cv2.VideoCapture(3)

# Keep looping, save every N seconds (N=3)
while True:
      # Get the current time, increase delta and update
      the previous variable
      current = time()
      delta += current - previous
      previous = current

      # Check if 3 (or some other value) seconds passed
      if delta > 3:
            # Operations on image # Reset the time
            counter
            delta = 0

            # save and keep streaming
            _, img = cam.read()

            frame_num=frame_num+1
            filename="frame"+str(frame_num)+".jpg"
            print("save frame: {}".format(frame_num))
            cv2.imwrite(filename, img)
```



Planar or Interleaved format: OpenCV is BGR (Pillow is RGB)
The order of color is BGR (blue, green, red). The OpenCV function imwrite() that saves an image assumes that the order of colors is BGR, so it is saved as a correct image.

https://note.nkmk.me/en/python-opencv-bgr-rgb-cvtcolor/

https://medium.com/@sue.sk.guo/opencv-color-in-bgr-order-you-must-know-53470396d18c

Hailo: model zoo and model explorer

https://github.com/hailo-ai/hailo_model_zoo

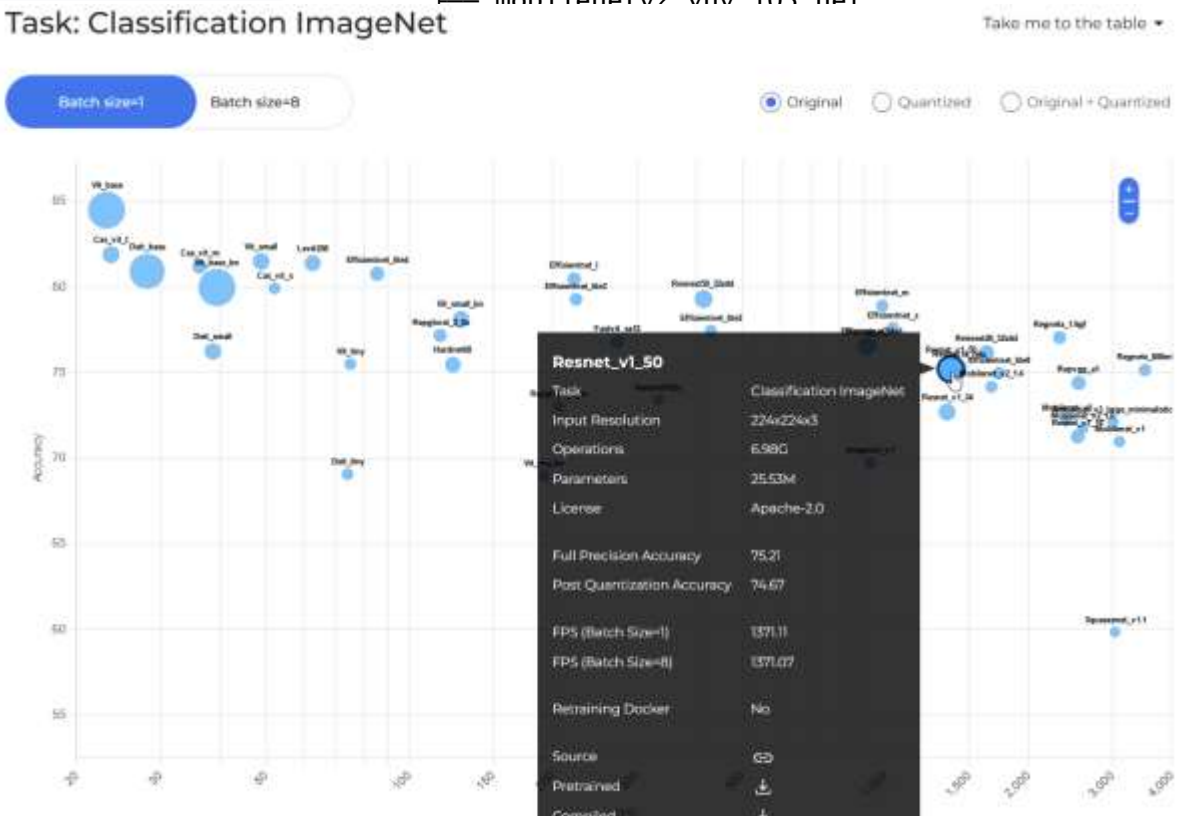
```
./02_hailortcli/
├── 00_see_model_zoo_hefs.txt
├── 01_hef_parse.txt
├── 02_hef_performance.txt
├── 03_hef_latency.txt
├── README.txt
├── docs
│   ├── DC-SPP-YOLO.pdf
│   ├── hailo_dataflow_compiler_v3.29.0_user_guide.pdf
│   ├── hailo_model_zoo_v2.13.0.pdf
│   ├── hailo_tappas_3.30.0_user_guide.pdf
│   └── hailort_4.19.0_user_guide.pdf
└── resources
    ├── mobilenetv2_yuy_105.hef
```

Public Pre-Trained Models

Here, we give the full list of publicly pre-trained models supported by the Hailo Model Zoo.

ImageNet

Network Name	Accuracy (top1)	HW Accuracy	FPS (Batch Size=1)	FPS (Batch Size=8)	Input Resolution (HxWxC)	Params (M)	OPS (G)
inception_v1	69.74	69.52	928	928	224x224x3	6.62	3
mobilenet_v1	70.97	70.3	3100	3100	224x224x3	4.22	1.14
mobilenet_v2_1.0	71.78	70.95	2596	2596	224x224x3	3.49	0.62
mobilenet_v2_1.4	74.18	73.18	1668	1668	224x224x3	6.09	1.18
mobilenet_v3	72.21	71.8	2400	2401	224x224x3	4.07	2
mobilenet_v3_large_minimalistic	72.12	70.61	3005	3005	224x224x3	3.91	0.42
regnetx_1.6gf	77.05	76.71	2321	2321	224x224x3	9.17	3.22
regnetx_800mf	75.16	74.83	3505	3505	224x224x3	7.24	1.6
repghost_1_0x	73.03	72.28	205	861	224x224x3	4.1	0.28
repghost_2_0x	77.18	76.91	115	480	224x224x3	9.8	1.04
repvgg_a1	74.4	72.35	2545	2545	224x224x3	12.79	4.7
repvgg_a2	76.52	74.48	911	911	224x224x3	25.5	10.2
resmlp12_relu	75.27	74.87	1429	1429	224x224x3	15.77	6.04
resnet_v1_18	71.27	71.09	2533	2533	224x224x3	11.68	3.64
resnet_v1_34	72.7	72.27	1346	1346	224x224x3	21.79	7.34
resnet_v1_50	75.21	74.67	1371	1371	224x224x3	25.53	6.98



Hailo: hailortcli to “parse” a .hef model

From HailoRT User Manual, cap 7:

7.2. Parse-HEF Tool

The `parse-hef` tool is used for parsing an existing HEF file and getting information about its content, such as input/output formats and type, singe/multi context, and so on.

```
./02_hailortcli/
├── 00_see_model_zoo_hefs.txt
├── 01_hef_parse.txt
├── 02_hef_performance.txt
├── 03_hef_latency.txt
├── README.txt
├── docs
│   ├── DC-SPP-YOLO.pdf
│   ├── hailo_dataflow_compiler_v3.29.0_user_guide.pdf
│   ├── hailo_model_zoo_v2.13.0.pdf
│   ├── hailo_tappas_3.30.0_user_guide.pdf
│   └── hailort_4.19.0_user_guide.pdf
└── resources
    ├── mobilenetv2_yuv_105.hef
    ├── yolov5m.hef
    ├── yolov5m_wo_spp.hef
    ├── yolov5m_yuv.hef
    ├── yolov7.hef
    ├── yolov8m.hef
    ├── yolov8s.hef
    └── yolox_tiny.hef
```

```
# 01: yolov5m, SINGLE CONTEXT, prev toolchain

> hailortcli parse-hef ./resources/yolov5m.hef

Architecture HEF was compiled for: HAILO8
Network group name: yolov5m, Single Context
    Network name: yolov5m/yolov5m
        VStream infos:
            Input  yolov5m/input_layer1 UINT8, NHWC(640x640x3)
            Output yolov5m/conv94 UINT8, FCR(20x20x255)
            Output yolov5m/conv85 UINT8, FCR(40x40x255)
            Output yolov5m/conv75 UINT8, FCR(80x80x255)
```

```
# 02: yolov5m SINGLE CONTEXT, YUY2 (YCbCr 4:2:2) input

> hailortcli parse-hef ./resources/yolov5m_yuv.hef

Architecture HEF was compiled for: HAILO8
Network group name: yolov5m_yuv, Single Context
    Network name: yolov5m_yuv/yolov5m_yuv
        VStream infos:
            Input  yolov5m_yuv/input_layer1 UINT8, YUY2(720x1280x2)
            Output yolov5m_yuv/conv93 UINT8, NHWC(20x20x255)
            Output yolov5m_yuv/conv83 UINT8, NHWC(40x40x255)
            Output yolov5m_yuv/conv73 UINT8, NHWC(80x80x255)
```

```
# custom mobilenet_v2, dual input stream

> hailortcli parse-hef ./resources/mobilenetv2_yuv_105.hef

Architecture HEF was compiled for: HAILO8
Network group name: mobilenetv2_yuv_105, Single Context
    Network name: mobilenetv2_yuv_105/mobilenetv2_yuv_105
        VStream infos:
            Input  mobilenetv2_yuv_105/input_layer1 UINT8, NHWC(224x224x1)
            Input  mobilenetv2_yuv_105/input_layer2 UINT8, NHWC(112x112x2)
            Output mobilenetv2_yuv_105/fc1 UINT8, NC(1000)
```

Hailo: hailortcli to measure performance

From HailoRT User Manual, cap 7:

7.3. Inference

The `run` tool is used for inference. It loads a given HEF to the device and then sends and receives data. While running, it performs several measurements such as FPS and power. It can also control the running mode:

- `streaming` - Streaming inference with random data.
- `hw-only` - Similar to the full streaming mode, but skips pre-infer and post-infer steps on host.

The `--batch-size` option of this tool sets the batch size in case of context switch, which means after how many frames a context switch will take place. It should not be confused with the `--frames-count` option that sets the total frame count to be sent to the device.

Note: `--frames-count` sets the total frame count, and must be dividable by `--batch-size`.

```
./02_hailortcli/
├── 00_see_model_zoo_hefs.txt
├── 01_hef_parse.txt
├── 02_hef_performance.txt
├── 03_hef_latency.txt
├── README.txt
├── docs
│   ├── DC-SPP-YOLO.pdf
│   ├── hailo_dataflow_compiler_v3.29.0_user_guide.pdf
│   ├── hailo_model_zoo_v2.13.0.pdf
│   ├── hailo_tappas_3.30.0_user_guide.pdf
│   └── hailort_4.19.0_user_guide.pdf
└── resources
    ├── mobilenetv2_yuv_105.hef
    ├── yolov5m.hef
    ├── yolov5m_wo_spp.hef
    ├── yolov5m_yuv.hef
    ├── yolov7.hef
    ├── yolov8m.hef
    ├── yolov8s.hef
    └── yolox_tiny.hef
```

```
# compare RAW performances (multi-thread input)

# YOLOV5m, 640x640, RGB 4:4:4
> hailortcli run ./resources/yolov5m.hef

Running streaming inference (./resources/yolov5m.hef):
  Transform data: true
    Type:      auto
    Quantized: true
Network yolov5m/yolov5m: 100% | 783 | FPS: 156.39

Inference result:
 Network group: yolov5m
    Frames count: 783
    FPS: 156.40
    Send Rate: 1537.48 Mbit/s
    Recv Rate: 2690.59 Mbit/s
```

```
# YOLOV5m, 1280x720P, yuv 4:2:0
> run ./resources/yolov5m_yuv.hef

Running streaming inference (./resources/yolov5m_yuv.hef):
  Transform data: true
    Type:      auto
    Quantized: true
Network yolov5m_yuv/yolov5m_yuv: 100% | 514 | FPS: 102.66
Inference result:
 Network group: yolov5m_yuv
    Frames count: 514
    FPS: 102.66
    Send Rate: 1513.85 Mbit/s
    Recv Rate: 1776.01 Mbit/s
```

Hailo: hailortcli to measure latency

```
./02_hailortcli/
├── 00_see_model_zoo_hefs.txt
├── 01_hef_parse.txt
├── 02_hef_performance.txt
├── 03_hef_latency.txt
├── README.txt
├── docs
│   ├── DC-SPP-YOLO.pdf
│   ├── hailo_dataflow_compiler_v3.29.0_user_guide.pdf
│   ├── hailo_model_zoo_v2.13.0.pdf
│   ├── hailo_tappas_3.30.0_user_guide.pdf
│   └── hailort_4.19.0_user_guide.pdf
└── resources
    ├── mobilenetv2_yuv_105.hef
    ├── yolov5m.hef
    ├── yolov5m_wo_spp.hef
    ├── yolov5m_yuv.hef
    ├── yolov7.hef
    ├── yolov8m.hef
    ├── yolov8s.hef
    └── yolox_tiny.hef
```

From HailoRT User Manual, cap 7:

7.3. Inference

The `run` tool is used for inference. It loads a given HEF to the device and then sends and receives data. While running, it performs several measurements such as FPS and power. It can also control the running mode:

- `streaming` - Streaming inference with random data.
- `hw-only` - Similar to the full streaming mode, but skips pre-infer and post-infer steps on host.

Note: When using `--measure-latency`, the `HW Latency` value measured is the time period between when the frame started to be sent to device until the frame was received on the host.

When using `--measure-latency --measure-overall-latency`, the value measure is the time period between when the application started to send the frame, and when the application finished receiving the frame (including format reordering, quantization if needed, etc.).

For most use cases and benchmarks, it is recommended to use `--measure-latency --measure-overall-latency`.

```
# YOLOV5m, 640x640, RGB
>run2 --measure-latency set-net ./resources/yolov5m.hef

[HailoRT CLI] [warning] Measuring latency; frames are sent one at a time and
FPS will not be measured
[====================>] 100% 00:00:00
yolov5m: hw latency: 17.02 ms

# YOLOV5m, 720P, YUY2 (single input)
> hailortcli run2 --measure-latency set-net ./resources/yolov5m_yuv.hef

[HailoRT CLI] [warning] Measuring latency; frames are sent one at a time and
FPS will not be measured
[====================>] 100% 00:00:00
yolov5m_yuv: hw latency: 21.67 ms
```

NOTE: check CPU governor for bigger models. Pcie bandwidth and CPU load will be affected by the CPU clock.

Object Detection with Hailo on Astrial

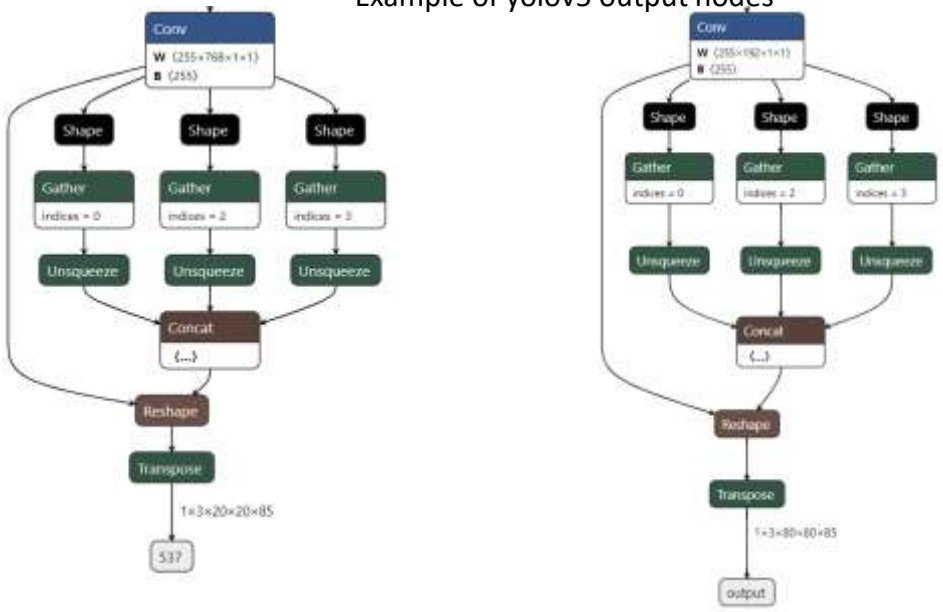
```
./03_object_detection/
├── cpp
├── gstreamer
├── hailo-rpi5-yolov8
└── python
```

DataFlowCompiler:
Hailo8 is able to run in hardware most of the CNN structures “from convolution to convolution” layer. Most of the time the “tail” of common neural nets is a serial process with a lot of non-parallel functions.

For this reason there is always a “CPU” post processing in addition to the usual data preprocessing (NOTE: data normalization can be done on the hw)

The TAPPAS library does provide post-proc code for most of the supported CNN
https://github.com/hailo-ai/tappas/tree/master/core/hailo/libs

Example of yolov5 output nodes



TAPPAS Framework
TAPPAS is a GStreamer based library of plug-ins. It enables using a Hailo devices within gstreamer pipelines to create intelligent video processing applications

https://github.com/hailo-ai/tappas/blob/master/docs/TAPPAS_architecture.rst

Hailo GStreamer Elements
HailoNet - Element for sending and reciveing data from Hailo-8 chip
HailoFilter - Element that enables the user to apply a postprocess or drawing operation to a frame and its tensors
HailoPython - Element that enables the user to apply a postprocess or drawing operation to a frame and its tensors via python.
HailoMuxer - Muxer element used for Multi-Hailo-8 setups
HailoDeviceStats - Hailodevicestats is an element that samples power and temperature
HailoAggregator - HailoAggregator is an element designed for applications with cascading networks. It has 2 sink pads and 1 source
HailoCropper - HailoCropper is an element designed for applications with cascading networks. It has 1 sink and 2 sources
HailoTileAggregator - HailoTileAggregator is an element designed for applications with tiles. It has 2 sink pads and 1 source
HailoTileCropper - HailoCropper is an element designed for applications with tiles. It has 1 sink and 2 sources
.....

Object Detection with Hailo on Astrial

#1 gstreamer detection from .jpeg file (use resources/bus.jpg)

- Run the detection gstreamer script
- Will generate output.jpeg in the same folder
- Copy (SCP) the file on your local computer to visualize on screen

```
./03_object_detection/
├── cpp
├── gstreamer
│   ├── detection_camera_MIPI-CSI.sh
│   ├── detection_camera_USB.sh
│   ├── detection_file_jpeg.sh
│   ├── detection_file_video.sh
│   └── resources
├── hailo-rpi5-yolov8
└── python
```

```
> sudo ./detection_jpeg.sh -i ./resources/bus.jpg

PIPELINE="gst-launch-1.0 \
    filesrc location=$input_source ! jpegdec ! videoconvert ! video/x-raw,format=YUY2,width=1280,height=720 ! \
    queue leaky=downstream max-size-buffers=5 max-size-bytes=0 max-size-time=0 ! \
    synchailonet hef-path=$hef_path ! \
    queue leaky=no max-size-buffers=30 max-size-bytes=0 max-size-time=0 ! \
    hailofilter function-name=$network_name config-path=$json_config_path so-path=$postprocess_so qos=false ! \
    queue leaky=no max-size-buffers=30 max-size-bytes=0 max-size-time=0 ! \
    hailooverlay ! \
    videoconvert ! jpegenc ! filesink location=\"output.jpg\""
```

Where are the components located? See the script code:

```
readonly POSTPROCESS_DIR="/usr/lib/hailo-post-processes"
readonly DEFAULT_POSTPROCESS_SO="$POSTPROCESS_DIR/libyolo_post.so"
readonly DEFAULT_NETWORK_NAME="yolov5"
readonly DEFAULT_HEF_PATH="${RESOURCES_DIR}/${DEFAULT_NETWORK_NAME}m_yuv.hef"
readonly DEFAULT_JSON_CONFIG_PATH="$RESOURCES_DIR/configs/yolov5.json"
```

```
hailortcli parse-hef ./yolov5m_yuv.hef
Architecture HEF was compiled for: HAILO8
Network group name: yolov5m_yuv, Single Context
    Network name: yolov5m_yuv/yolov5m_yuv
        VStream infos:
            Input  yolov5m_yuv/input_layer1 UINT8, YUY2(720x1280x2)
            Output yolov5m_yuv/conv93 UINT8, NHWC(20x20x255)
            Output yolov5m_yuv/conv83 UINT8, NHWC(40x40x255)
            Output yolov5m_yuv/conv73 UINT8, NHWC(80x80x255)
```

Object Detection with Hailo on Astrial

#1 gstreamer detection from .jpeg file (use resources/bus.jpg)

- Run the detection gstreamer script
- Will generate output.jpeg in the same folder
- Copy (SCP) the file on your local computer to visualize on screen

```
./03_object_detection/
├── cpp
├── gstreamer
│   ├── detection_camera_MIPI-CSI.sh
│   ├── detection_camera_USB.sh
│   ├── detection_file_jpeg.sh
│   ├── detection_file_video.sh
│   └── resources
├── hailo-rpi5-yolov8
└── python
```

```
> sudo ./detection_jpeg.sh -i ./resources/bus.jpg

PIPELINE="gst-launch-1.0 \
    filesrc location=$input_source ! jpegdec ! videoconvert ! video/x-raw,format=YUY2,width=1280,height=720 ! \
    queue leaky=downstream max-size-buffers=5 max-size-bytes=0 max-size-time=0 ! \
    synchailonet hef-path=$hef_path ! \
    queue leaky=no max-size-buffers=30 max-size-bytes=0 max-size-time=0 ! \
    hailofilter function-name=$network_name config-path=$json_config_path so-path=$postprocess_so qos=false ! \
    queue leaky=no max-size-buffers=30 max-size-bytes=0 max-size-time=0 ! \
    hailooverlay ! \
    videoconvert ! jpegenc ! filesink location=\"output.jpg\""
```

#2 gstreamer detection from camera input (will generate file .mp4)

Using “vpuenc_h264” and “h264parse” we encode a Elementary Stream video file. QTMux will create the .mp4 container.

```
> sudo ./detection_camera_MIPI-CSI.sh -i /dev/video3

PIPELINE="gst-launch-1.0 -e \
    v4l2src device=$input_source ! "video/x-raw,format=YUY2,width=1280,height=720,framerate=30/1" !\
    queue leaky=downstream max-size-buffers=5 max-size-bytes=0 max-size-time=0 ! \
    synchailonet hef-path=$hef_path ! \
    queue leaky=no max-size-buffers=30 max-size-bytes=0 max-size-time=0 ! \
    hailofilter function-name=$network_name config-path=$json_config_path so-path=$postprocess_so qos=false ! \
    queue leaky=no max-size-buffers=30 max-size-bytes=0 max-size-time=0 ! \
    hailooverlay ! \
    queue leaky=downstream max-size-buffers=5 max-size-bytes=0 max-size-time=0 ! \
    imxvideoconvert_g2d ! vpuenc_h264 ! h264parse ! qtmux ! filesink location=\"output.mp4\""
```

NOTE: “synchailonet” will be deprecated in favour of “hailonet” (this is specific to ARM hosts)

Object Detection with Hailo on Astrial

#1 python detection from .jpeg file (use resources/bus.jpg)

```
./object_detection.py -n ./resources/yolov7.hef -i ./resources/bus.jpg
```

```
./03_object_detection/
├── cpp
├── gstreamer
├── hailo-rpi5-yolov8
└── python
    ├── README.txt
    ├── coco.txt
    ├── object_detection.py
    ├── object_detection_utils.py
    ├── output_images
    ├── requirements.txt
    ├── resources
    ├── runme.sh
    └── utils.py
```

```python
def process_output(
    output_queue: queue.Queue,
    output_path: Path,
    width: int,
    height: int,
    utils: ObjectDetectionUtils
) -> None:
    """
    Process and visualize the output results.

    Args:
        output_queue (queue.Queue): Queue for output results.
        output_path (Path): Path to save the output images.
        width (int): Image width.
        height (int): Image height.
        utils (ObjectDetectionUtils): Utility class for object detection visualization.
    """
    image_id = 0
    while True:
        result = output_queue.get()
        if result is None:
            break  # Exit the loop if sentinel value is received

        processed_image, infer_results = result
        detections = utils.extract_detections(infer_results)

        # Deals with the expanded results from hailort versions < 4.19.0
        if len(infer_results) == 1:
            infer_results = infer_results[0]

        detections = utils.extract_detections(infer_results)
        utils.visualize(
            detections, processed_image, image_id,
            output_path, width, height
        )
        image_id += 1

    output_queue.task_done()  # Indicate that processing is complete
```

```python
def extract_detections(self, input_data: list, threshold: float = 0.5) -> dict:
    """
    Extract detections from the input data.

    Args:
        input_data (list): Raw detections from the model.
        threshold (float): Score threshold for filtering detections. Defaults to 0.5.

    Returns:
        dict: Filtered detection results.
    """
    boxes, scores, classes = [], [], []
    num_detections = 0

    for i, detection in enumerate(input_data):
        if len(detection) == 0:
            continue

        for det in detection:
            bbox, score = det[:4], det[4]

            if score >= threshold:
                boxes.append(bbox)
                scores.append(score)
                classes.append(i)
                num_detections += 1

    return {
        'detection_boxes': boxes,
        'detection_classes': classes,
        'detection_scores': scores,
        'num_detections': num_detections
    }
```

http://www.astrial.ai

Gianluca Filippini
EBV / ML Specialist

Ai@edge in December 2040: in China …

https://edition.cnn.com/2024/08/22/tech/china-drone-delivery-great-wall-intl-hnk/index.html

Unimore - Internation... Qualcomm RB3 Gen 2 ... home - Unimore Adva... Embedded Linux for

CNN Business Markets Tech Media Calculators Videos Watc

Business / Tech

Tourists scaling the Great Wall of China can now get takeout delivered by drone

By Nectar Gan and Hassan Tayir, CNN

3 minute read · Published 4:17 AM EDT, Thu August 22, 2024

A drone carries a package from Chinese food delivery giant Meituan to the Badaling section of the Great Wall in Beijing, China, on August 16, 2024. Huang Liang/Beijing Youth Daily/VCG/Getty Images

Editor's Note: Sign up for CNN's Meanwhile in China newsletter which explores what you need to know about the country's rise and how it impacts the world.

Hong Kong (CNN) — Hungry tourists hiking on the Great Wall of China can now get their lunch delivered – from the air.

https://www.linkedin.com/posts/alvinfsc_china-is-in-2050-activity-7268879963173666816-UaRb

Ai@edge in December 2040: in Cremona.

https://www.cremonaoggi.it/2024/12/04/il-trattore-del-futuro-nei-campi-di-cremona-un-dispositivo-robotico/

CremonaOggi

Cronaca Politica Economia Cultura Spettacolo Sport Cremona allo specchi

...ione con il commissario straordinario · 4 Dic 2024 Morte del prof Bedeschi: per il camionista chiesti otto mesi · 4 Dic 2024 Croce R

CRONACA Oggi alle 18:06

Il trattore del futuro: nei campi di Cremona un dispositivo robotico

CREMONA

Nei campi di via Ca' del Binda a Cremona, ecco l'agricoltura del futuro. Un terreno di due ettari è stato mappato nel pomeriggio con un macchina agricola autonoma, un trattore, senza nessun uomo a bordo, comandato da remoto. Un pezzo di alta ingegneria e che sembra uscito dal futuro e invece è realtà: i lavori sono stati fatti in piena autonomia con un **sistema robotico, ovvero si tratta dispositivi meccatronici, mobili, totalmente autonomi, programmati per raccogliere ed elaborare dati, "prendere decisioni" e compiere operazioni specifiche senza la supervisione di alcun operatore.**

TECHNOLOGY.

PASSION.

EBV.