Embedded Video Systems With Zephyr

Josuah Demangeon, Panoramix Labs, tinyVision.ai

Background

A contractor working essentially with tinyVision.ai


Helion
ENGINEERING VISION
tinyvision.ai

Autoportrait: I am curious about a lot of topics, but only scratch the surface.

You are welcome and invited to dive in depth!

Video Systems

Famous example: home cinema

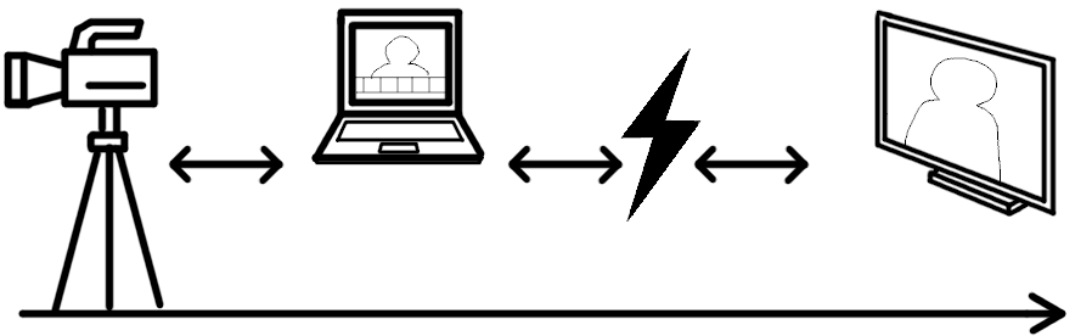

BRAVIA
DONOR
CURRENT WORK UNIT
SONY

Pro Video Camera

Video editing software

Distribution network

Final destination: human eyes

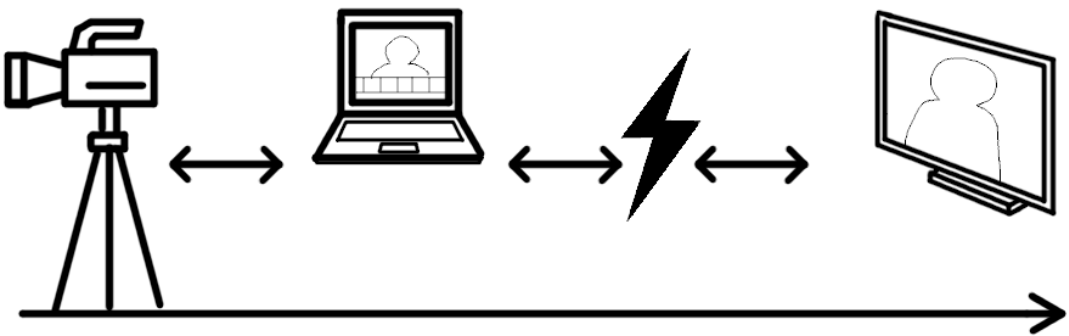


Pro Video Camera

Video editing software

Distribution network

Final destination: human eyes

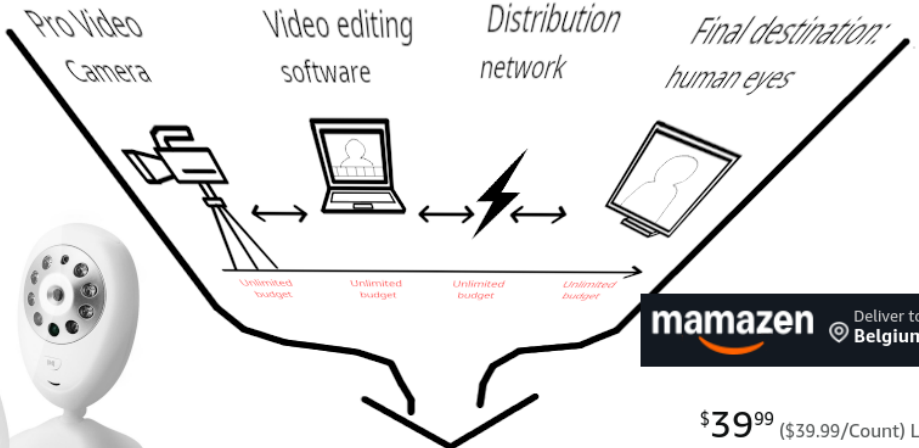


Unlimited budget

Unlimited budget

Unlimited budget

Unlimited budget



Babyphone

mamazen Deliver to Belgium All ba

$39.99 ($39.99/Count) List: $49.99

Embedded Video Systems

Constraints:

→ Cost budget

→ Processing budget

→ Time budget (low-latency, real-time)

Can only work at low-resolution...

Embedded Video Systems

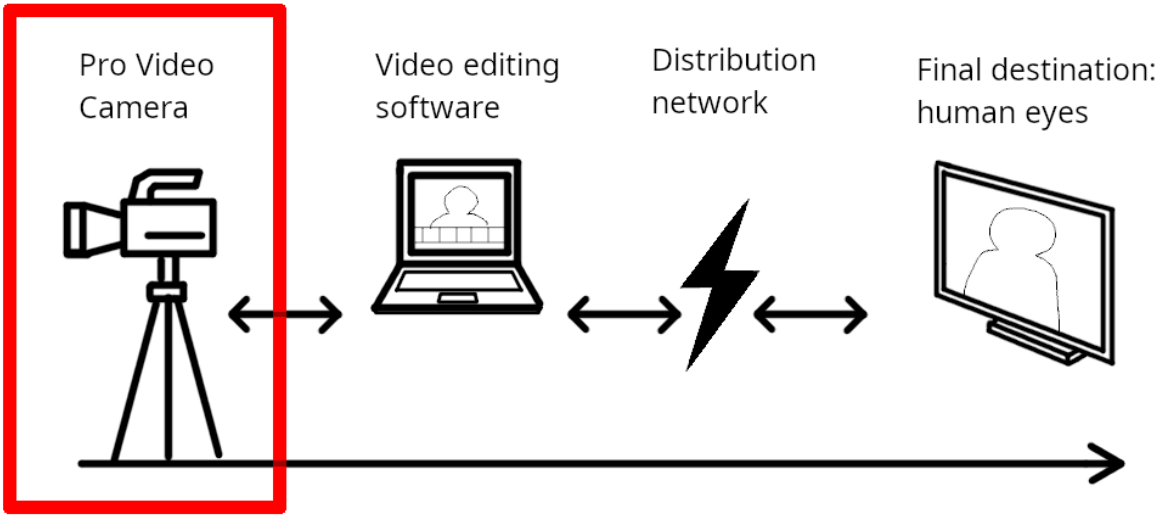
Constraints:

→ Cost budget

→ Processing budget

→ Time budget (low-latency, real-time)

Can only work at low-resolution... <- FALSE!



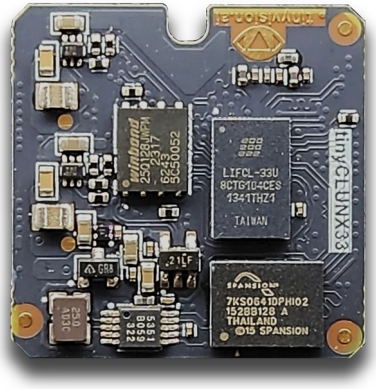
Needs an operating system too!

Embedded is not always low-end.

Embedded Video Systems

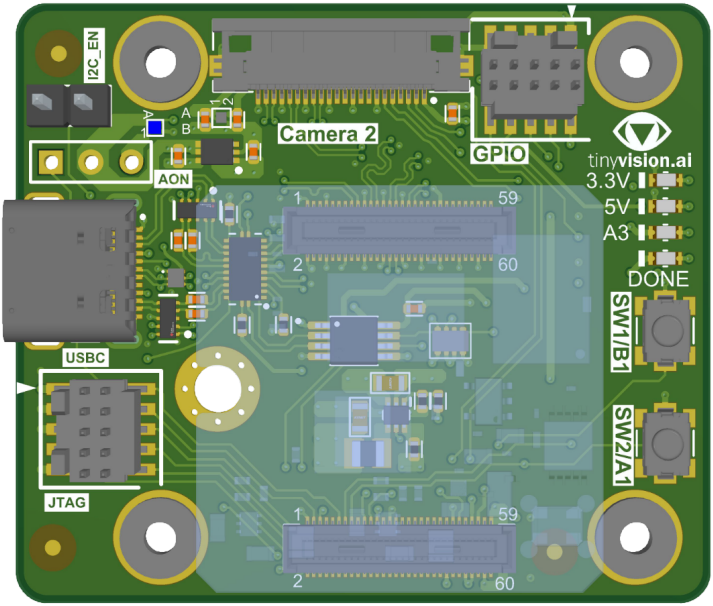
"Why not use an USB camera?"

We are now implementing the USB camera *itself*.



tinyCLUNX33: the heart of an USB 3 webcam.

3.4 Gbit/s under 200 mW


I2C_EN
A
B
1
2
Camera 2
GPIO
tinyvision.ai
3.3V
5V
A3
DONE
AON
1
59
2
60
USBC
SW1/B1
SW2/A1
JTAG
1
59
2
60

Embedded Video Systems

"Why not just a Raspberry Pi?"

→ Power budget

→ Performance

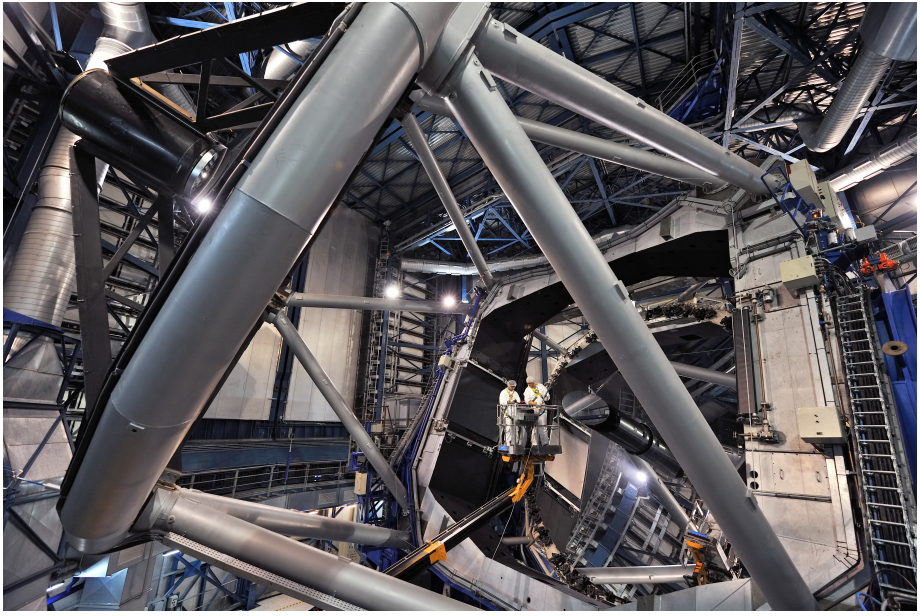
→ Cost

→ Latency

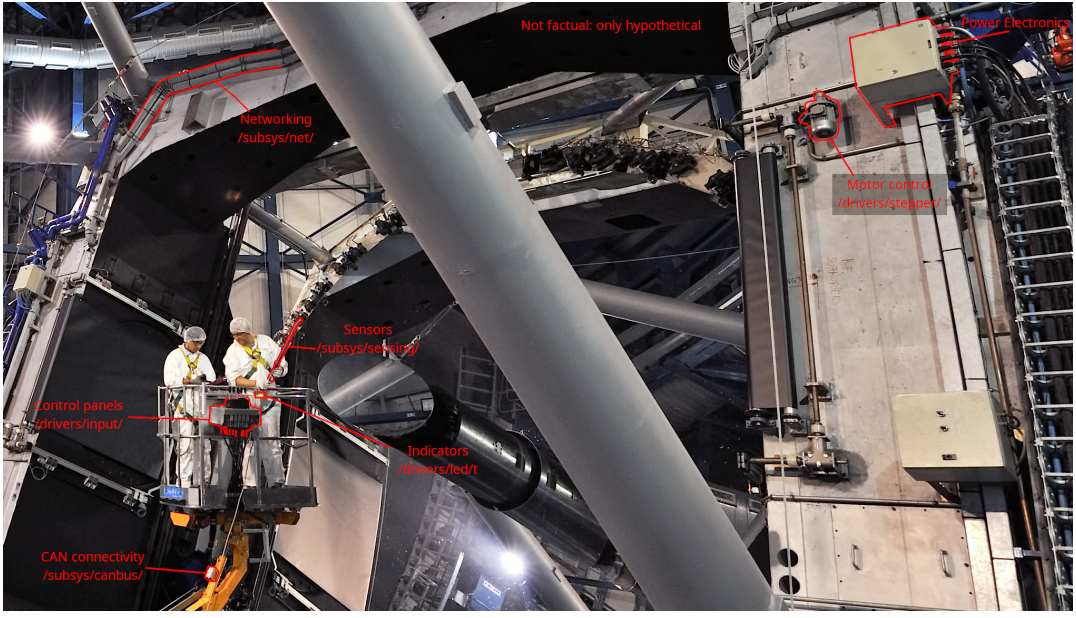

ArduCam

Embedded Video Systems

Can be very large:



We can imagine a lot involved to assist the video function:


Not factual: only hypothetical
Power Electronics
Networking
/subsys/net/
Motor control
/drivers/stepper/
Sensors
/subsys/sensing/
Control panels
/drivers/input/
Indicators
/drivers/led/t
CAN connectivity
/subsys/canbus/

Still there on small embedded systems:

→ Motor for auto-focus ("VCM" motor `#include <zephyr/drivers/video-controls.h>`)

→ I2C communication with other chips (`#include <zephyr/drivers/i2c.h>`)

→ Turning on/off the chip power (Power Management)

Embedded Video Systems

But usually the smaller the better: how to shrink?

Switch from Linux OS → RTOS like Zephyr

Which board running it can do video?

Video-capable boards running Zephyr

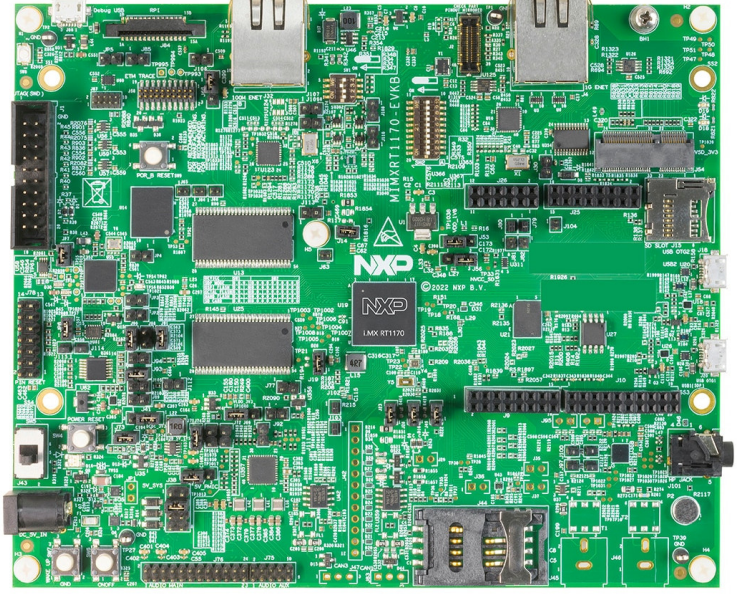
Video-capable means DVP (a.k.a. parallel port) or MIPI support.

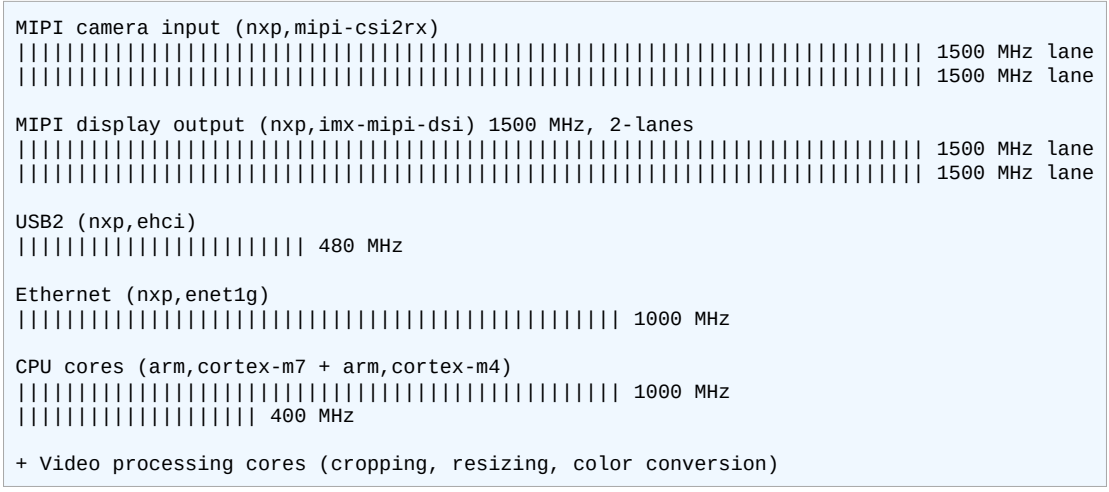
How small can we shrink (size, price, heat, power usage)?

i.MX RT1170

Cortex-M7 (small-medium) running at 1 GHz.

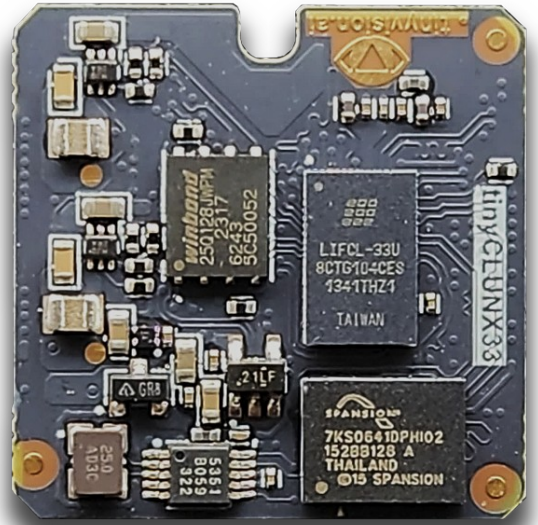
A fast CPU is good to reduce RAM usage: transmit *more often* rather than *more at once*.

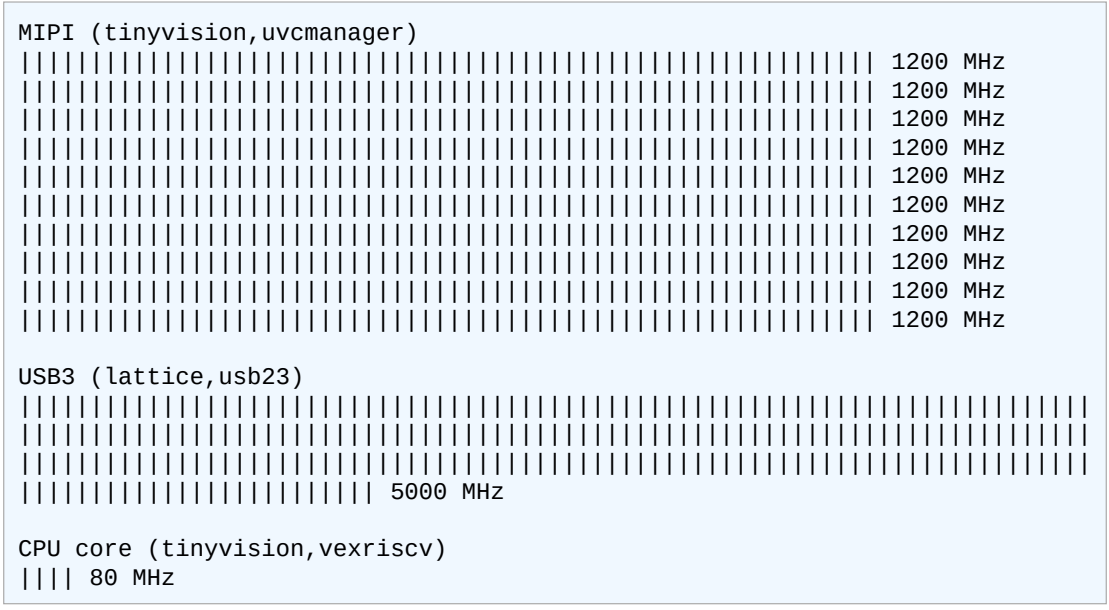

MIMXRT1170-EVKB
NXP
i.MX RT1170
©2022 NXP B.V.


MIPI camera input (nxp,mipi-csi2rx)
1500 MHz lane
1500 MHz lane
MIPI display output (nxp,imx-mipi-dsi) 1500 MHz, 2-lanes
1500 MHz lane
1500 MHz lane
USB2 (nxp,ehci)
480 MHz
Ethernet (nxp,enet1g)
1000 MHz
CPU cores (arm,cortex-m7 + arm,cortex-m4)
1000 MHz
400 MHz
+ Video processing cores (cropping, resizing, color conversion)

tinyVision.ai tinyCLUNX33

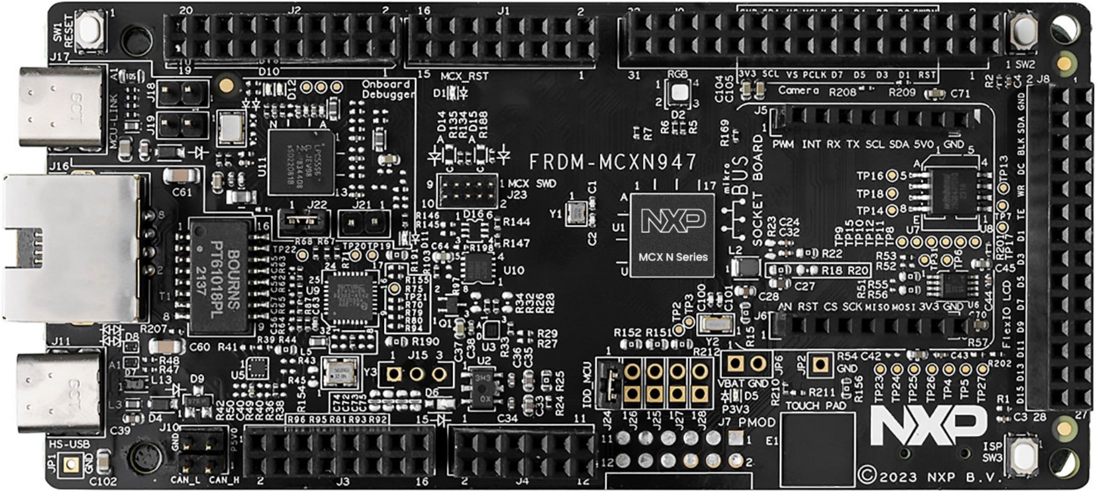
A system specialized for MIPI to USB3 camera systems. An FPGA: very slow CPU and needs to "build your own video cores". Not upstream yet.


winbond
25Q128JWPM
2317
6243
5C50052
LIFCL-33U
8CTG104CES
1341THZ1
TAIWAN
.21UF
SPANSION
7KS0641DPHI02
15288128 A
THAILAND
©15 SPANSION
tinyCLUNX33


MIPI (tinyvision,uvcmanager)
1200 MHz
1200 MHz
1200 MHz
1200 MHz
1200 MHz
1200 MHz
1200 MHz
1200 MHz
1200 MHz
1200 MHz
USB3 (lattice,usb23)
5000 MHz
CPU core (tinyvision,vexriscv)
80 MHz

FRDM-MCXN947

Dual Cortex-M33 (small) system with peripherals usually only found on larger Linux-capable devices: "do more with less"

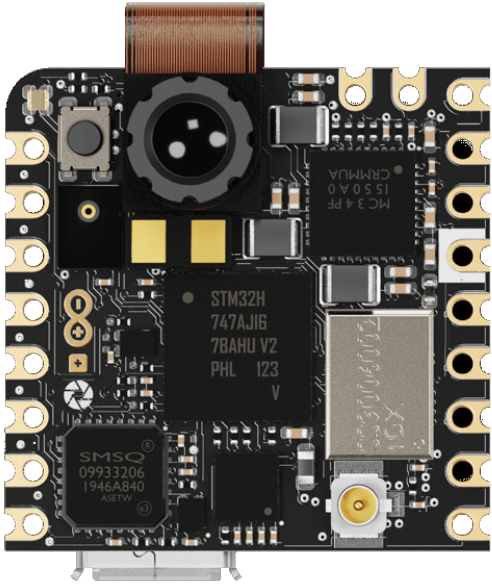

FRDM-MCXN947
NXP
MCX N Series
Onboard Debugger
MCU-LINK
SW1 RESET
MCX_RST
MCX SWD
RGB
Camera
3V3 SCL VS PCLK D7 D5 D3 D1 RST
mikro BUS SOCKET BOARD
PWM INT RX TX SCL SDA 5V0 GND
AN RST CS SCK MISO MOSI 3V3 GND
FlexIO LCD
BOURNS PT61018PL 2137
IDD_MCU
VBAT GND
TOUCH PAD
PMOD
HS-USB
CAN_L CAN_H
ISP SW3
©2023 NXP B.V.

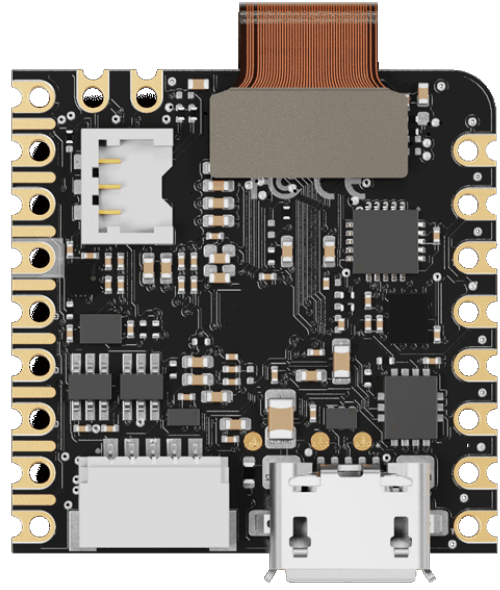
```
DVP camera input (nxp,video-smartdma)
|||||||| |||||||| |||||||| |||||||| |||||||| |||||||| |||||||| 8 pins (16 max), 150 MHz each

USB2 (nxp,ehci)
|||||||||||||||||||||||| 480 MHz

Ethernet (nxp,enet-qos)
||||| 100 MHz

CPU cores (arm,cortex-m33f)
|||||||| 150 MHz
|||||||| 150 MHz

+ eIQ NPU on-board for A.I. inference (release planned 2025 [1])
```

[1]: eIQ application note

Arduino Nicla Vision (STM32H747)

All-in-one board with IMU, microphone, 2 MP camera built-in, fast USB.

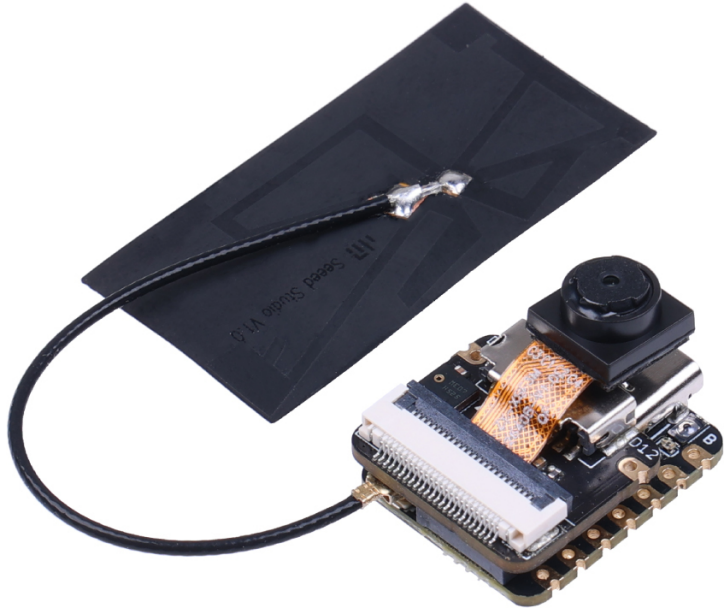

STM32H
747AJI6
7BAHU V2
PHL 123
V
SMSQ
09933206
1946A840
ASETW



```
DVP camera input (st,stm32-dcmi)
|||| |||| |||| |||| |||| |||| |||| |||| 8-pins (14 max), 80 MHz each

USB2 (st,stm32-otghs)
|||||||||||||||||||||| 480 MHz

Wi-Fi (murata,1dx)
|||| 65 Mbit/s

CPU cores (arm,cortex-m7 + arm,cortex-m4)
|||||||||||||||||||||| 480 MHz
|||||||||||| 240 MHz

+ JPEG compression core
+ Video processing operations (cropping, resizing, color conversion)
```

XIAO ESP32S3 Sense

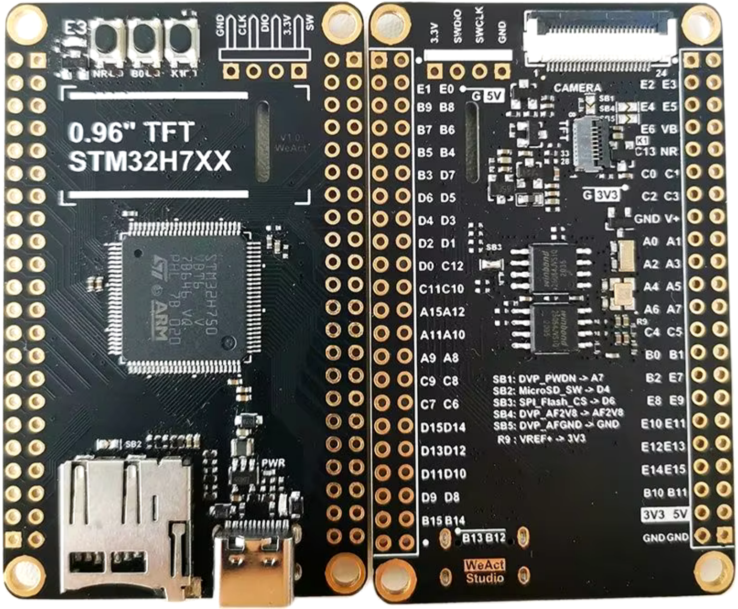
Self-contained board for wireless (WiFi, Bluetooth), coming with a camera and microphone.


Seeed Studio V1.0

```
DVP (espressif,lsd-cam)
|||| |||| |||| |||| |||| |||| |||| |||| 8 pins (16 max) 80 MHz each

Wi-Fi (espressif,esp32-wifi)
|||||||| 150 Mbit/s

CPU core (espressif,xtensa-lx7 + espressif,xtensa-lx7)
|||||||||||| 240 MHz
|||||||||||| 240 MHz
```

WeAct MiniSTM32H7xx

Minimalist approach to a video devboard, comes with a camera and a display and fast USB.


E3
GND
CLK
DIO
3.3V
SW
0.96" TFT
STM32H7XX
PWR
SB2
3.3V
SWDIO
SWCLK
GND
CAMERA
E1 E0
G 5V
B9 B8
B7 B6
B5 B4
B3 D7
D6 D5
D4 D3
D2 D1
D0 C12
C11C10
A15A12
A11A10
A9 A8
C9 C8
C7 C6
D15D14
D13D12
D11D10
D9 D8
B15 B14
B13 B12
E2 E3
E4 E5
E6 VB
C13 NR
C0 C1
G 3V3
C2 C3
GND V+
A0 A1
A2 A3
A4 A5
A6 A7
C4 C5
B0 B1
B2 E7
E8 E9
E10 E11
E12E13
E14E15
B10 B11
3V3 5V
GND GND
SB3
SB1: DVP_PWDN -> A7
SB2: MicroSD_SW -> D4
SB3: SPI_Flash_CS -> D6
SB4: DVP_AF2V8 -> AF2V8
SB5: DVP_AFGND -> GND
R9 : VREF+ -> 3V3
WeAct
Studio

```
DVP camera input (st,stm32-dcmi)
|||| |||| |||| |||| |||| |||| |||| |||| 8 pins (14 max), 80 MHz each

USB2 (st,stm32-otghs / st,stm32-otghs)
|||||||||||||||||||||| 480 MHz
| 12 MHz

Ethernet (st,stm32h7-ethernet)
||||| 100 MHz

CPU core (arm,cortex-m7)
|||||||||||||||||||||| 480 MHz

+ JPEG compression core
+ Video processing operations (cropping, resizing, color conversion)
```

Different software ecosystem

FFmpeg → ???

Gstreamer → ???

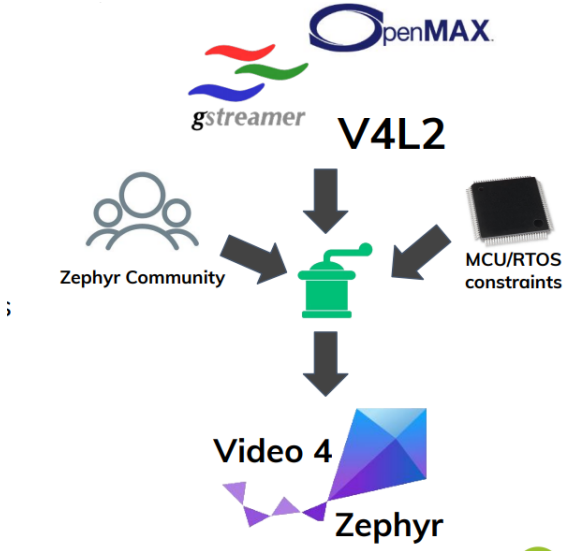
OpenCV → ???

PyTorch → ???

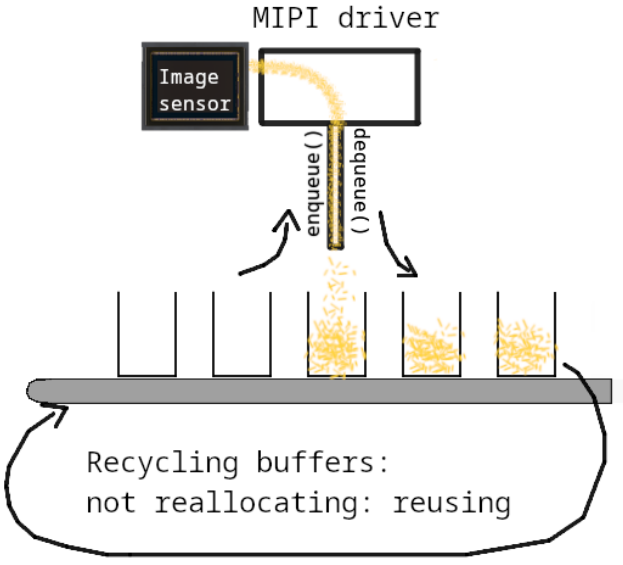
`/dev/video0` → ???

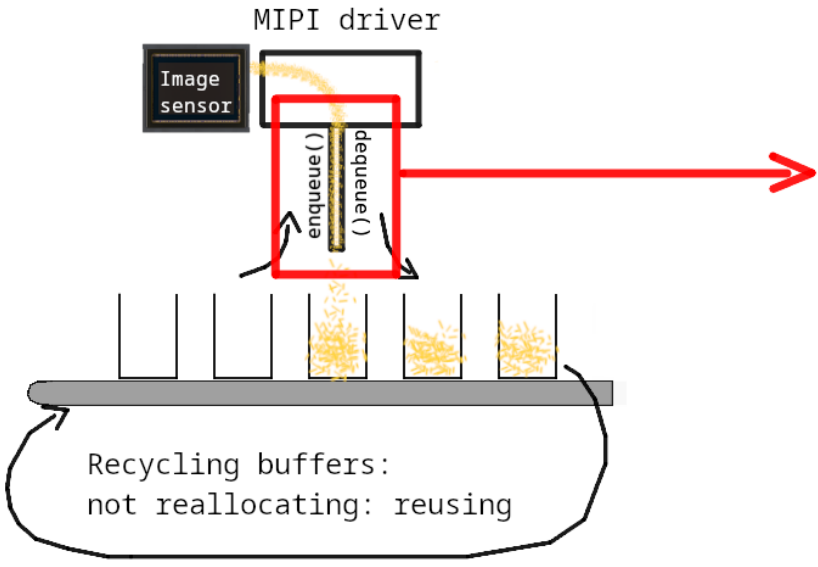
These softwares assume a lot of resources. Everything to reinvent!

Zephyr Video APIs


OpenMAX
gstreamer
V4L2
Zephyr Community
MCU/RTOS
constraints
Video 4
Zephyr

;

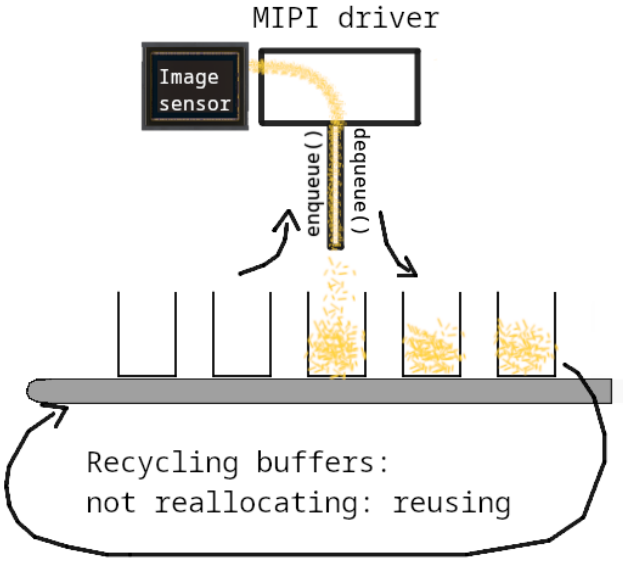

MIPI driver
Image sensor
enqueue()
dequeue()
Recycling buffers:
not reallocating: reusing

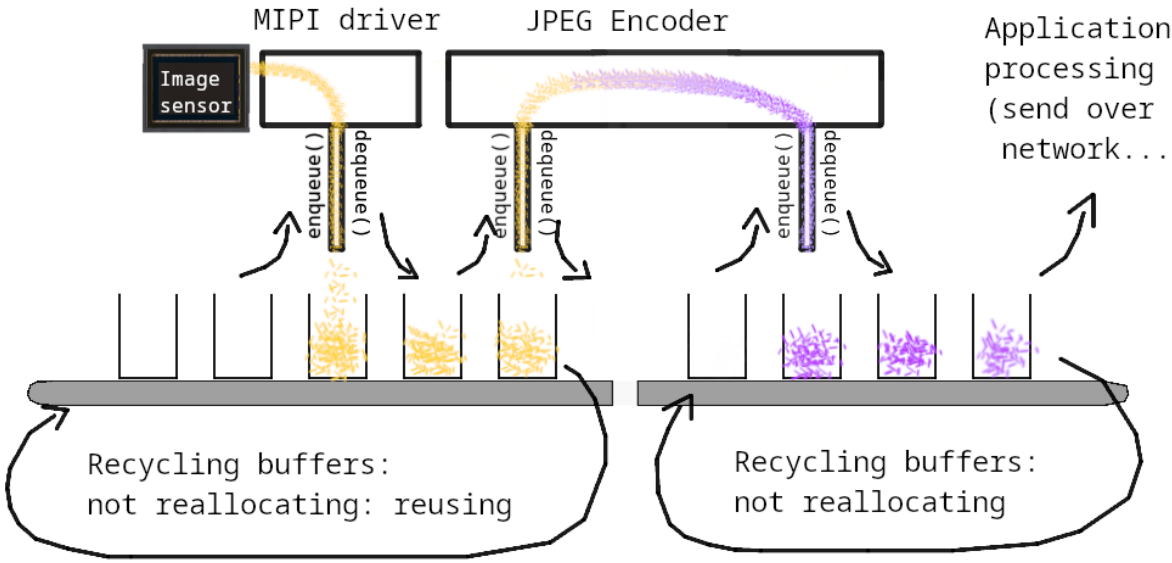


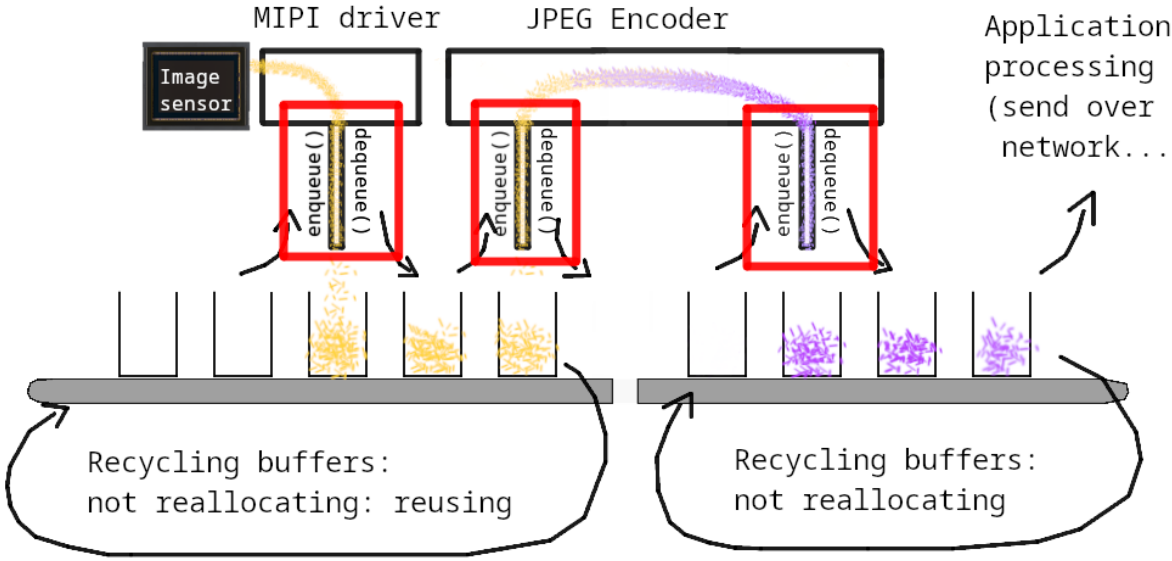
Data API

not implemented
by all drivers

```
/* Error handling ommitted */

video_stream_start(mipi_dev);
vbuf = video_buffer_alloc(1280 * 720 * 3, K_FOREVER);
while (true) {
    video_enqueue(mipi_dev, VIDEO_EP_OUT, vbuf);
    video_dequeue(mipi_dev, VIDEO_EP_OUT, &vbuf, K_FOREVER);
    /* do something with vbuf→data */
}
```


MIPI driver
Image sensor
enqueue()
dequeue()
Recycling buffers:
not reallocating: reusing


MIPI driver
JPEG Encoder
Application processing (send over network...)
Image sensor
enqueue()
dequeue()
enqueue()
dequeue()
enqueue()
dequeue()
Recycling buffers:
not reallocating: reusing
Recycling buffers:
not reallocating


MIPI driver
JPEG Encoder
Application processing (send over network...)
Image sensor
enqueue()
dequeue()
enqueue()
dequeue()
enqueue()
dequeue()
Recycling buffers:
not reallocating: reusing
Recycling buffers:
not reallocating

```
/* Only one buffer of each */
vbuf_raw = video_buffer_alloc(1280 * 720 * 3, K_FOREVER);
vbuf_jpeg = video_buffer_alloc(JPEG_BUF_SIZE, K_FOREVER);
while (true) {
    video_enqueue(mipi_dev, VIDEO_EP_OUT, vbuf_raw);
    video_dequeue(mipi_dev, VIDEO_EP_OUT, &vbuf_raw, K_FOREVER);
    video_enqueue(jpeg_dev, VIDEO_EP_IN, vbuf_raw);
    video_dequeue(jpeg_dev, VIDEO_EP_IN, &vbuf_raw, K_FOREVER);
    video_enqueue(jpeg_dev, VIDEO_EP_OUT, vbuf_jpeg);
    video_dequeue(jpeg_dev, VIDEO_EP_OUT, &vbuf_jpeg, K_FOREVER);
    /* Do something with the JPEG data */
}
```

Nice and simple, but no concurrency: a lot of time spent waiting.

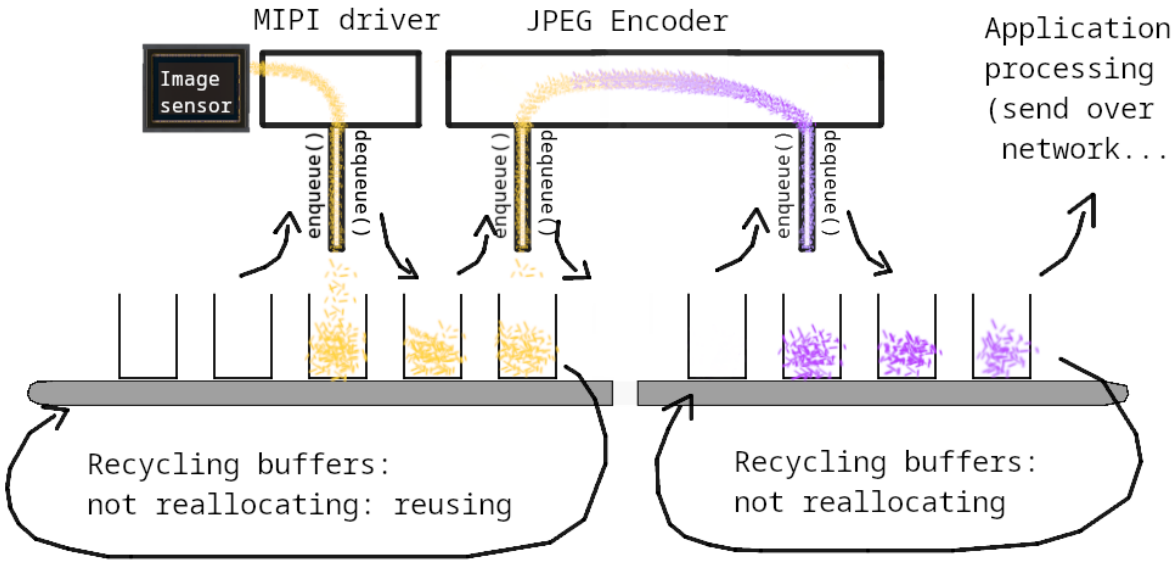
```
for (int i = 0; i < 3; i++) {
    vbuf_raw = video_buffer_alloc(1280 * 720 * 3, K_FOREVER);
    video_enqueue(mipi_dev, VIDEO_EP_OUT, vbuf_raw);
}

for (int i = 0; i < 3; i++) {
    vbuf_jpeg = video_buffer_alloc(JPEG_BUF_SIZE, K_FOREVER);
    video_enqueue(jpeg_dev, VIDEO_EP_OUT, vbuf_jpeg);
}

/* Thread 1: Handle the filled buffers */
while (true) {
    video_dequeue(mipi_dev, VIDEO_EP_OUT, &vbuf_raw, K_FOREVER);
    video_enqueue(jpeg_dev, VIDEO_EP_IN, vbuf_raw);
}
```

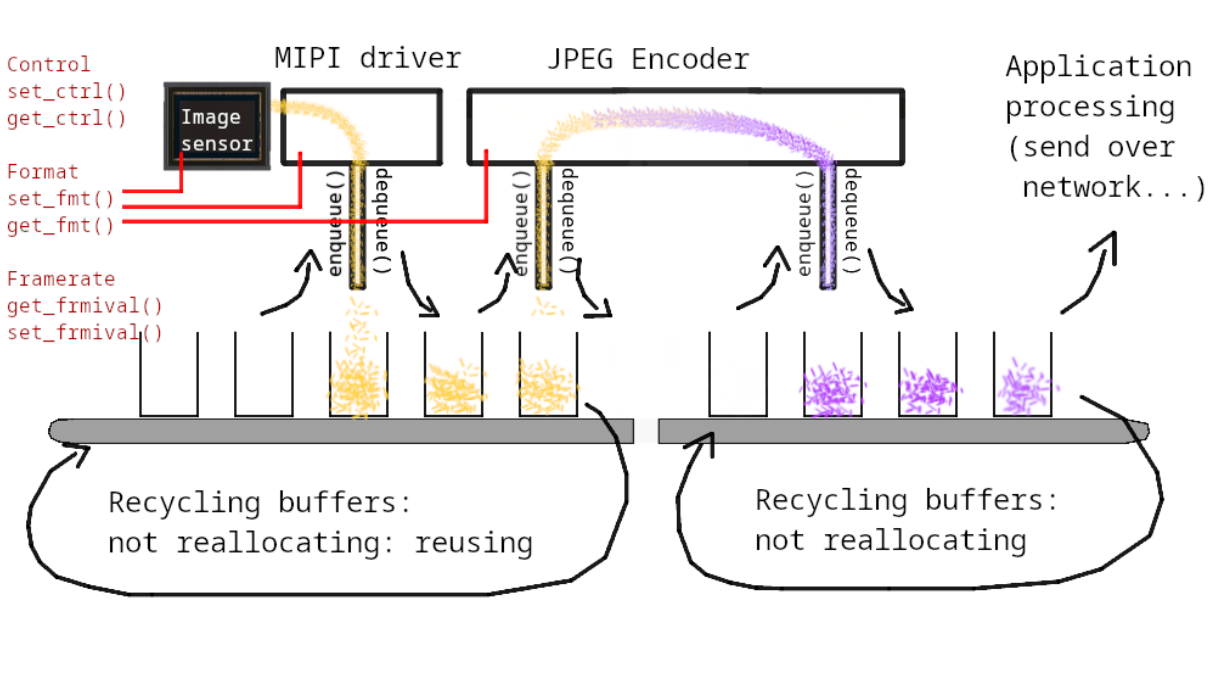
```
/* Thread 2: Handle the emptied buffers */
while (true) {
    video_dequeue(jpeg_dev, VIDEO_EP_IN, &vbuf_raw, K_FOREVER);
    video_enqueue(mipi_dev, VIDEO_EP_OUT, vbuf_raw);
}

/* Thread 3: Handle the JPEG buffers */
while (true) {
    video_dequeue(jpeg_dev, VIDEO_EP_IN, &vbuf_jpeg, K_FOREVER);
    app_use_jpeg_buffer(vbuf_jpeg);
    video_enqueue(jpeg_dev, VIDEO_EP_OUT, &vbuf_jpeg, K_FOREVER);
}
```

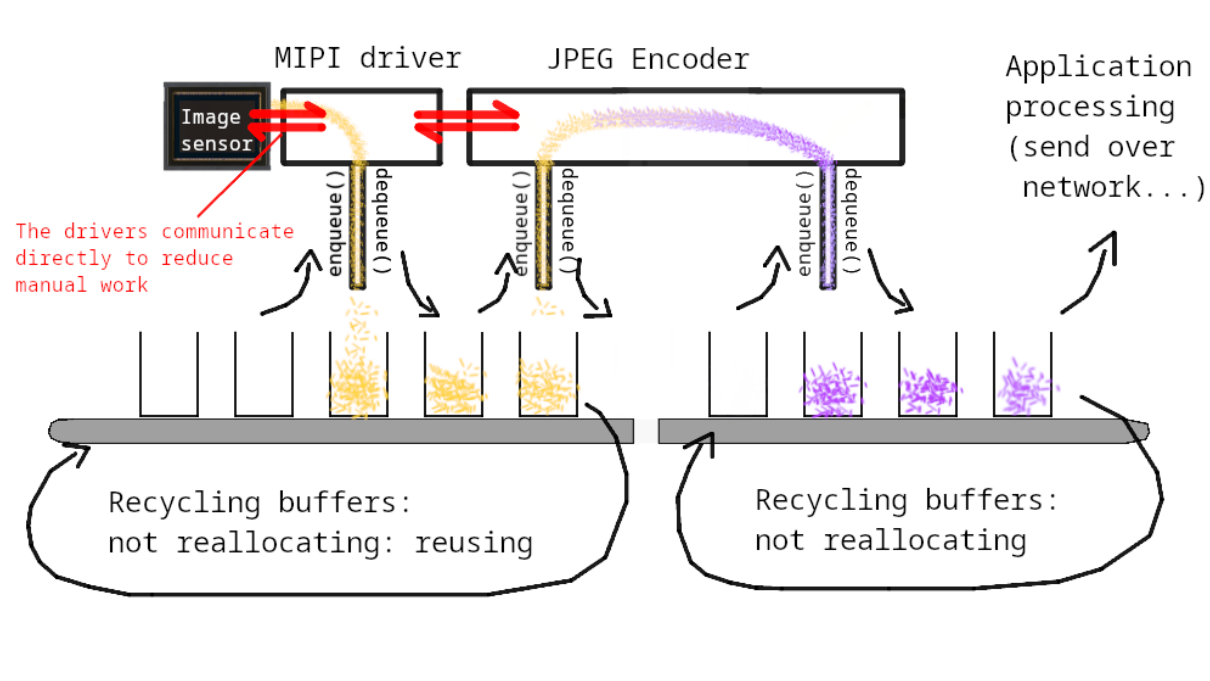
Fully parallel, but consumes a lot of threads: more memory overhead and context switching.

```c
/* First prepare a few buffers */

/* Subscribe to the events */
video_set_signal(mipi_dev, &signal);
video_set_signal(jpeg_dev, &signal);

/* React to the events */
while (k_poll(&events, 1, K_FOREVER) == 0) {
    if (video_dequeue(mipi_dev, VIDEO_EP_OUT, &vbuf_raw, K_NO_WAIT) == 0) {
        video_enqueue(jpeg_dev, VIDEO_EP_IN, vbuf_raw);
    }
    if (video_dequeue(jpeg_dev, VIDEO_EP_IN, &vbuf_raw, K_NO_WAIT) == 0) {
        video_enqueue(mipi_dev, VIDEO_EP_OUT, vbuf_raw);
    }
    if (video_dequeue(jpeg_dev, VIDEO_EP_OUT, &vbuf_jpeg, K_NO_WAIT) == 0) {
        app_use_jpeg_buffer();
        video_enqueue(jpeg_dev, VIDEO_EP_OUT, vbuf_jpeg);
    }
}
```

Good parallelism, single thread (more scalable). Maybe this can be automated for easier pipelines.


MIPI driver
JPEG Encoder
Application processing (send over network...)
Image sensor
enqueue()
dequeue()
enqueue()
dequeue()
enqueue()
dequeue()
Recycling buffers:
not reallocating: reusing
Recycling buffers:
not reallocating


Control
set_ctrl()
get_ctrl()
Format
set_fmt()
get_fmt()
Framerate
get_frmival()
set_frmival()
Image sensor
MIPI driver
JPEG Encoder
Application processing (send over network...)
enqueue()
dequeue()
enqueue()
dequeue()
enqueue()
dequeue()
Recycling buffers:
not reallocating: reusing
Recycling buffers:
not reallocating


MIPI driver
JPEG Encoder
Application processing (send over network...)
Image sensor
The drivers communicate directly to reduce manual work
enqueue()
dequeue()
enqueue()
dequeue()
enqueue()
dequeue()
Recycling buffers:
not reallocating: reusing
Recycling buffers:
not reallocating

```
imx219: imx219@10 {
        compatible = "sony,imx219";
        port {
                imx219_ep_out: endpoint {
                        remote-endpoint-label = "mipi0_ep_in";
                };
        };
};
```

```
mipi0: mipi@b1000010 {
        compatible = "tinyvision,mipi";
        port {
                mipi0_ep_in: endpoint {
                        remote-endpoint-label = "imx219_ep_out";
                };
```

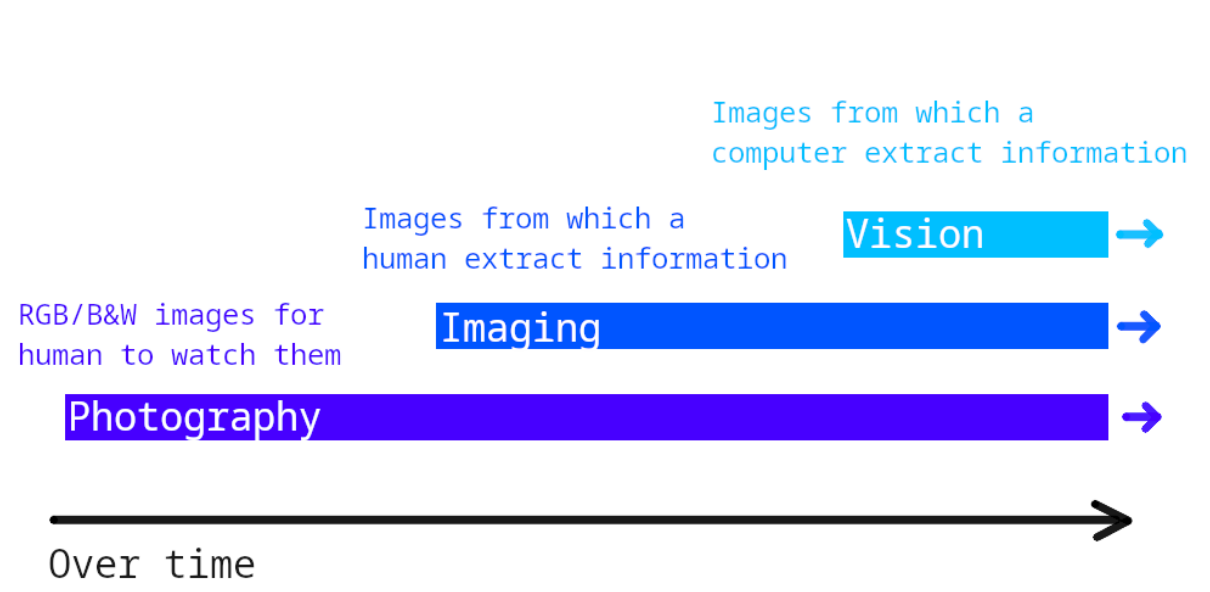
```
                mipi0_ep_out: endpoint {
                        remote-endpoint-label = "jpegenc_ep_in";
                };
        };
};
```

```
jpegenc0: jpegenc@b1000010 {
        compatible = "tinyvision,jpegenc";
        port {
                jpegenc0_ep_in: endpoint {
                        remote-endpoint-label = "mipi0_ep_in";
                };
                /* jpegenc0_ep_out: to the application */
        };
};
```

Next steps: automating more of the pipeline? What would make it more convenient?

Systems doing what?

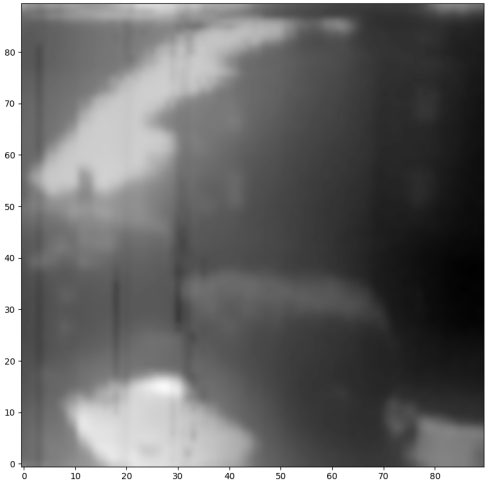
On a journey from Phontons to Video, and how that is used

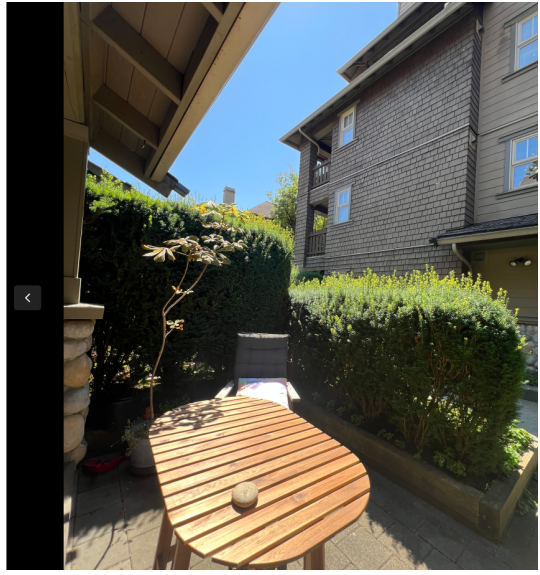


Inspired from a MIPI alliance presentation (2018)

Photodiode

Phenomenon of semiconductors producing voltage when exposed to the light.


0
10
20
30
40
50
60
70
80



Note: photoresistor instead of photodiode here

```
#include <zephyr/drivers/pwm.h> // if using servomotors
#include <zephyr/drivers/stepper.h> // if using stepper motors
#include <zephyr/drivers/adc.h> // measure the light intensity
```

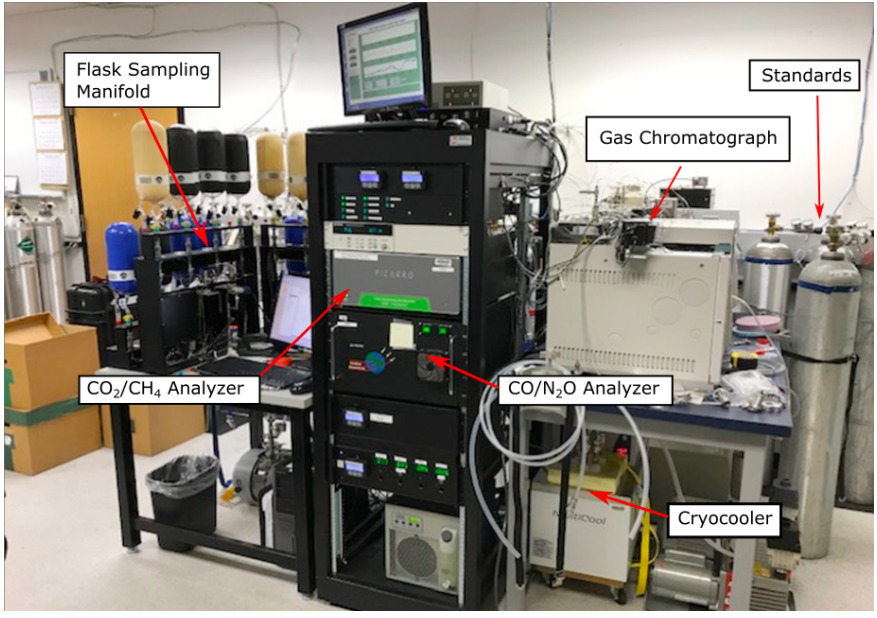
Photons → Photonics

Much more than just video:

→ Gas detection/characterization, i.e. NDIR CO2 sensors

Industrial, safety, medical use-cases.

Since 1958: measuring Earth atmospheric CO2 with "1-pixel image sensors"

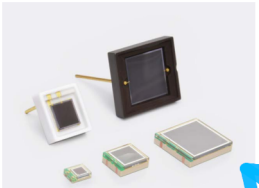

Flask Sampling Manifold
Standards
Gas Chromatograph
CO_2/CH_4 Analyzer
CO/N_2O Analyzer
Cryocooler

→ Biology/medical research, i.e. DNA sequencing

HAMAMATSU
PHOTON IS OUR BUSINESS

MPPC® (multi-pixel photon counter)

S13360 series



MPPCs for precision measurement

MPPC is a type of device called SiPM (silicon photomultipliers). It is a new type of photon counting device that consists of multiple Geiger mode APD (avalanche photodiode) pixels. It is an opto-semiconductor with outstanding photon counting capability and low operating voltage and is immune to the effects of magnetic fields.
The S13360 series are MPPCs for precision measurement. The MPPCs inherits the superb low afterpulse characteristics of previous products and further provide lower crosstalk and lower dark count. They are suitable for precision measurement, such as flow cytometry, DNA sequencer, laser microscope, and fluorescence measurement, that requires low noise characteristics.

Features

- **Reduced crosstalk and dark count (compared to previous products)**
- **Outstanding photon counting capability (outstanding photon detection efficiency versus numbers of incident photons)**
- **Compact**
- **Operates at room temperature**
- **Low voltage (V_{BR}=53 V typ.) operation**

Applications

- **Fluorescence measurement**
- **Laser microscopes**
- **Flow cytometry**
- **DNA sequencers**
- **Environmental analysis**
- **Various academic research**

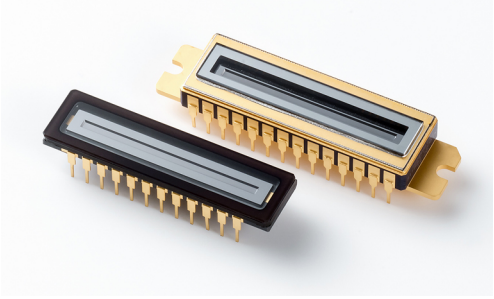
The main purpose of the electronics system associated with the sensor is sensing voltage from the small image sensor pixel-by-pixel.

A single line of pixels

Line sensors: a single line at a time.

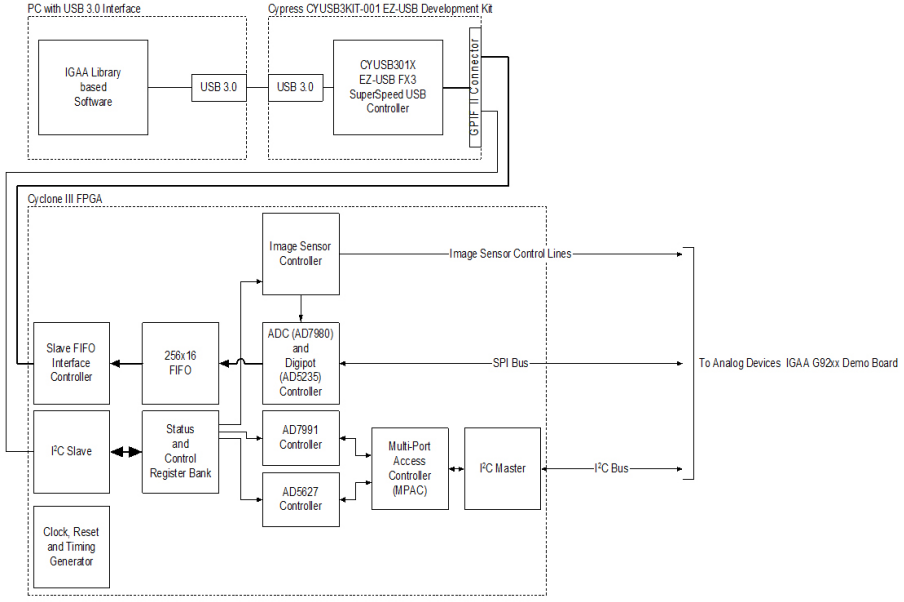
Not digital interface but analog interface: need an ADC to get digital readout.

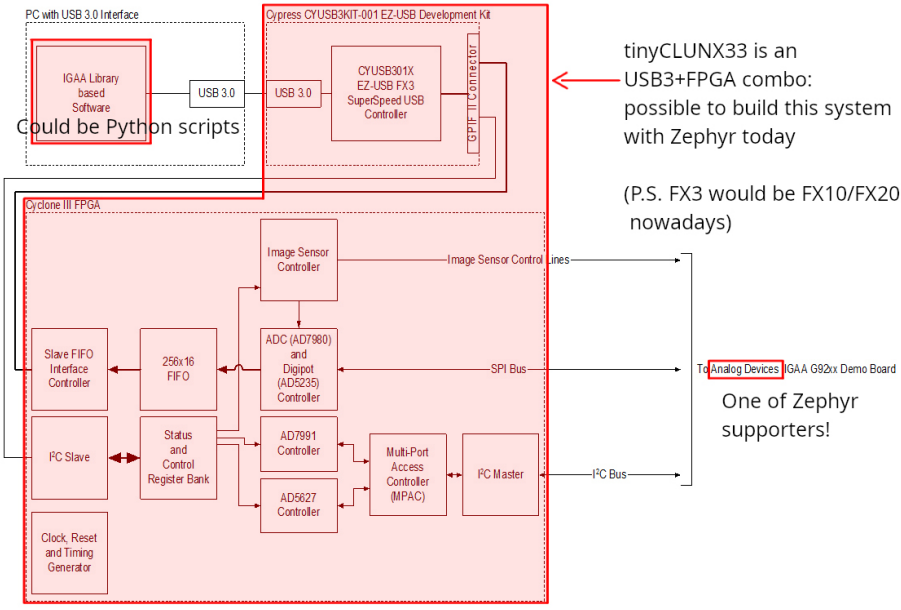
Sensing one pixel at a time, need to go fast!



Requires a fast ADC to handle the input.

Hamamatsu recommends an FPGA (customizable chip) + Analog Devices Analog-Digital-Converter (ADC).


PC with USB 3.0 Interface
IGAA Library based Software
USB 3.0
Cypress CYUSB3KIT-001 EZ-USB Development Kit
USB 3.0
CYUSB301X EZ-USB FX3 SuperSpeed USB Controller
GPIF II Connector
Cyclone III FPGA
Image Sensor Controller
Image Sensor Control Lines
Slave FIFO Interface Controller
256x16 FIFO
ADC (AD7980) and Digipot (AD5235) Controller
SPI Bus
To Analog Devices IGAA G92xx Demo Board
I²C Slave
Status and Control Register Bank
AD7991 Controller
AD5627 Controller
Multi-Port Access Controller (MPAC)
I²C Master
I²C Bus
Clock, Reset and Timing Generator


PC with USB 3.0 Interface
IGAA Library based Software
Could be Python scripts
USB 3.0
Cypress CYUSB3KIT-001 EZ-USB Development Kit
USB 3.0
CYUSB301X EZ-USB FX3 SuperSpeed USB Controller
GPIF II Connector
tinyCLUNX33 is an USB3+FPGA combo: possible to build this system with Zephyr today
(P.S. FX3 would be FX10/FX20 nowadays)
Cyclone III FPGA
Image Sensor Controller
Image Sensor Control Lines
Slave FIFO Interface Controller
256x16 FIFO
ADC (AD7980) and Digipot (AD5235) Controller
SPI Bus
To Analog Devices IGAA G92xx Demo Board
One of Zephyr supporters!
I²C Slave
Status and Control Register Bank
AD7991 Controller
AD5627 Controller
Multi-Port Access Controller (MPAC)
I²C Master
I²C Bus
Clock, Reset and Timing Generator

Multiple lines

Tools that can be used for building video systems: hardware to access the sensors implement all of that chain

→ Difficulty of embedded video: accessing parallel port or MIPI

→ Can use adapter chips like Himax HX6538 (not yet supported) or small FPGAs

What comes out of an image sensor

Dark (no auto-exposure) Green (no color correction)

Steps of an ISP.


タント 200 色
（印刷本と現物は異なります）
TANT 200 COLOR PAPERS
※パッケージの写真と製品は多少異なる場合があります。
※折り方は表紙の裏面にあります。
注意（ちゅうい）
保護者の方へ ぜひお読みください。
●カラーペーパーや、袋を口の中に入れたり袋を重ったり絶対にしないでください。窒息の危険があります。
●折り上がった作品で危険な遊びを絶対にしないでください。
●誤飲・窒息の危険がありますので、3才未満のお子様には絶対に与えないでください。
MADE IN JAPAN
4 902031 317691


タント 200 色
（色見本と現物は異なります）
TANT 200 COLOR PAPERS
※パッケージの写真と製品は多少異なる場合があります。
※折り図は表紙の裏面にあります。
注意（ちゅうい）
保護者の方へ ぜひお読みください。
●カラーペーパーや、袋を口の中に入れたり袋を破ったり絶対にしないでください。窒息の危険があります。
●折り上がった作品で危険な遊びを絶対にしないでください。
●誤飲・窒息の危険がありますので、3才未満のお子様には絶対に与えないでください。
03-3668-7821
http://www.kitaboshi.co.jp MADE IN JAPAN
商品コード068007-1200
4 902031 317699


タント 200 色
（色見本と現物は異なります）
TANT 200 COLOR PAPERS
注意（ちゅうい）
●カラーペーパーや、袋を口の中に入れたり袋を頭にかぶったりは絶対にしないでください。窒息の危険があります。
●折り上がった作品で危険な遊びを絶対にしないでください。
●誤飲・窒息の危険がありますので、3才未満のお子様には絶対に与えないでください。


タント 200 色
（色見本と現物は異なります）
TANT 200 COLOR PAPERS
※パッケージの写真と製品は多少異なる場合があります。
※折り目は表現の都合上あります。
注意（ちゅうい）
●カラーペーパーや、袋を口の中に入れたり噛んだり絶対にしないでください。誤飲の危険があります。
●折り上がった作品の包装を遊びや絶対にしないでください。
●誤飲・窒息の危険がありますので、3才未満の子供には絶対に与えないでください。
TANT


タント 200 色
TANT 200 COLOR PAPERS

Why an ISP is useful for robotics?

→ Get always values withing same range

→ Poor exposure: no data at all

→ De-fisheye

→ Avoid artefacts to trigger a detection on the NPU or other vision algorithm

Conclusion: A lot to handle to get a reasonable image out of a sensor!

Hardware that can help accessing this image.

What it takes...

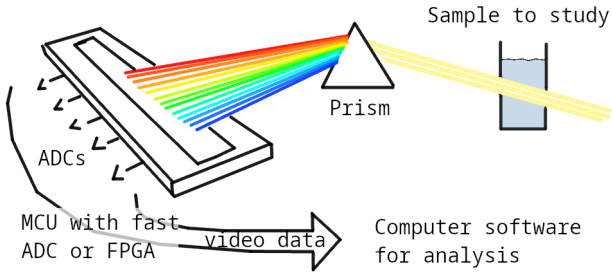
What would it take to build various devices on Zephyr

!! disclaimer: hardware is hard !!

!! disclaimer: not everything shown has drivers !!

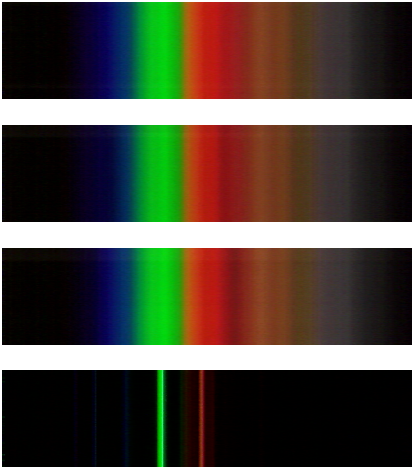
Also works without Zephyr, just putting things in perspective.

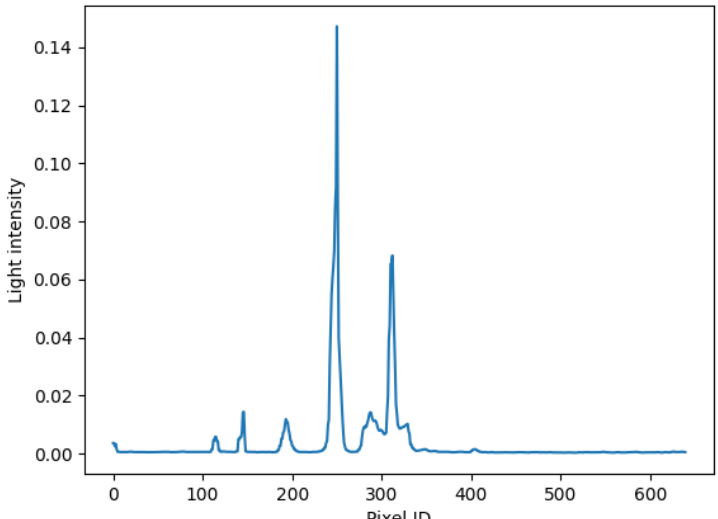
What it takes... Spectrophotometer



Need a very fast ADC! Not many board will have one...

→ Good to have a lot of options.




0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Light intensity
0
100
200
300
400
500
600
Pixel ID

What it takes... Drones

VMU RT1170

Overview

The VMU RT1170 features an i.MX RT1176 dual core MCU with the Cortex-M7 core at 1 GHz and a Cortex-M4 at 400 MHz. The i.MX RT1176 MCU offers support over a wide temperature range and is qualified for consumer, industrial and automotive markets. The VMU RT1170 is the default VMU for CogniPilot's Cerebri, a Zephyr RTOS based Autopilot.

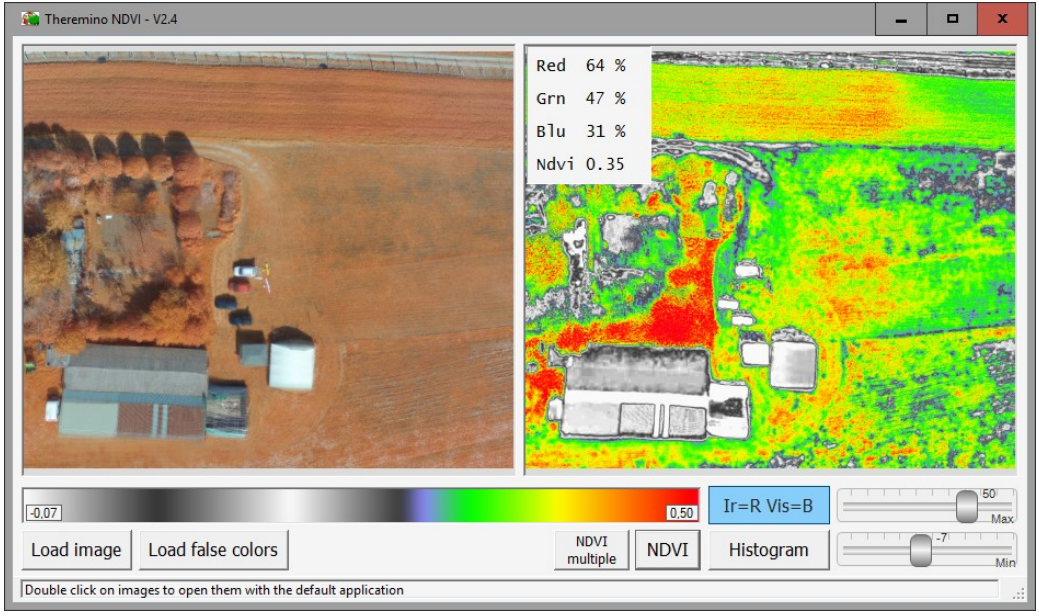
Hardware

- MIMXRT1176DVMAA MCU
  - 1GHz Cortex-M7 & 400Mhz Cortex-M4
  - 2MB SRAM with 512KB of TCM for Cortex-M7 and 256KB of TCM for Cortex-M4
- Memory
  - 512 Mbit Octal Flash
  - TF socket for SD card
- Ethernet
  - 2 wire 100BASE-T1
- USB
  - USB 2.0 connector
- Power
  - Redundant dual picoflex power ports
- Debug
  - 10 pin debug and shell adapter board to 20 Pin JTAG debugger and USB-C shell
- Sensor

Board Overview

VMU RT1170

Name:	`vmu_rt1170`
Vendor:	NXP Semiconductors
Architecture:	arm
SoC:	mimxrt1176


Theremino NDVI - V2.4
Red 64 %
Grn 47 %
Blu 31 %
Ndvi 0.35
-0,07
0,50
Ir=R Vis=B
50
Max
-7
Min
Load image
Load false colors
NDVI multiple
NDVI
Histogram
Double click on images to open them with the default application

What it takes... Yeast monitoring station

Monitoring process of beer, kombucha, lactic fermentation

Video but also...

CO2 polling for building charts.

LED API for illuminating when taking a capture.

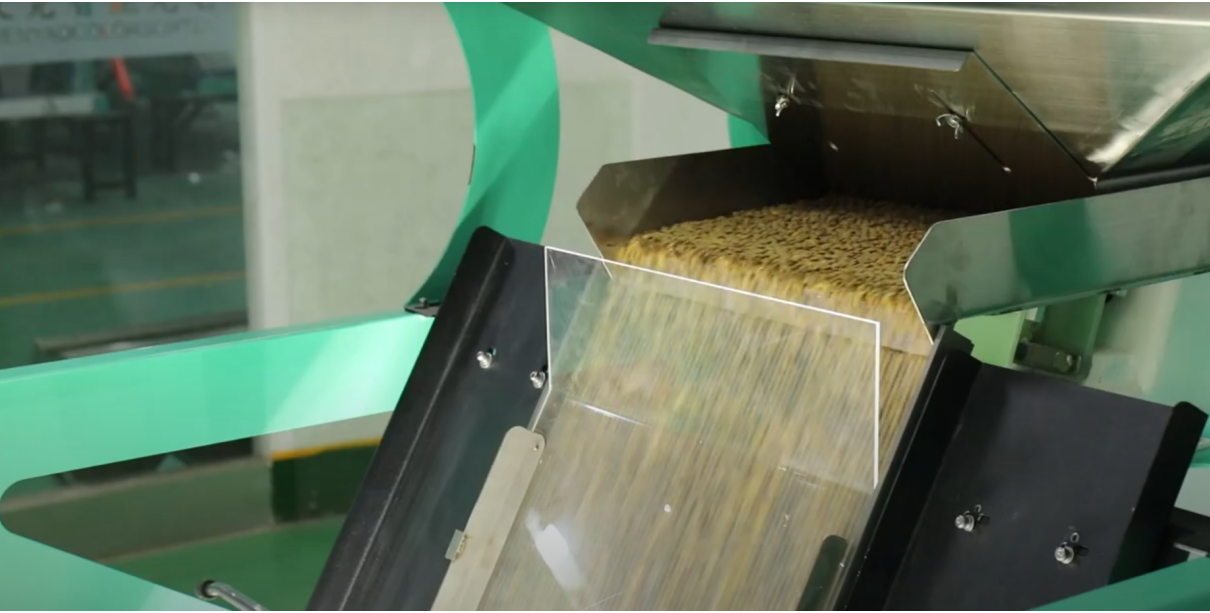
Video controls API for manual exposure tuning during "flash".

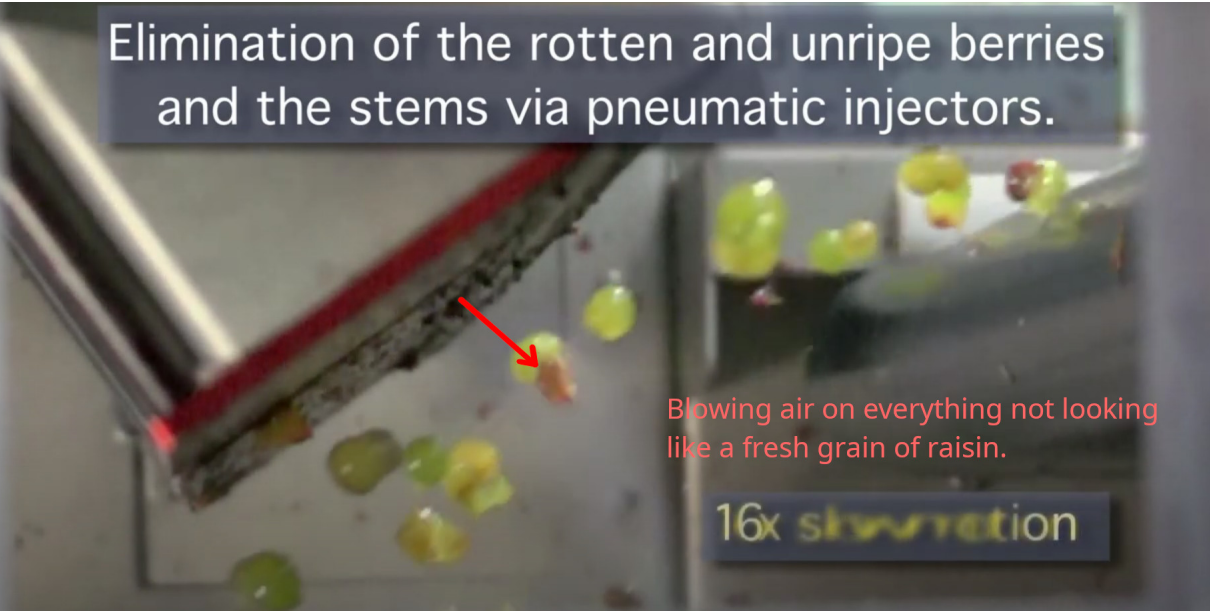
Wi-Fi to the home router.

HTTP client for sending the results.

What it takes... Sorting machine

Fastest board you can get! Most real-time you can get!




Elimination of the rotten and unripe berries and the stems via pneumatic injectors.
Blowing air on everything not looking like a fresh grain of raisin.
16x slowmotion

What it takes... Endoscopes/Borescope

Cameras usged by surgeons

Example of real endoscope camera module (CAMEMAKE):

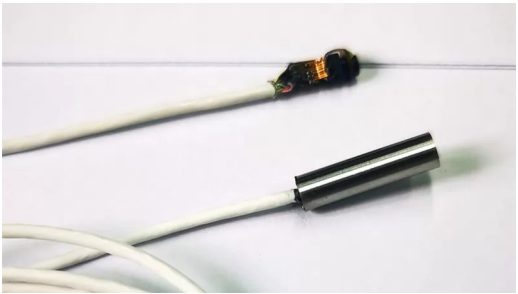
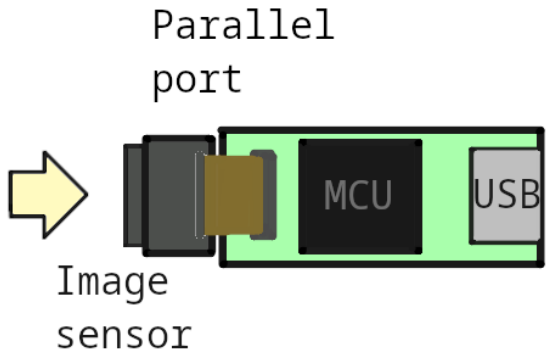
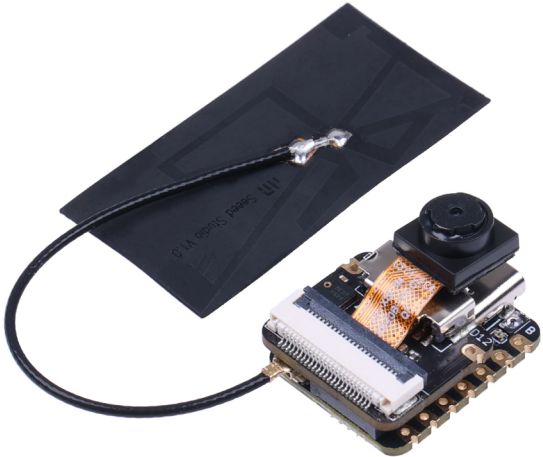


Image sensor driver
Video transfer driver <-> Video APIs <-> UVC

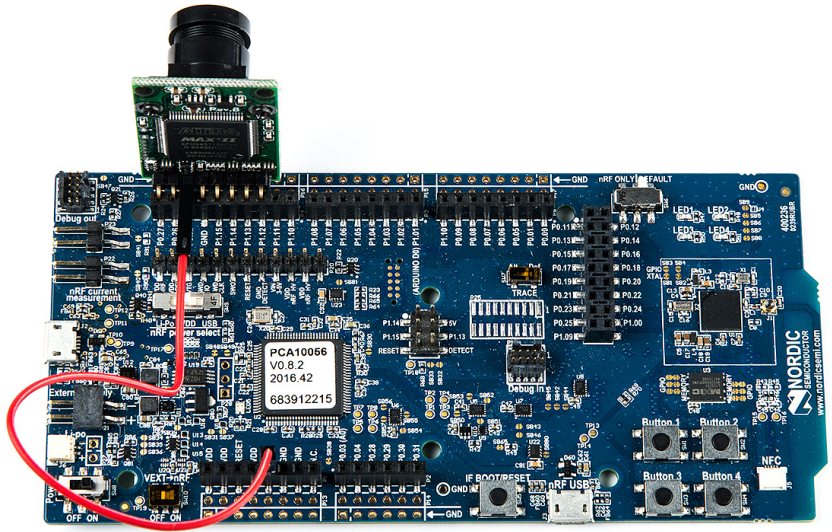


What it takes... Wi-Fi Smartglasses



What it takes... Bluetooth Smartglasses

Pre-Zephyr Nordic era: needs conversion.


PCA10056
V0.8.2
2016.42
683912215
Debug out
Debug in
nRF Current measurement
nRF Power select
VEXT-nRF
OFF ON
TRACE
RESET
DETECT
IF BOOT/RESET
nRF USB
Button 1
Button 2
Button 3
Button 4
NFC
LED1
LED2
LED3
LED4
nRF ONLY
DEFAULT
ARDUINO I/O HEADER
NORDIC SEMICONDUCTOR
www.nordicsemi.com

Any MCU with Bluetooth

SPI

Himax HM0360

(not yet supported)

640x480

Next in Zephyr Video

A sneak peek at what is WIP and could land in Zephyr

→ Zephyr Video Control Framework

→ Zephyr Video Shell

→ Zephyr Video over USB

Video Control Framework

Move repeated operations on every drivers into the API, making drivers simpler and easier to use.

→ **More similar to Linux**

Porting drivers becomes easier.

→ **Drivers do not have to implement get(min/max/default/current)**

(just declare them once during `init()`) but only set(current).

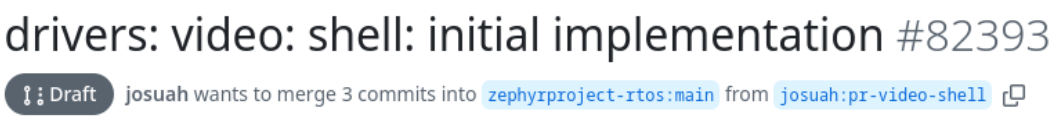
→ Adds types to controls,

→ Adds range checks for min/max.

→ Apply the default value at startup automatically.

→ **No more need to forward controls to the child manually.**
Resolves which driver to call.

Zephyr Shell



→ Zephyr Shell: a debug tap for tinkering the video devices

→ Explore the video API from command line.

→ Note: command names not stable yet

```
uart:~$ video show emul_rx
min vbuf count: 0
min line count: 0
max line count: 0
current format: RGBP 64x20 (2560 bytes)
pixel format RGBP 64x20 (2560 bytes)
- 1/15 sec (15 FPS)
- 1/30 sec (30 FPS)
- 1/60 sec (60 FPS)
pixel format YUYV 64x20 (2560 bytes)
- 1/15 sec (15 FPS)
- 1/30 sec (30 FPS)
uart:~$ video alloc 4096
video buffer 0x40000a14 ready with 4096 bytes
uart:~$ video enqueue emul_rx
sent video buffer of size 4096 to emul_rx, came back with 2560 bytes
```

An old idea: tiv, imcat, catimg, n³, viu...

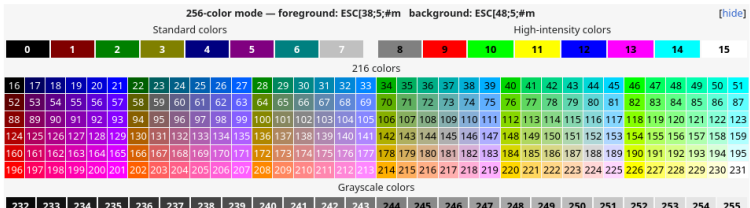
8-bit [edit]

As 256-color lookup tables became common on graphic cards, escape sequences were added to select from a pre-defined set of 256 colors:[*citation needed*]

```
ESC[38;5;⟨n⟩m Select foreground color      where n is a number from the table below
ESC[48;5;⟨n⟩m Select background color
  0-  7:  standard colors (as in ESC [ 30–37 m)
  8- 15:  high intensity colors (as in ESC [ 90–97 m)
 16-231:  6 × 6 × 6 cube (216 colors): 16 + 36 × r + 6 × g + b (0 ≤ r, g, b ≤ 5)
232-255:  grayscale from dark to light in 24 steps
```

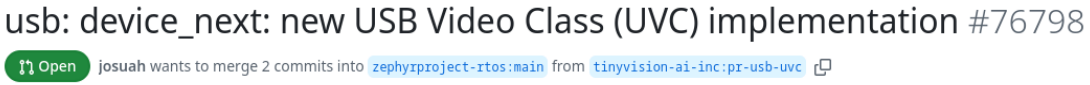
The ITU's T.416 Information technology - Open Document Architecture (ODA) and interchange format: Character content architectures[34] uses ":" as separator characters instead:

```
ESC[38:5:⟨n⟩m Select foreground color      where n is a number from the table below
ESC[48:5:⟨n⟩m Select background color
```



There has also been a similar but incompatible 88-color encoding using the same escape sequence, seen in `rxvt` and `xterm-88color`. Not much is known about the scheme besides the color codes. It uses a 4×4×4 color cube.

Video over USB (UVC)



The webcam protocol for any Zephyr video device.

→ Video API: Exposes a normal video sink to

→ Zero conf: only implement the video driver and declare it

→ Extended UVC support: custom formats and controls, multiple streams

→ Cross-platform support: tested on various desktop and mobile OSes

→ Used in real products: ConstructiveRealities 3D camera (depth-sensing aka ToF)

Beyond Zephyr: ecosystem around it

What UVC adds to the table:

→ USB camera protocol supported on Linux, Windows, MaxOS, Android, iPad (not iOS yet), BSDs, 9front, QNX... (thanks to laptop lid cameras)

→ Linux interoperability: Standardize all the video controls with Linux

→ ROS2: integration of robotics (via USB cameras)

→ OpenCV: Computer Vision (via USB cameras)

→ Want to suport a new sensor on any ecosystem? Bring Zephyr support, and now it's everywhere

→ OpenMV: MicroPython video APIs + devboards (not related to Zephyr)

Question time?

You are welcome to visit the Zephyr stand at building K.