PULP PLATFORM
Open Source Hardware, the way it should be!

Bitcraze Workshop: PULP Introduction

Lorenzo Lamberti, Hanna Müller, Vlad Niculescu, Manuele Rusci,
Daniele Palossi

http://pulp-platform.org @pulp_platform https://www.youtube.com/pulp_platform

Team

Lorenzo

Hanna

Vlad

Manuele

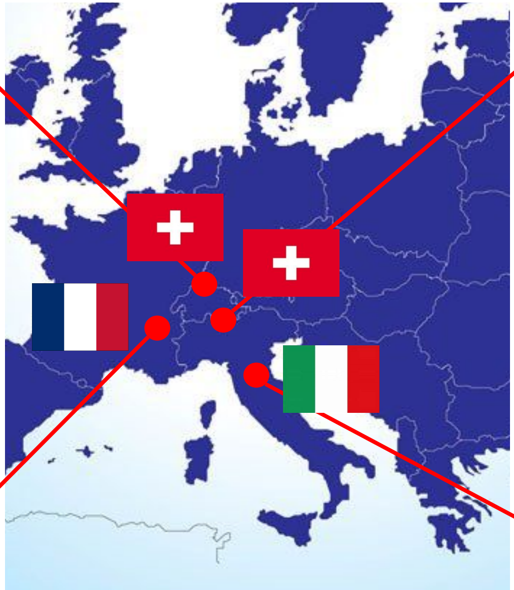
Daniele

Lorenzo Lamberti	*University of Bologna*	lorenzo.lamberti@unibo.it
Hanna Müller	*ETH Zürich*	hanmuell@iis.ee.ethz.ch
Vlad Niculescu	*ETH Zürich*	vladn@iis.ee.ethz.ch
Dr. Manuele Rusci	*University of Bologna / Greenwaves Tech.*	manuele.rusci@greenwaves-technologies.com
Dr. Daniele Palossi	*IDSIA Lugano / ETH Zürich*	dpalossi@iis.ee.ethz.ch

Team affiliations

Polytechnic of Zürich (ETHZ)

Greenwaves Tech. in Grenoble (GWT)



University of Lugano (USI/SUPSI)

University of Bologna (UniBO)

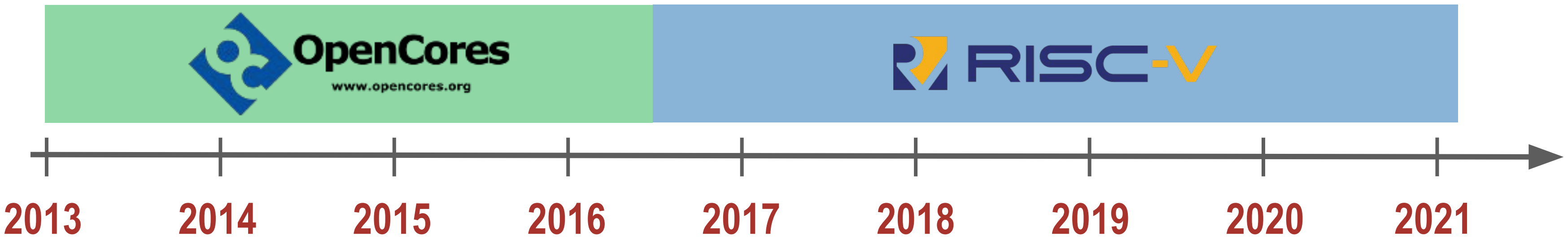
! We are looking for outstanding Ph.D. candidates: https://www.supsi.ch/home_en/supsi/lavora-con-noi/2021-02-24-bando816.html

Agenda

	Topic	Time	Description	Speaker
Overview	**PULP introduction**	15’	Parallel Ultra-low Power (PULP) overview	Daniele
	GAP8 architecture	10’	System-on-Chip hardware architecture	Manuele
	AI-deck	15’	Printed circuit board overview & GAP8 SDK	Hanna
	Break	15’		
Hands-on	**Basic programming**	10’	JTAG programming & ‘Hello World’ example	Hanna
	Image manipulation	10’	Image acquisition, parallel image filter	Hanna
	Firmware integration	15’	App-layer integration, UART communication	Vlad
	Video streaming	20’	Basic Wi-Fi streaming, JPEG image compression	Lorenzo
	Conclusion	5’	Final remarks	Daniele

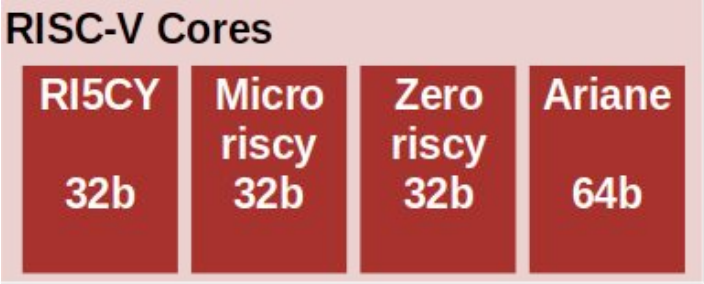
Parallel Ultra-low Power (PULP)

- The **PULP** project started in **2013**
- Collaboration between the **University of Bologna** and **ETH Zürich**
  - Large team, about 60 people, not all are working on PULP
- Academic/Research goals:
  - Create a compute platform used for **research** (e.g., autonomous nano-drones) by the PULP and other groups in **Europe** and in the **World**
  - Push **energy efficiency** of IoT computing systems as much as possible (we target research on low-power MCUs)
  - **Open-source** approach
- We wanted to start with a clean slate, no need to remain compatible with legacy systems, **no dependency with any commercial IP**
- We started with **OpenRISC** and around mid-2016 we moved to **RISC-V** ISA:
  - Larger community, more momentum



PULP ecosystem

We have developed several optimized RISC-V cores



RISC-V Cores

RI5CY 32b

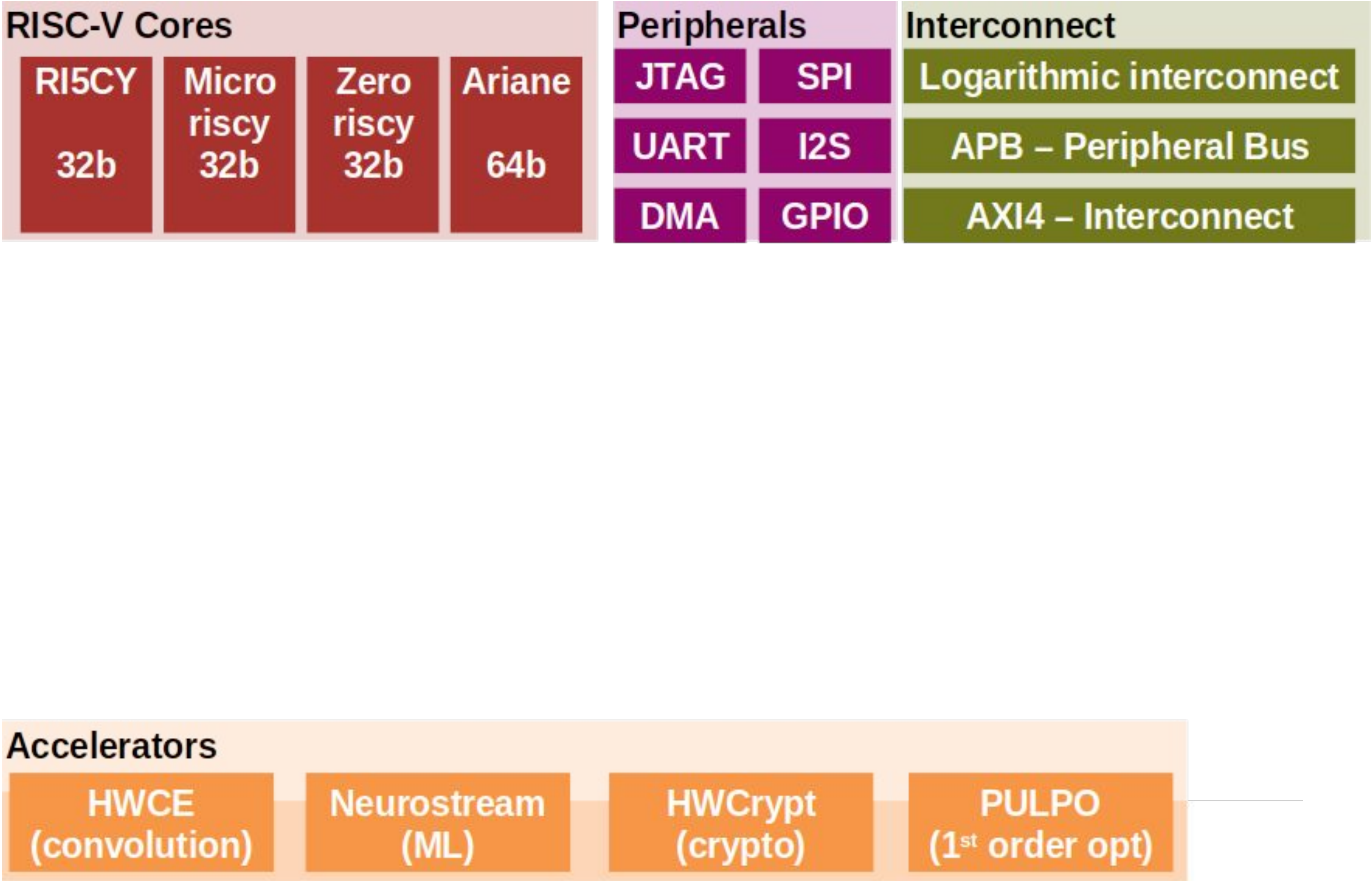
Micro riscy 32b

Zero riscy 32b

Ariane 64b

PULP ecosystem

Only processing cores are not enough, we need more



RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

Peripherals

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

Interconnect

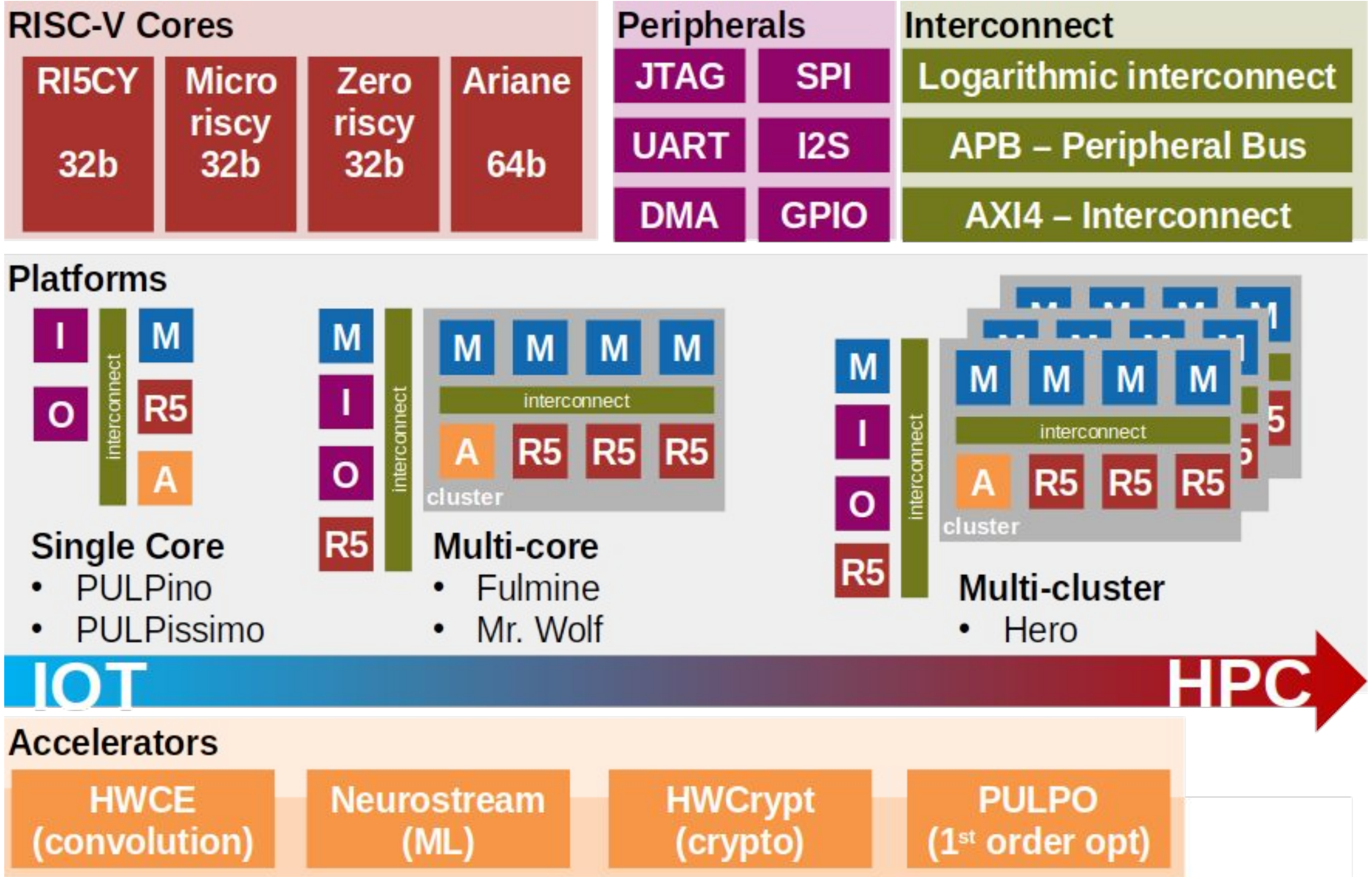
- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

PULP ecosystem

All these components are combined into platforms



RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

Peripherals

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Platforms

Single Core

- PULPino
- PULPissimo

Multi-core

- Fulmine
- Mr. Wolf

Multi-cluster

- Hero

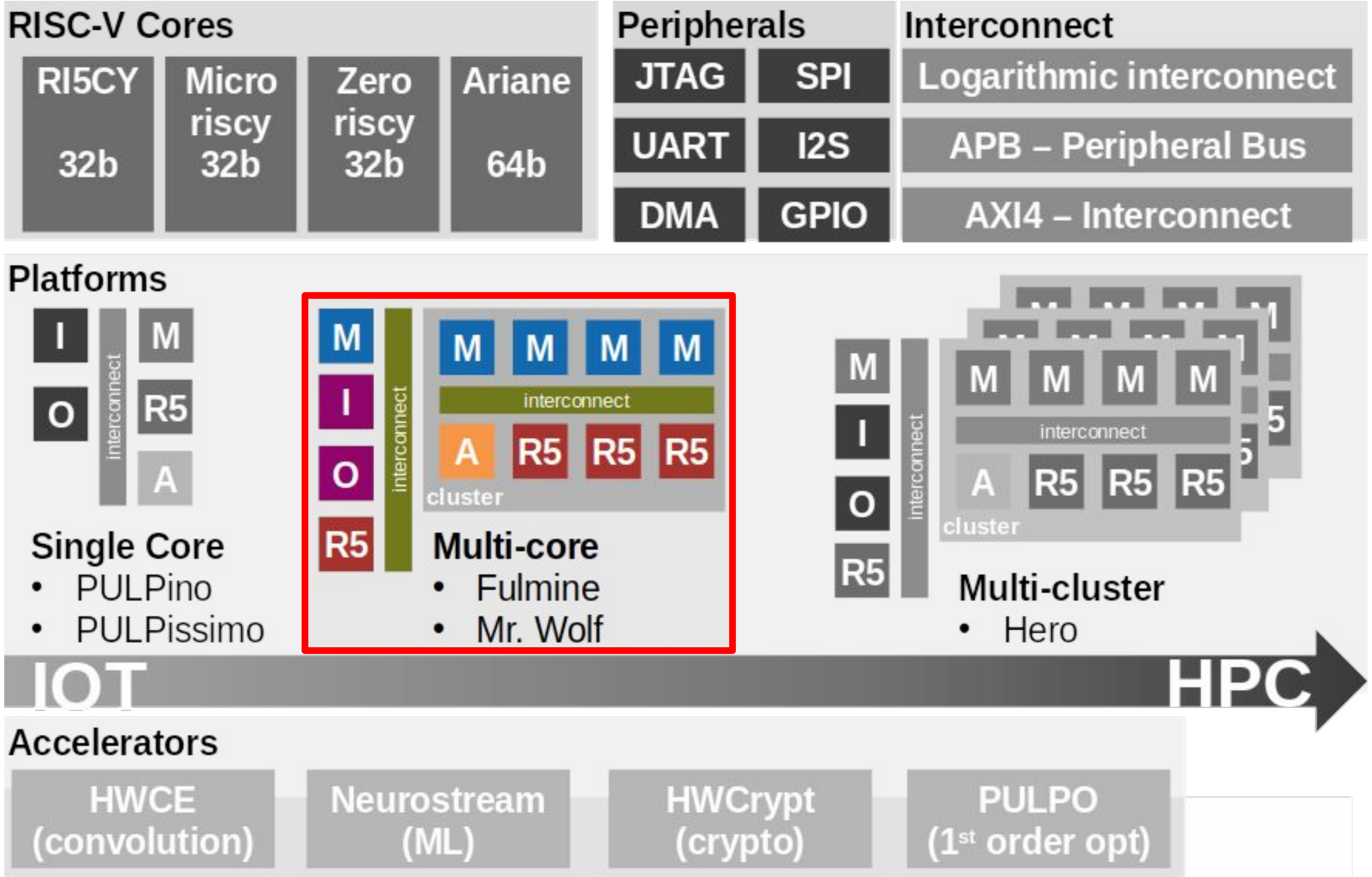
IOT → HPC

Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

PULP ecosystem

All these components are combined into platforms



RISC-V Cores

- RI5CY 32b
- Micro riscy 32b
- Zero riscy 32b
- Ariane 64b

Peripherals

- JTAG
- SPI
- UART
- I2S
- DMA
- GPIO

Interconnect

- Logarithmic interconnect
- APB – Peripheral Bus
- AXI4 – Interconnect

Platforms

Single Core

- PULPino
- PULPissimo

Multi-core

- Fulmine
- Mr. Wolf

Multi-cluster

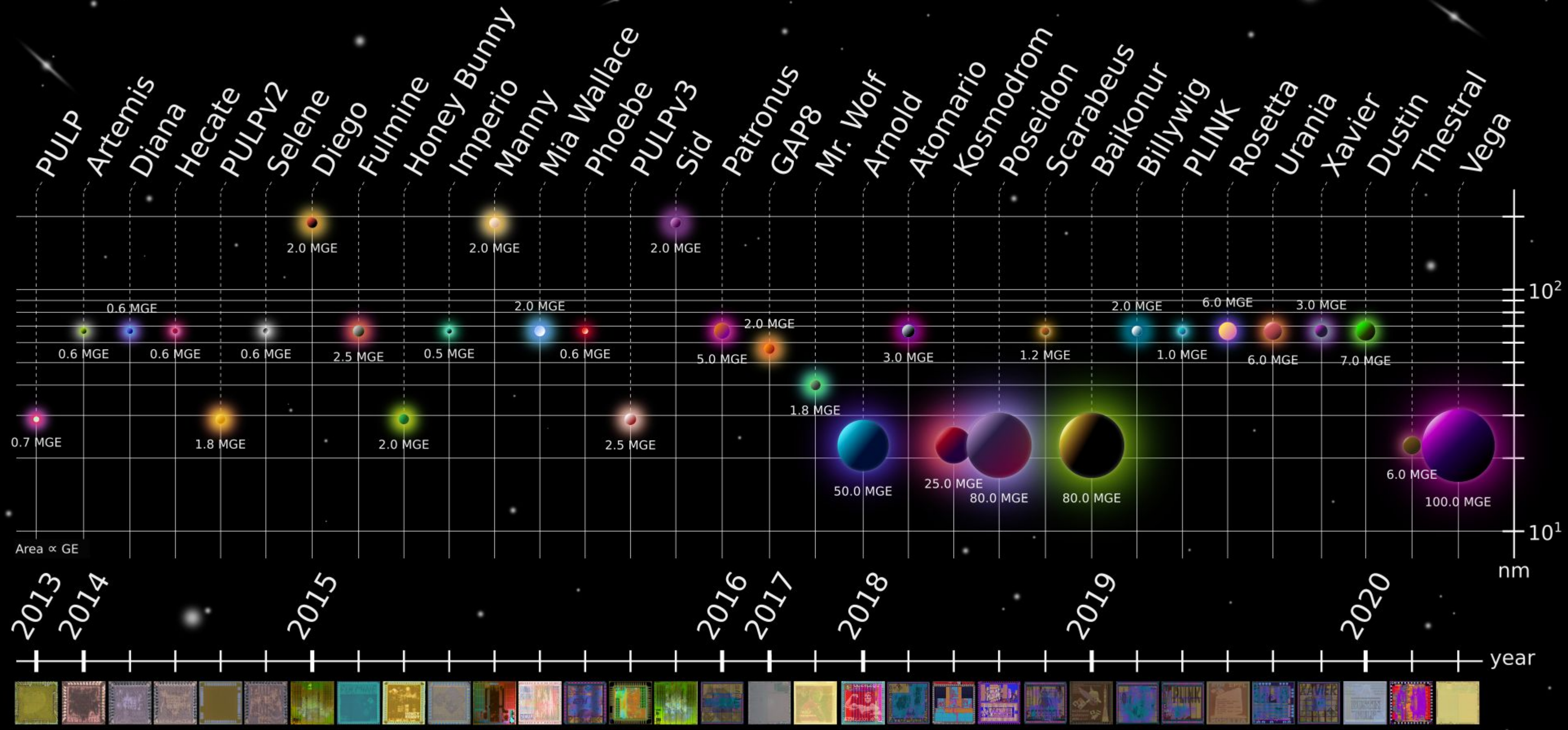
- Hero

Accelerators

- HWCE (convolution)
- Neurostream (ML)
- HWCrypt (crypto)
- PULPO (1st order opt)

PULP Silicon Prototypes

History of the PULP:



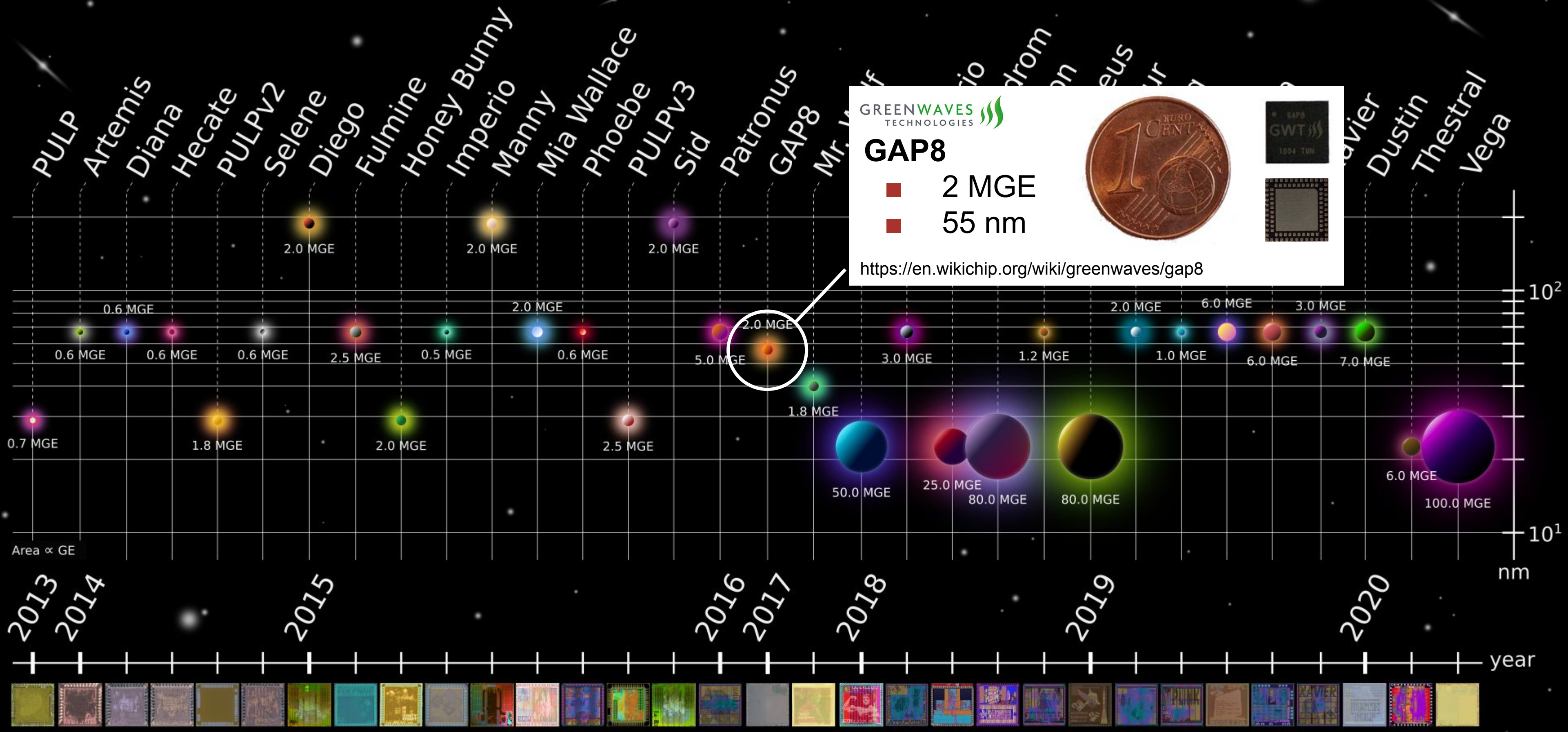
Copyright 2021 © ETH zürich

http://asic.ethz.ch/applications/Pulp.html

Credit: Daniele Palossi

PULP Silicon Prototypes

History of the PULP:



GREENWAVES TECHNOLOGIES

GAP8

- 2 MGE
- 55 nm

https://en.wikichip.org/wiki/greenwaves/gap8

Copyright 2021 © ETH zürich

http://asic.ethz.ch/applications/Pulp.html

Credit: Daniele Palossi

Who uses PULP?

Industrial users:

- OPENHW GROUP PROVEN PROCESSOR IP
- lowRISC
- Google
- IBM
- NXP
- GREENWAVES TECHNOLOGIES
- ST life.augmented
- GLOBAL FOUNDRIES
- EMBECOSM
- DOLPHIN INTEGRATION
- CEVA
- IQ-ANALOG
- Mentor A Siemens Business
- cadence
- QuickLogic
- onespin
- imperas
- SILICON LABS
- axiomise predictable formal verification
- ASHLING
- ACP AG Advanced Circuit Pursuit
- Valtrix SYSTEMS
- antmicro
- WITTENSTEIN
- THALES
- Semidynamics silicon design and verification services
- HENSOLDT

Direct research collaborators:

- Politecnico di Torino
- University of Cambridge
- USI Lugano
- TU Kaiserslautern
- University of Cagliari

- IBM Research Zurich
- EPF Lausanne
- CSEM Neuchatel
- Princeton University

- Technische Universität Graz
- CEA-Leti Grenoble
- Fraunhofer-Gesellschaft
- Sapienza Università di Roma

Academic users we are aware of:

- Università di Genova
- Politecnico di Milano
- Fondazione Bruno Kessler
- Lund University

- Stanford University
- UC Los Angeles
- UC San Diego
- Columbia University

- Universitat Bar-Ilan
- İstanbul Teknik Üniversitesi
- NCTU Hsinchu
- University of Zagreb, FER

- TUT Tampere
- RWTH Aachen
- IST University of Lisboa
- UFRN Rio Grande do Norte

- TU Darmstadt
- Universität Bremen
- Hongik University Seoul
- IIT Kharagpur

- LIRMM Montpelier
- University of Stuttgart
- University of Tübingen
- TU Münich

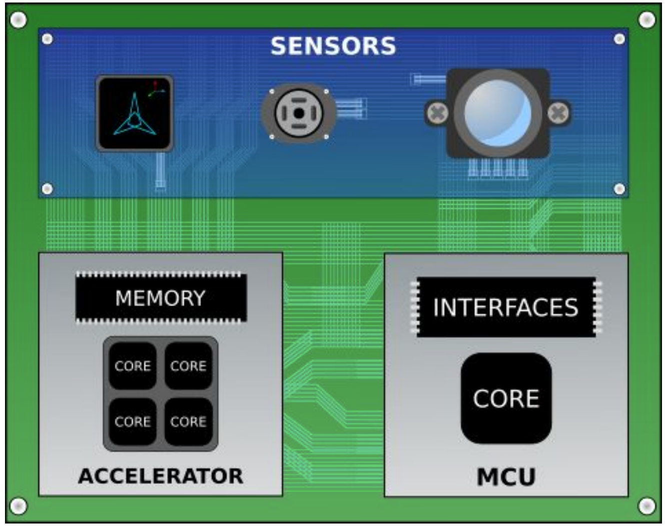
- FORTH Hellas
- Kyoto University
- Tecnologico de Costa Rica

- Chalmers Göteborg
- NTNU Trondheim
- IDSIA Manno

- FAU Erlangen-Nürnberg
- TU Dresden
- SVNIT Surat

The PULP-Shield

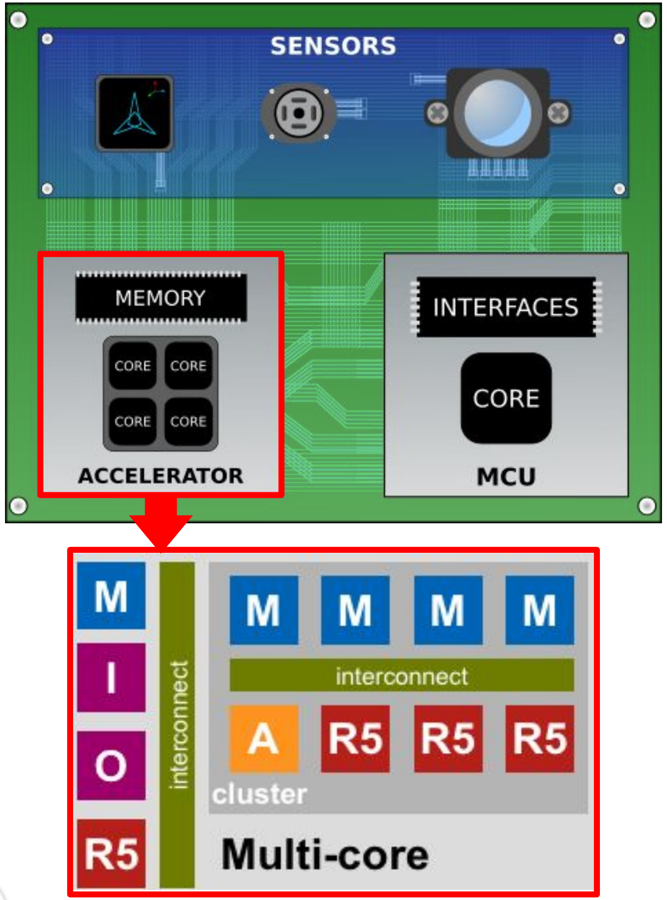
ULP heterogeneous model [1]



[1] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini. "Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms." IEEE DATE, *2016*.

The PULP-Shield

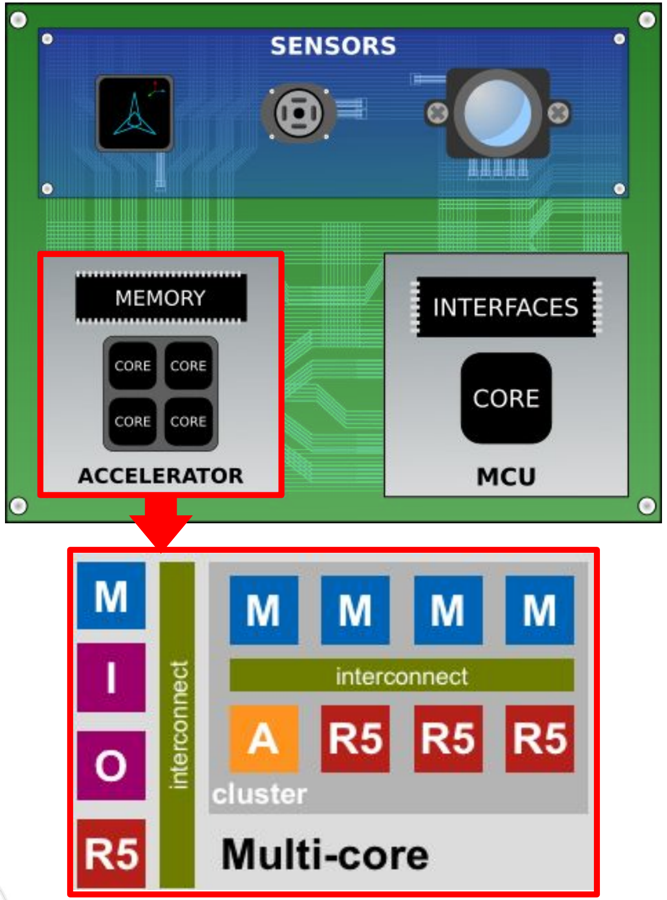
ULP heterogeneous model [1]

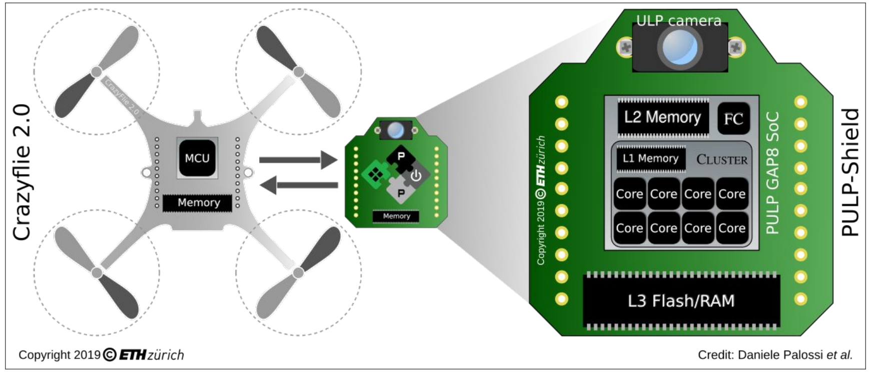


[1] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini. "Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms." IEEE DATE, *2016*.

The PULP-Shield

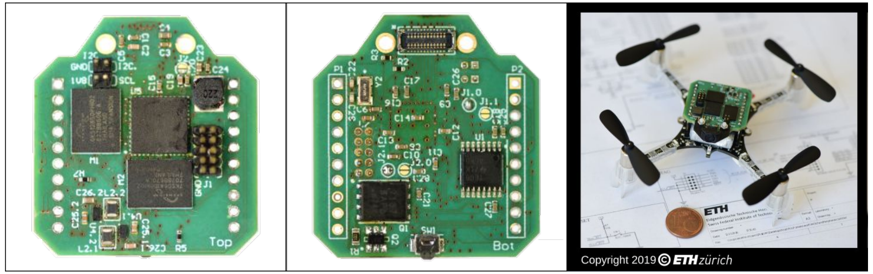
ULP heterogeneous model [1] → PULP-Shield [2]





Copyright 2019 © ETH zürich

Credit: Daniele Palossi *et al.*



Copyright 2019 © ETH zürich

- ~ 5 g – 30x28 mm
- PULP GAP8 SoC
- Off-chip DRAM/Flash
- QVGA ULP Camera
- **Open source hardware**

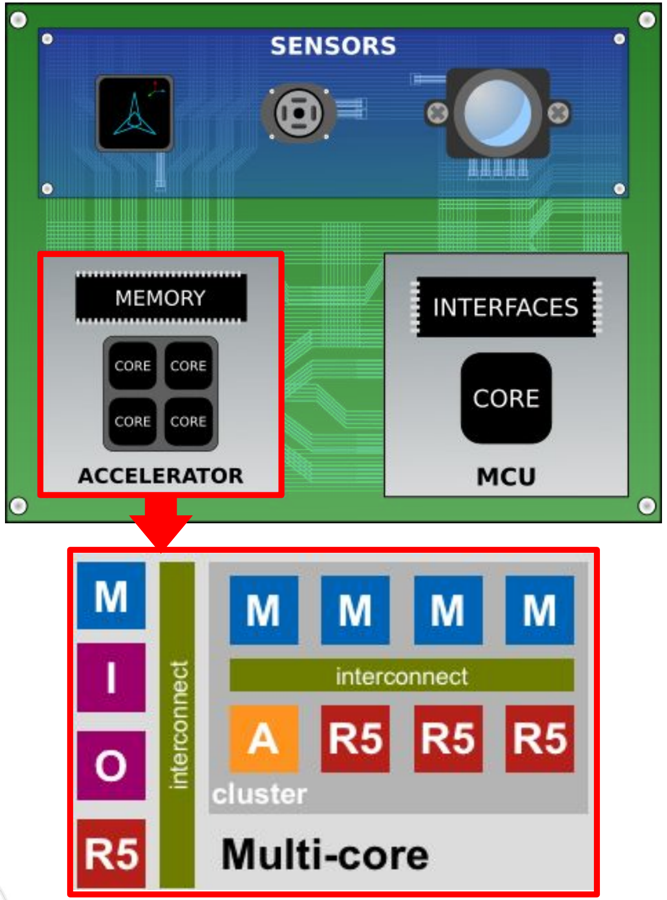
ALTIUM DESIGNER

GitHub

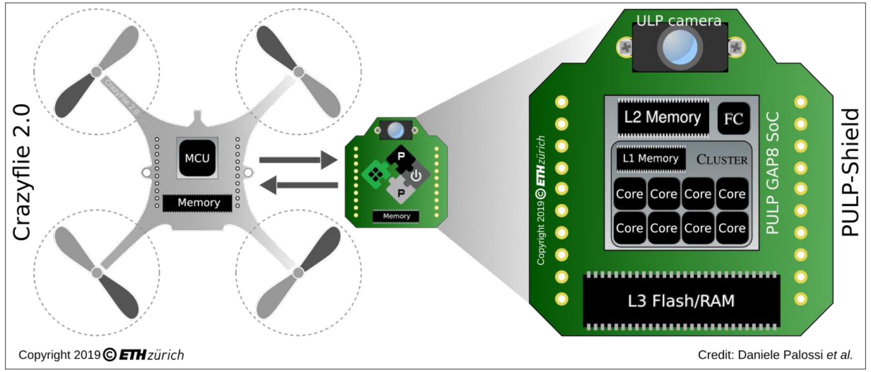
[1] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini. "Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms." IEEE DATE, *2016.*
[2] D. Palossi, F. Conti, and L. Benini "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs." IEEE DCOSS, *2019.*

The PULP-Shield

ULP heterogeneous model [1]

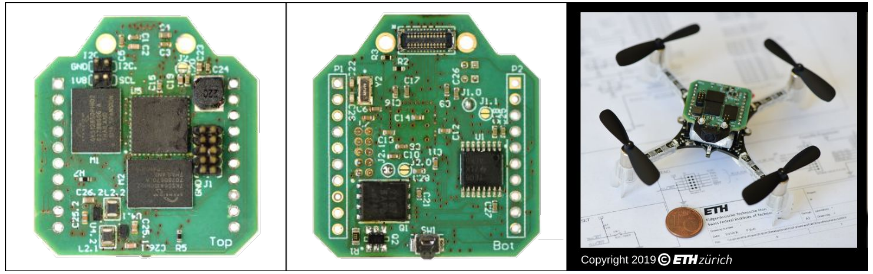


PULP-Shield [2]



Copyright 2019 © ETH zürich

Credit: Daniele Palossi *et al.*



Copyright 2019 © ETH zürich

- ~ 5 g – 30x28 mm
- PULP GAP8 SoC
- Off-chip DRAM/Flash
- QVGA ULP Camera
- **Open source hardware**

ALTIUM DESIGNER

GitHub

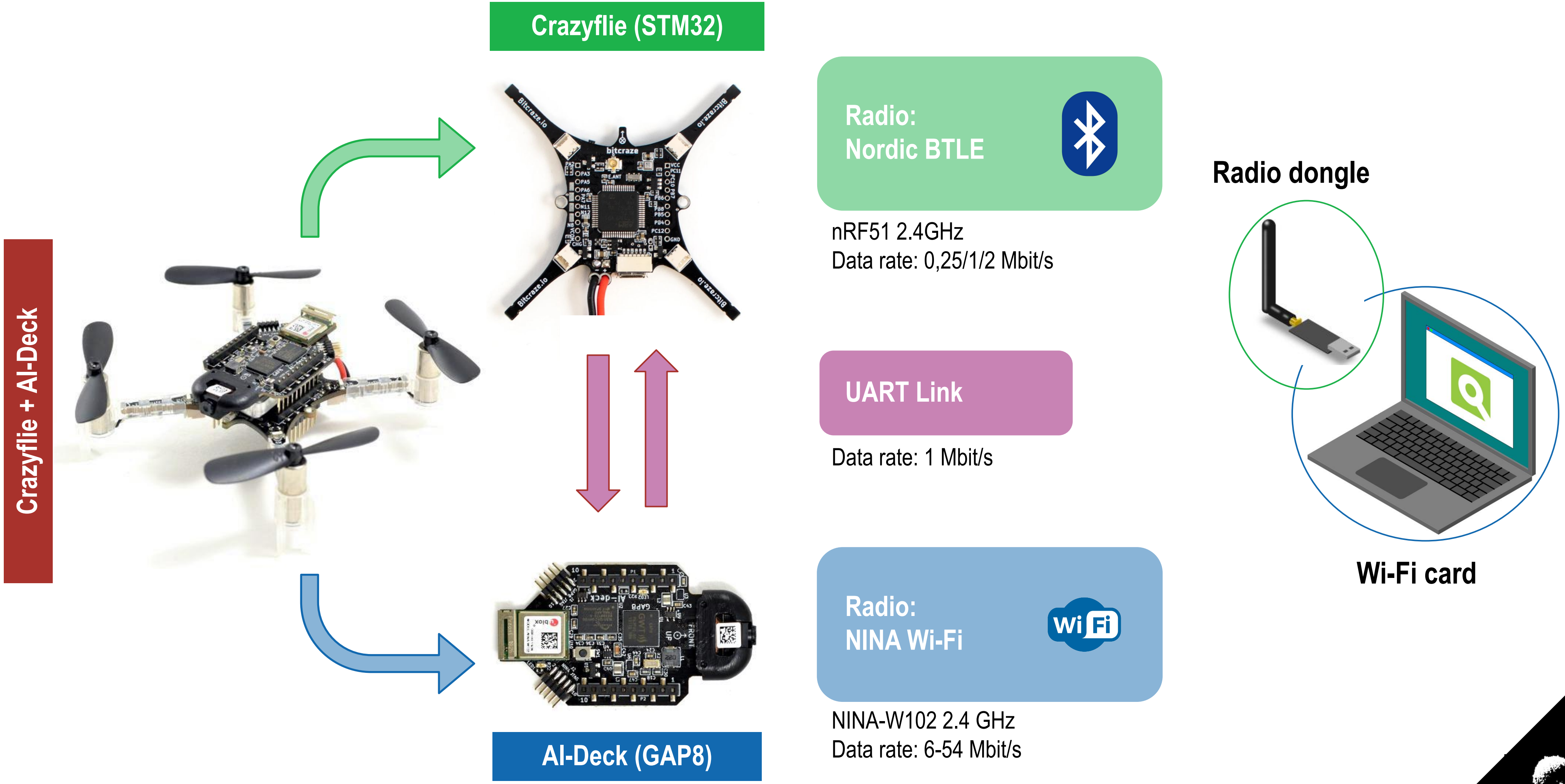
AI-Deck

bitcraze

- ~ 8 g – 40x28 mm
- PULP GAP8 SoC
- 8/64 MB DRAM/Flash
- QVGA ULP Camera
- WiFi module

[1] F. Conti, D. Palossi, A. Marongiu, D. Rossi, and L. Benini. "Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms." IEEE DATE, *2016*.
[2] D. Palossi, F. Conti, and L. Benini "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs." IEEE DCOSS, *2019*.

The AI-Deck



Crazyflie + AI-Deck

Crazyflie (STM32)

AI-Deck (GAP8)

Radio:
Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

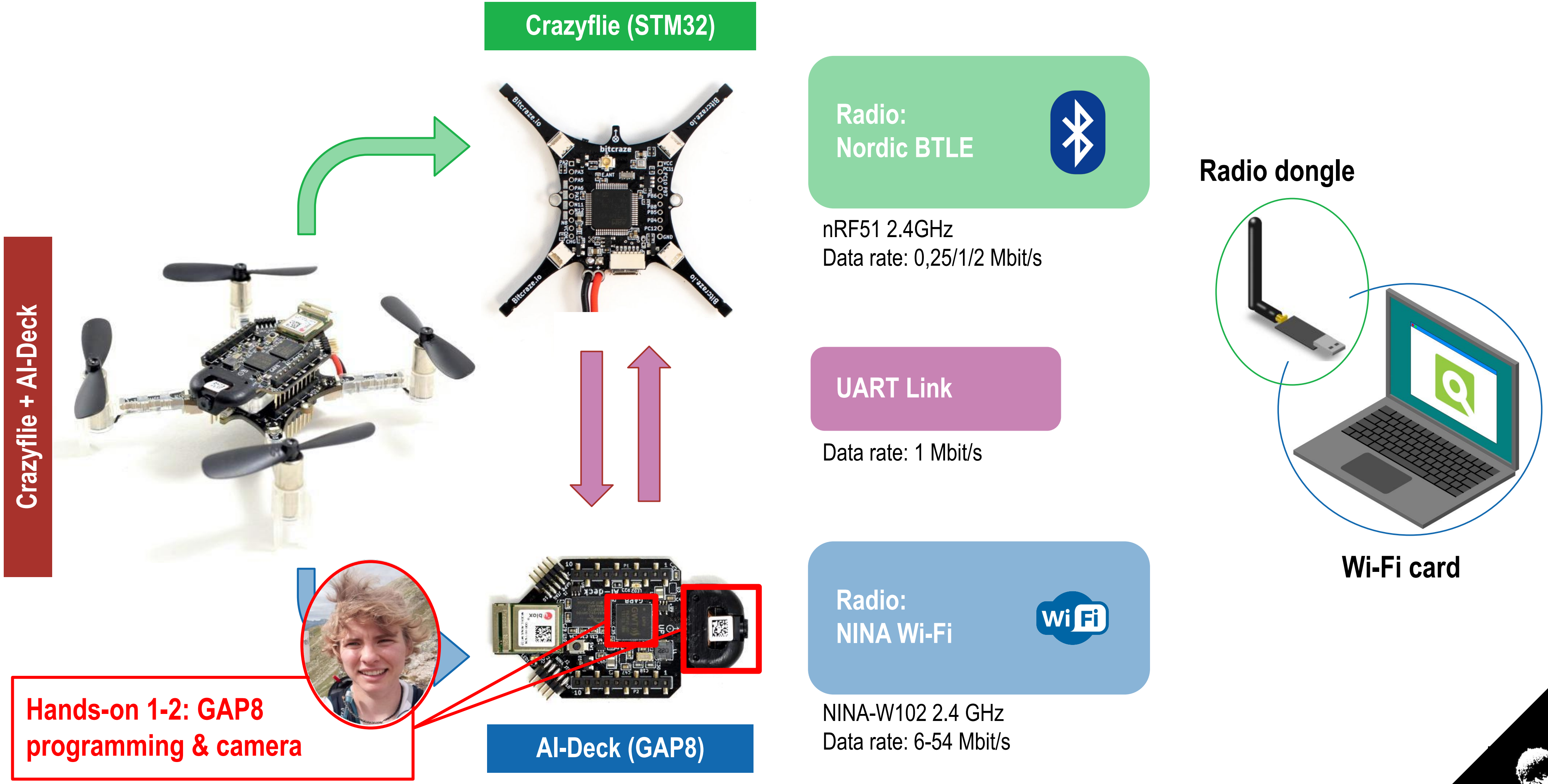
Radio:
NINA Wi-Fi

NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

Radio dongle

Wi-Fi card

The AI-Deck



Crazyflie (STM32)

Crazyflie + AI-Deck

Radio:
Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

Radio:
NINA Wi-Fi

NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

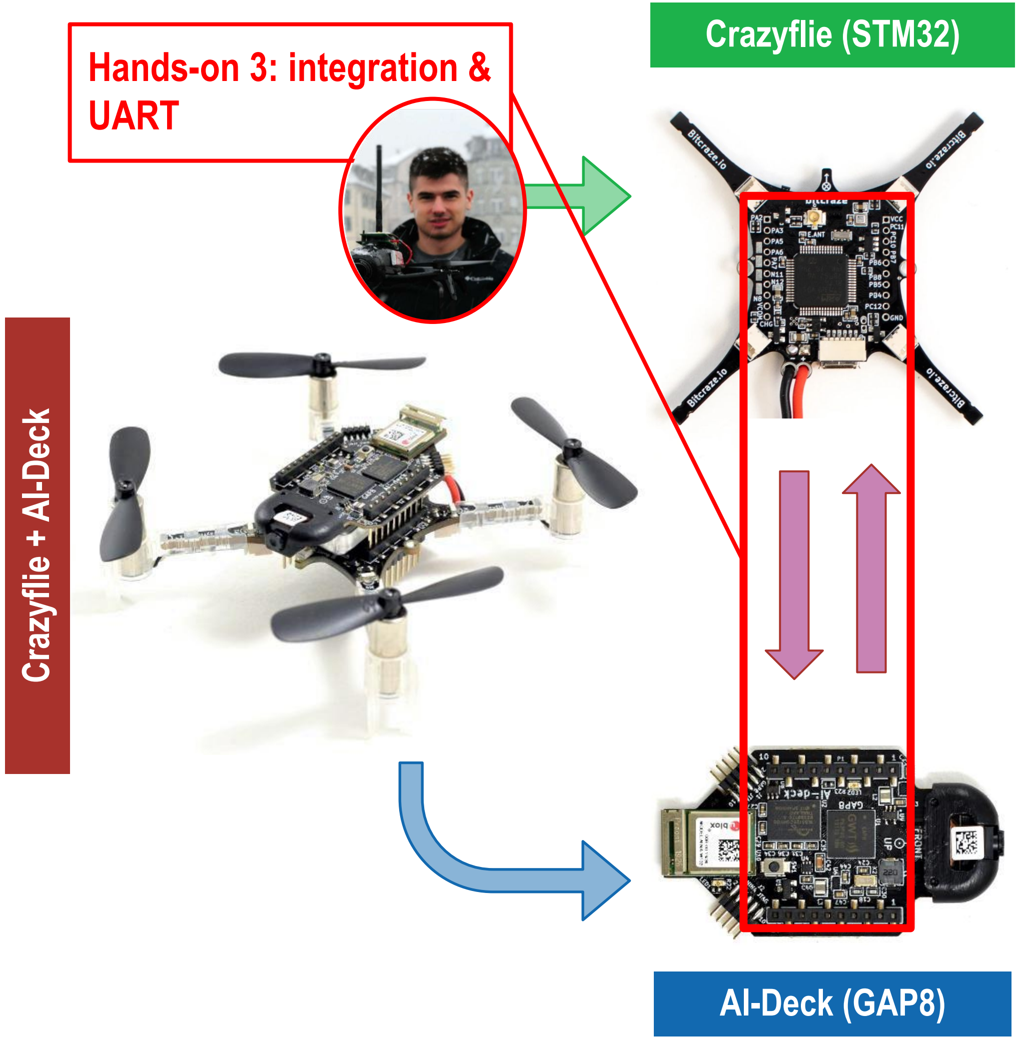
Radio dongle

Wi-Fi card

AI-Deck (GAP8)

Hands-on 1-2: GAP8
programming & camera

The AI-Deck



Radio: Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

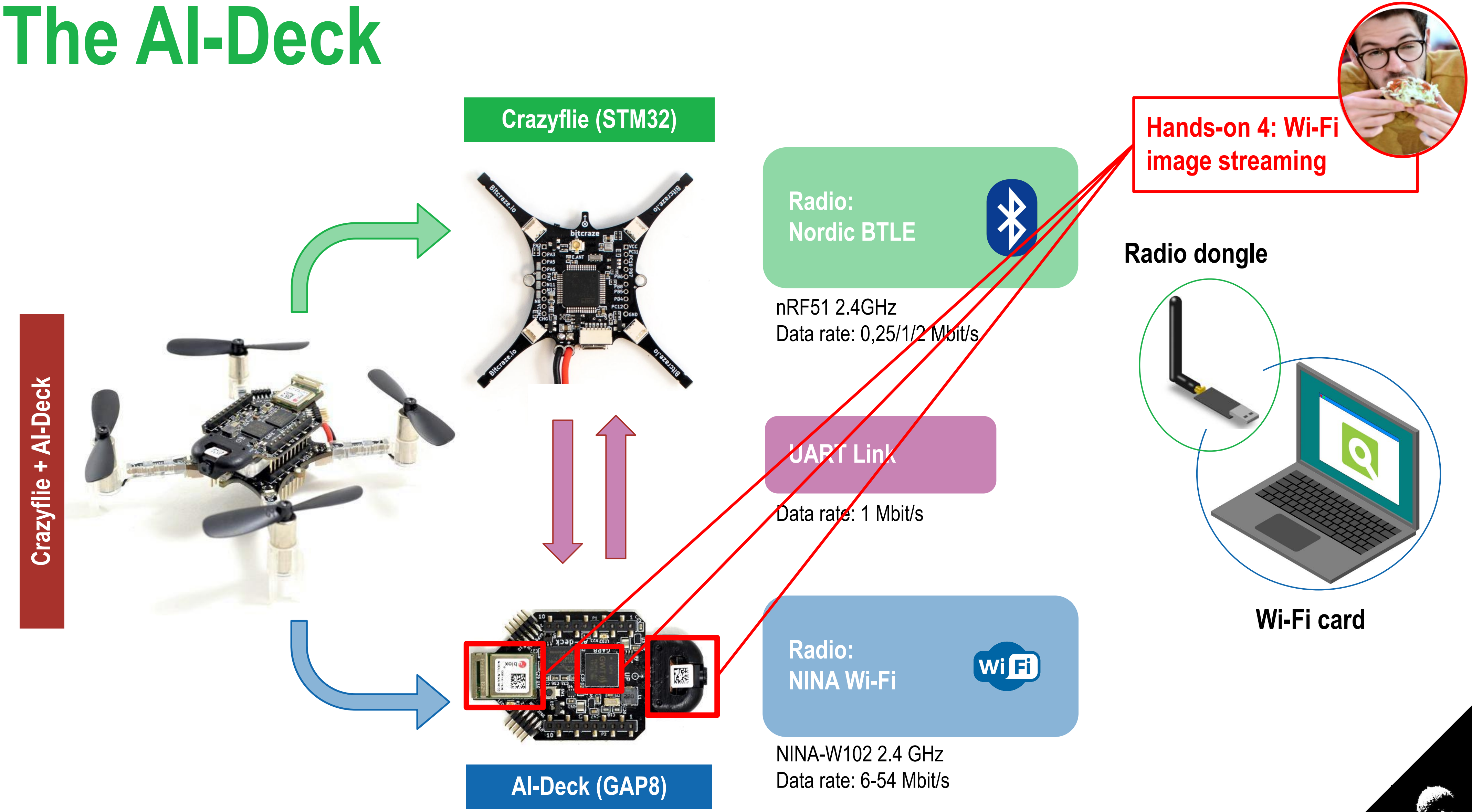
UART Link

Data rate: 1 Mbit/s

Radio: NINA Wi-Fi

NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

The AI-Deck



Crazyflie + AI-Deck

Crazyflie (STM32)

Radio:
Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

Radio:
NINA Wi-Fi

NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

AI-Deck (GAP8)

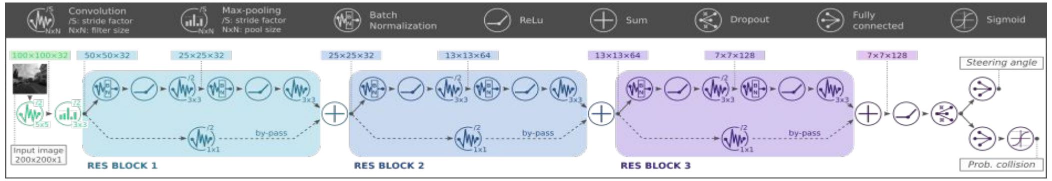
Hands-on 4: Wi-Fi image streaming

Radio dongle

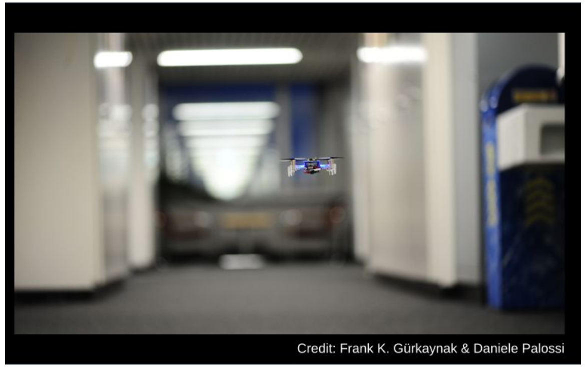
Wi-Fi card

AI-based applications (not in this workshop)

PULP-Dronet:



Task:	Lane detection / Obstacle avoidance
CNN:	41 MMAC/frame
Onboard:	6fps@45mW / 18fps@272mW
Device:	PULP-Shield (GAP8)
arXiv.org	https://arxiv.org/abs/1805.01831



Credit: Frank K. Gürkaynak & Daniele Palossi

GitHub

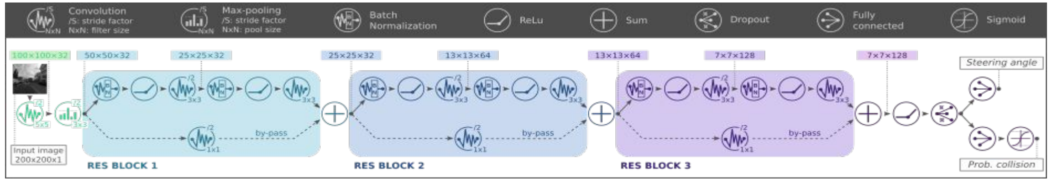
https://github.com/pulp-platform/pulp-dronet

YouTube

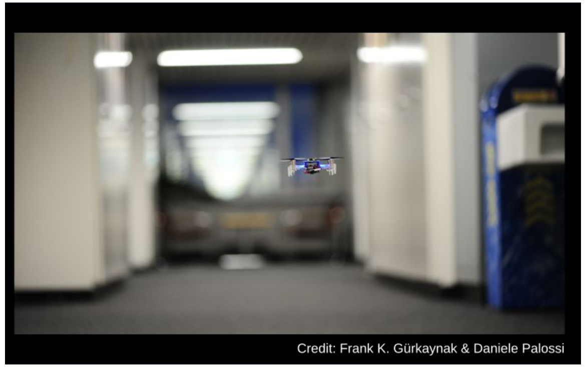
https://www.youtube.com/watch?v=JKY03NV3C2s

AI-based applications (not in this workshop)

PULP-Dronet:



Task:	Lane detection / Obstacle avoidance
CNN:	41 MMAC/frame
Onboard:	6fps@45mW / 18fps@272mW
Device:	PULP-Shield (GAP8)
arXiv.org	https://arxiv.org/abs/1805.01831


Credit: Frank K. Gürkaynak & Daniele Palossi

GitHub

https://github.com/pulp-platform/pulp-dronet

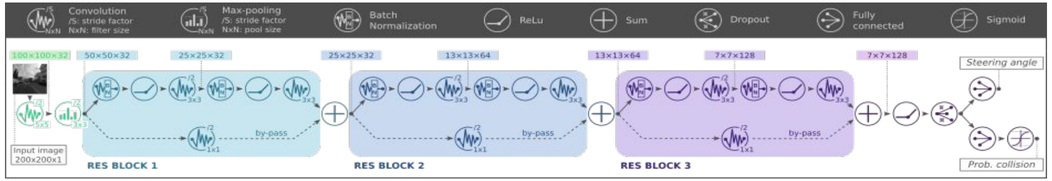
YouTube

https://www.youtube.com/watch?v=JKY03NV3C2s

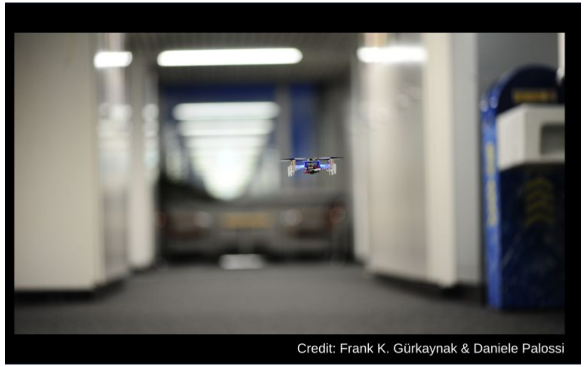
! **PULP-Dronet v2** for the AI-Deck coming soon on GitHub

AI-based applications (not in this workshop)

PULP-Dronet:



Task:	Lane detection / Obstacle avoidance
CNN:	41 MMAC/frame
Onboard:	6fps@45mW / 18fps@272mW
Device:	PULP-Shield (GAP8)
arXiv.org	https://arxiv.org/abs/1805.01831

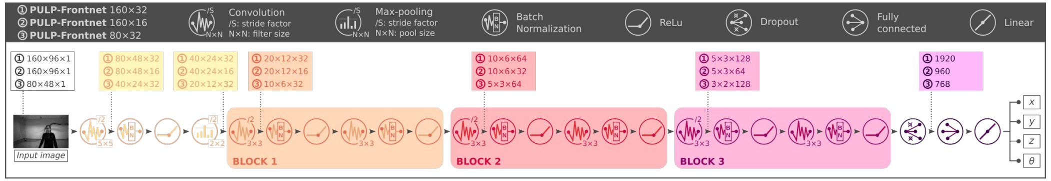

Credit: Frank K. Gürkaynak & Daniele Palossi

GitHub
https://github.com/pulp-platform/pulp-dronet

YouTube
https://www.youtube.com/watch?v=JKY03NV3C2s

! **PULP-Dronet v2** for the AI-Deck coming soon on GitHub

PULP-Frontnet:

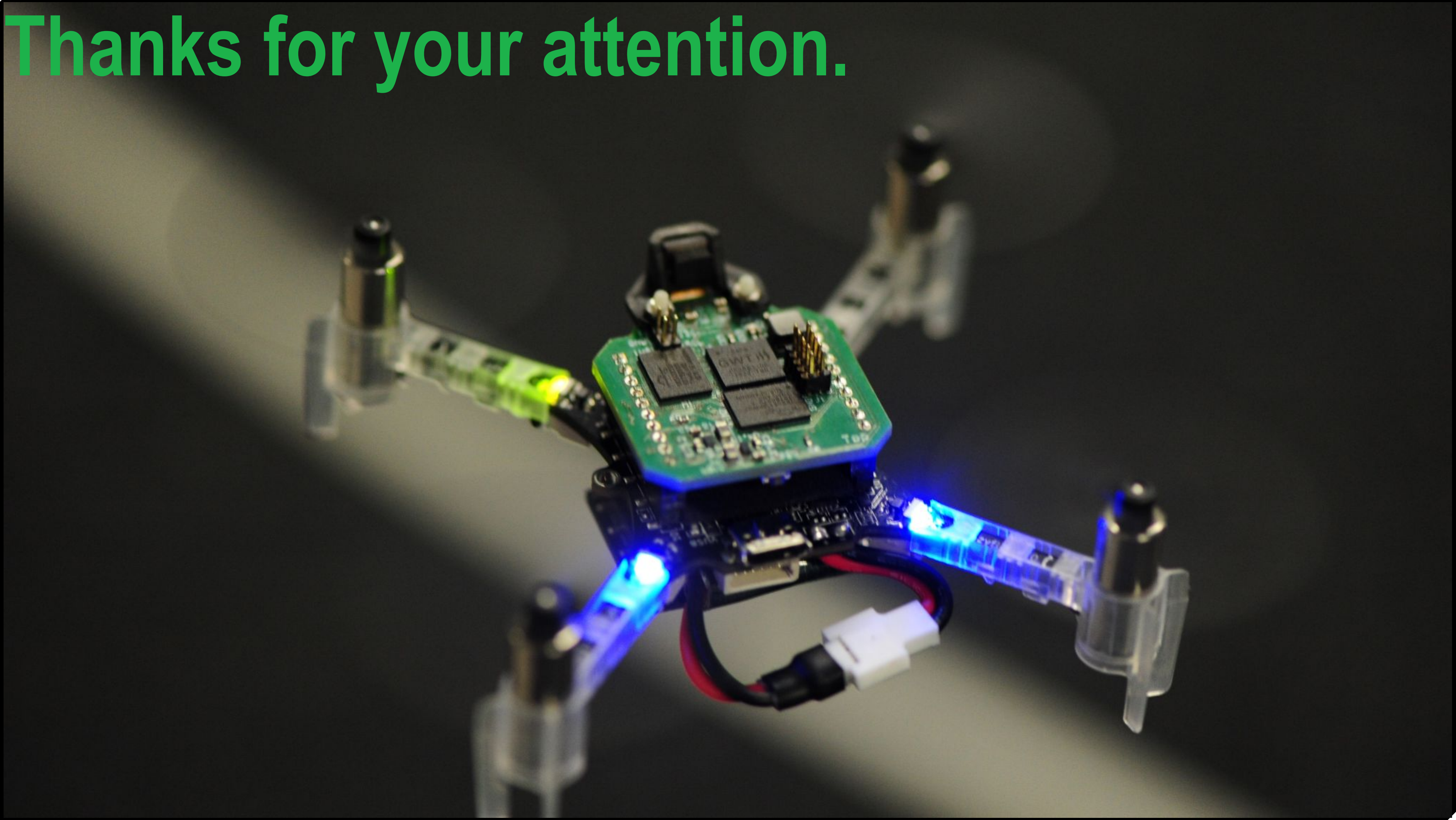


Task:	Human pose estimation
CNN:	14 / 4.3 / 4 MMAC/frame
Onboard:	48fps@20mW / 135fps@86mW
Device:	AI-Deck (GAP8)
arXiv.org	https://arxiv.org/abs/2103.10873

GitHub
Coming soon!

YouTube
Coming soon!

Thanks for your attention.

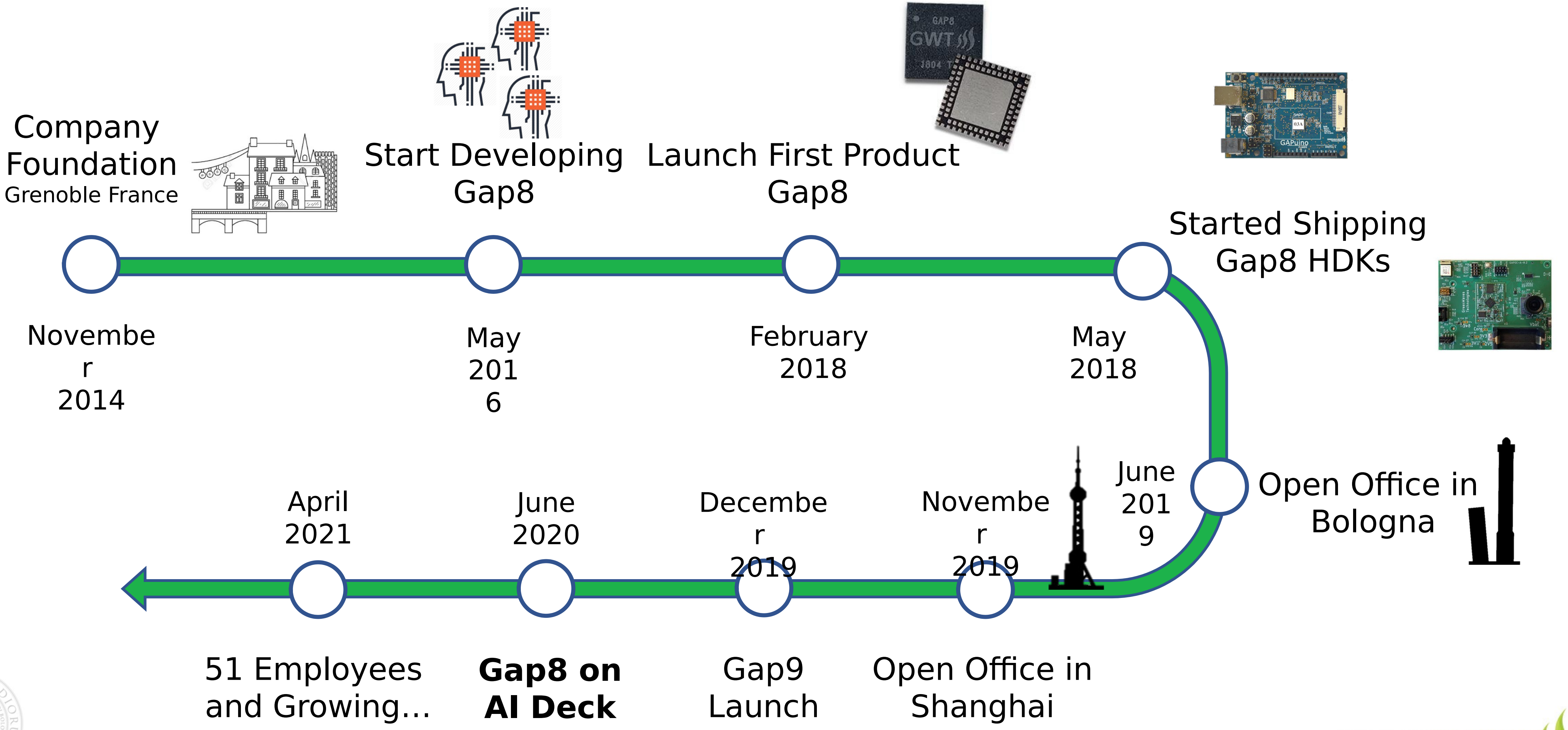


PULP PLATFORM
Open Source Hardware, the way it should be!

Bitcraze Workshop: GAP8 Architecture Overview

Lorenzo Lamberti, Hanna Müller, Vlad Niculescu, *Manuele Rusci*, Daniele Palossi

Greenwaves Technologies



Company
Foundation
Grenoble France

Novembe
r
2014

Start Developing
Gap8

May
201
6

Launch First Product
Gap8

February
2018

Started Shipping
Gap8 HDKs

May
2018

Open Office in
Bologna

June
201
9

Open Office in
Shanghai

Novembe
r
2019

Gap9
Launch

Decembe
r
2019

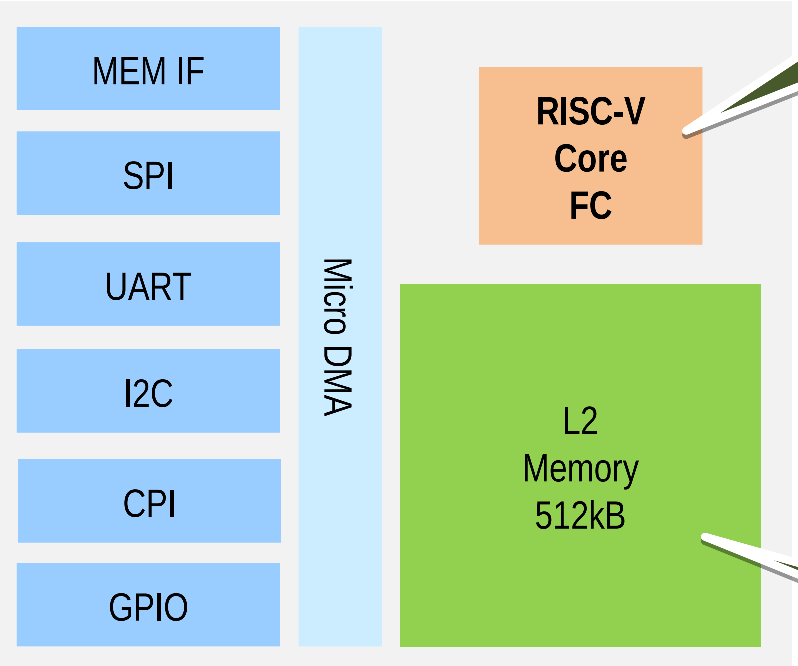
Gap8 on
AI Deck

June
2020

51 Employees
and Growing…

April
2021

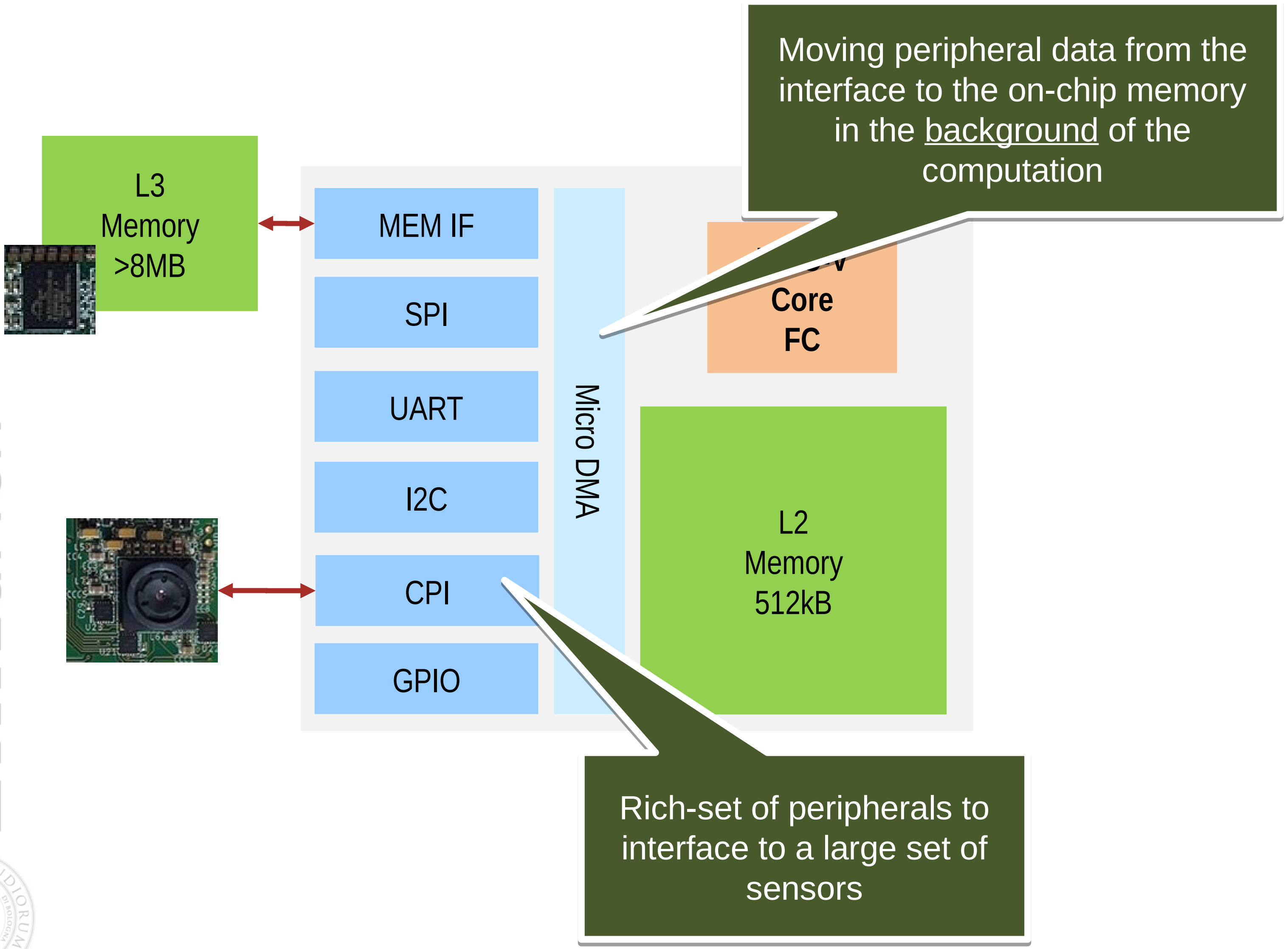
GAP8: a RISC-V IoT Application Processor



RISC-V Instruction Set optimized for **Digital Signal Processing** computation

On-Chip Memory for data storage

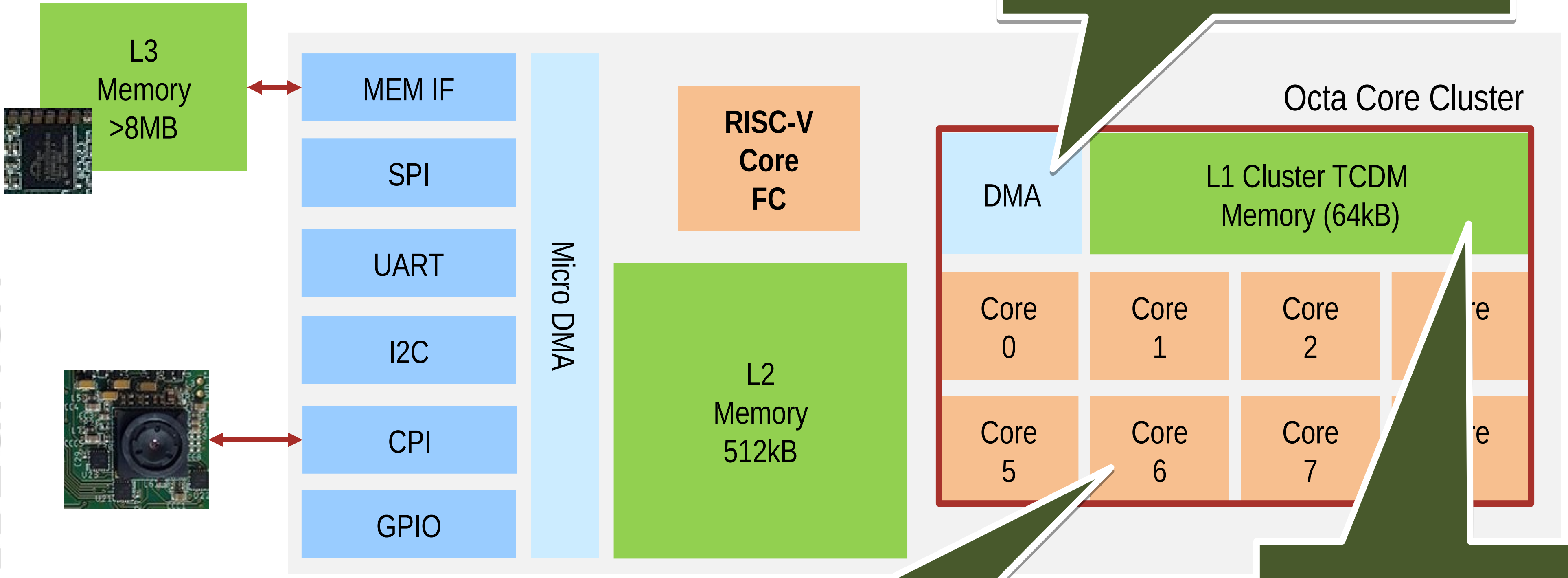
GAP8: a RISC-V IoT Application Processor



Moving peripheral data from the interface to the on-chip memory in the background of the computation

Rich-set of peripherals to interface to a large set of sensors

GAP8: a RISC-V IoT Application Processor



L3 Memory >8MB

MEM IF

SPI

UART

I2C

CPI

GPIO

Micro DMA

RISC-V Core FC

L2 Memory 512kB

Octa Core Cluster

DMA

L1 Cluster TCDM Memory (64kB)

Core 0

Core 1

Core 2

re

Core 5

Core 6

Core 7

re

Efficiently copying data from L2 memory to L1 memory

Parallel Processing for compute-intensive tasks on sensor data

Tightly-coupled On-chip memory with low-latency access

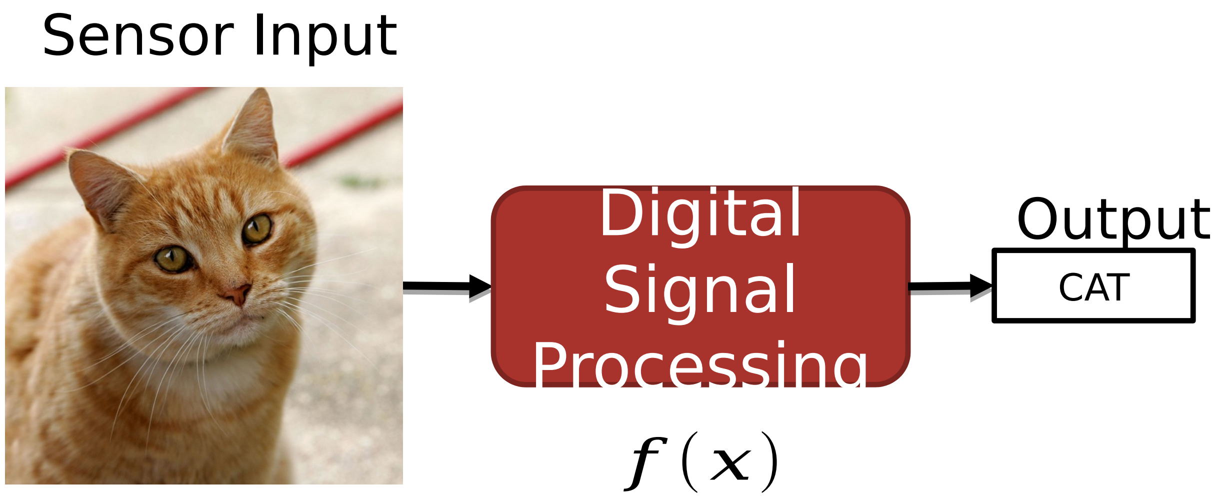
Enabling AI on the Edge

- Parallel Processing
  - Up to 9x faster than traditional single-core MCUs
  - Targeting highly-parallelizable AI workloads

- Flexibility
  - General Purpose RISC-V Cores programmable via SW

- Energy-efficiency
  - Optimized for low-power: ~100mW at 200MHz clock frequency

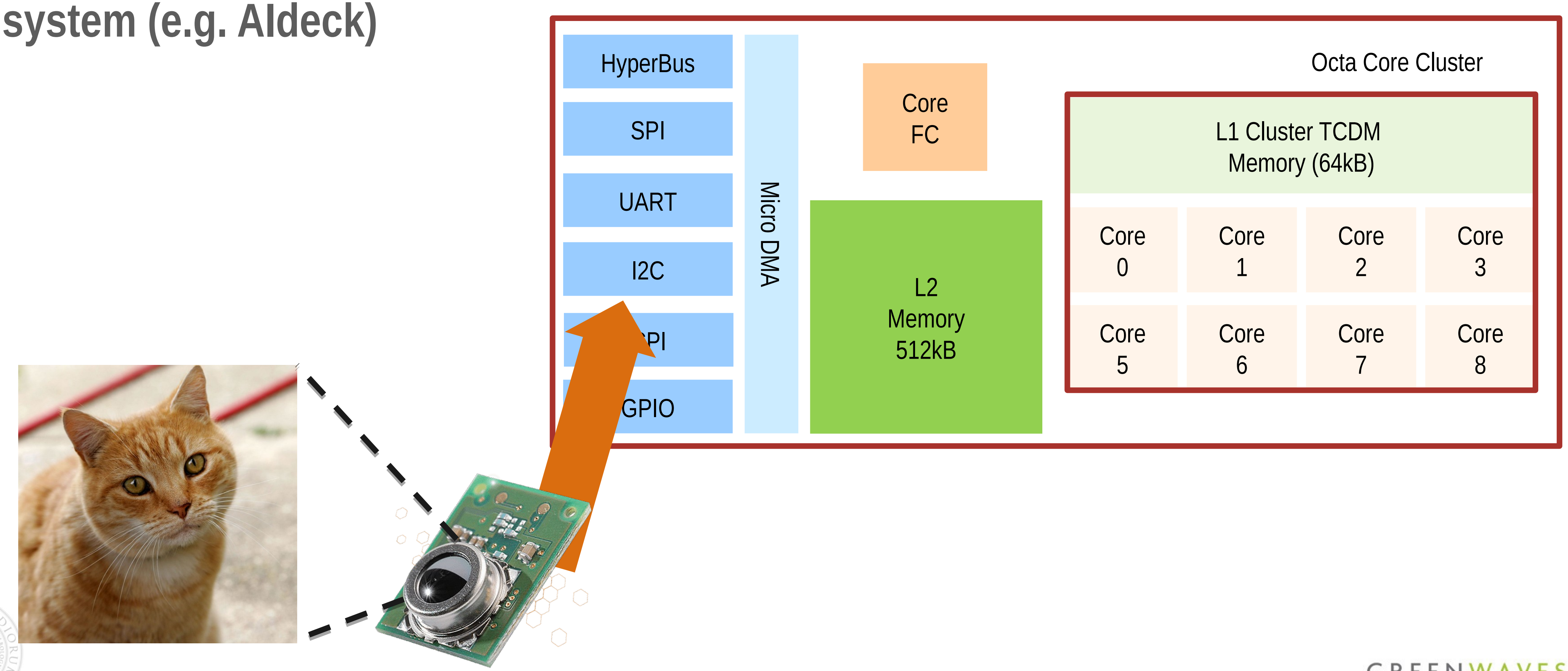
Data Analytics at the edge with GAP8



How to deploy it on a GAP8-based system?

A Low-Power Intelligent System

1) Get your GAP8-based system (e.g. AIdeck)



HyperBus

SPI

UART

I2C

CPI

GPIO

Micro DMA

Core FC

L2 Memory 512kB

Octa Core Cluster

L1 Cluster TCDM Memory (64kB)

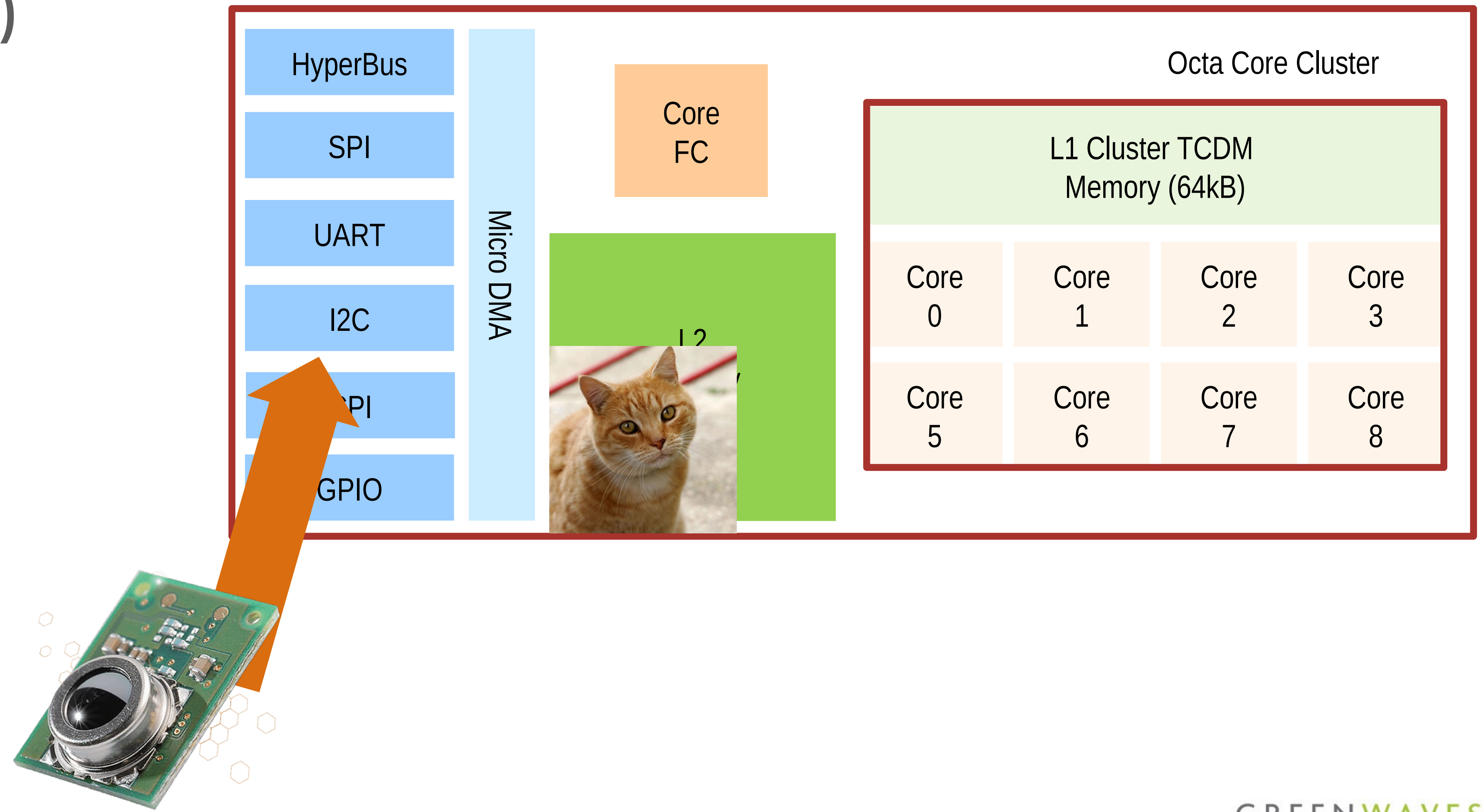
Core 0 | Core 1 | Core 2 | Core 3

Core 5 | Core 6 | Core 7 | Core 8

A Low-Power Intelligent System

1) Get your GAP8-based system (e.g. AIdeck)

2) Data Acquisition

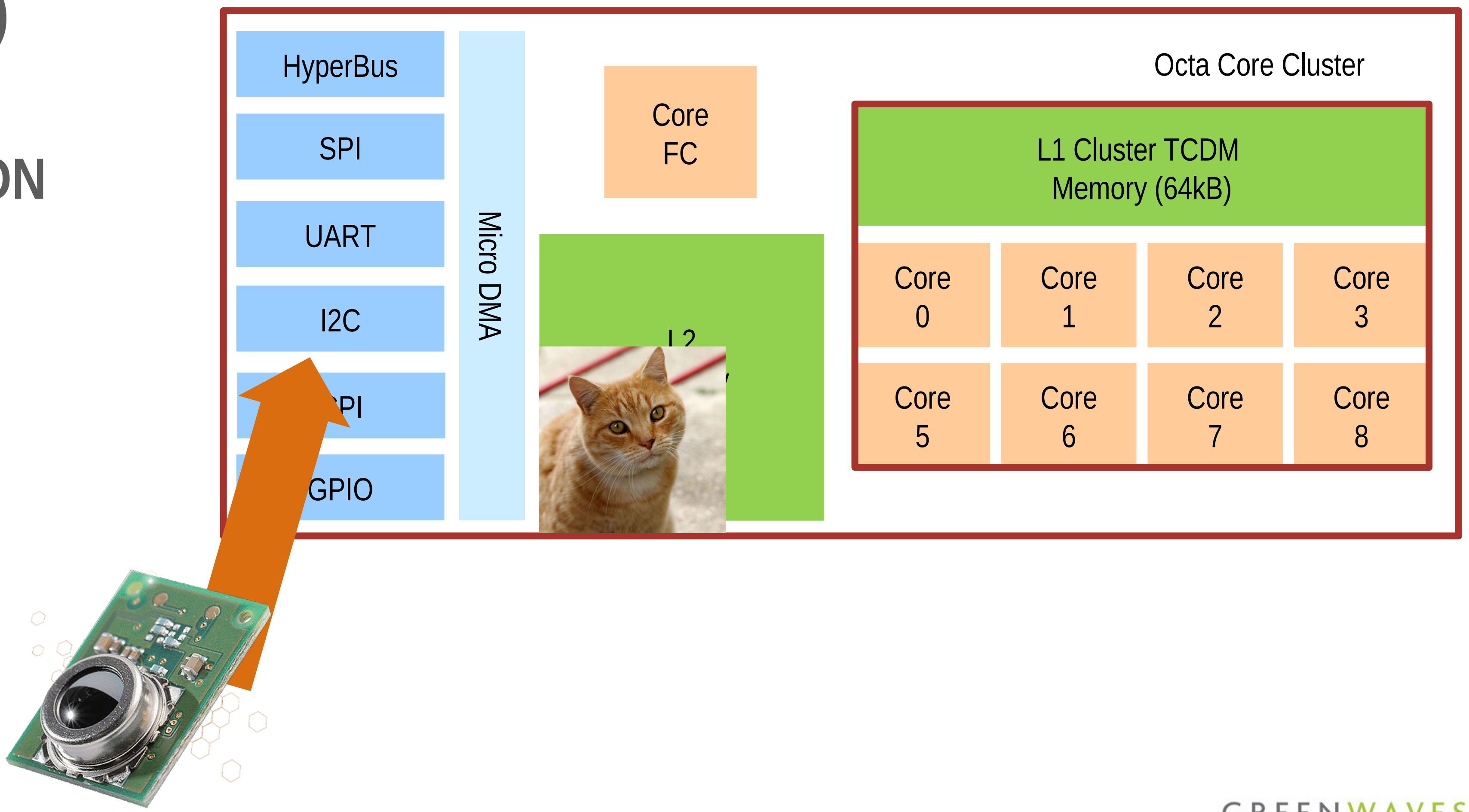


A Low-Power Intelligent System

1) Get your GAP8-based system (e.g. AIdeck)

2) Data Acquisition

3) Turn the cluster ON



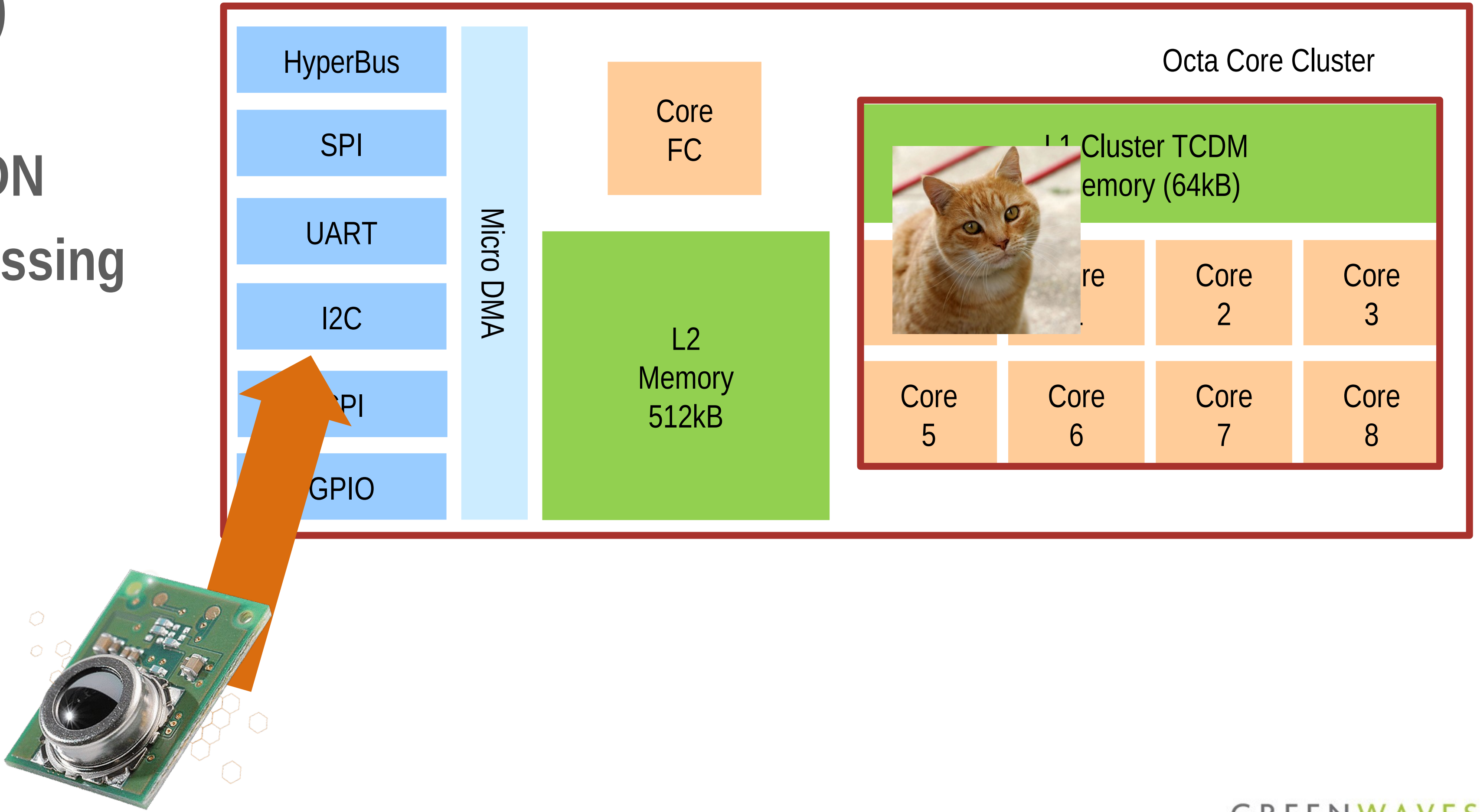
A Low-Power Intelligent System

1) Get your GAP8-based system (e.g. AIdeck)

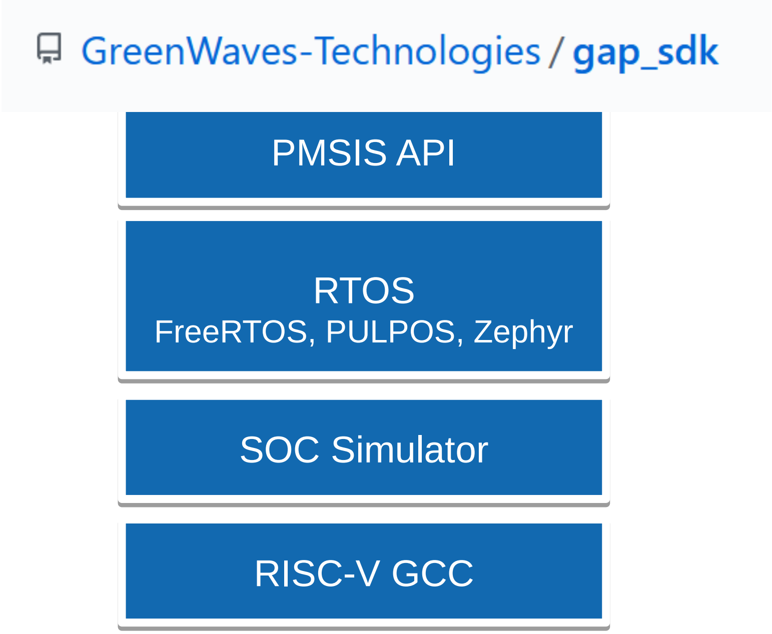
2) Data Acquisition

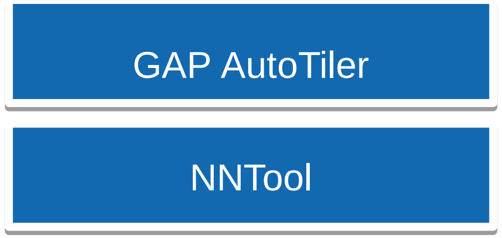
3) Turn the cluster ON

4) Run Digital Processing on Sensor Data



GAP8 – A complete solution for embedded machine learning at the very edge





- RISC-V 8 + 1 core MCU
- ISA Extensions
- Fine grained parallelism
- Application Boards

- GCC Based toolchain
- PC SoC Simulator
- Variety of different RTOS's
- PMSIS API unifies API across RTOS's

- GAP*flow* toolchain for embedded ML development

GAP NN Menu

GreenWaves-Technologies / **nn_menu**

The **Neural Network Menu** is a collection of software that implements Neural Networks on Greenwaves Application Processors (GAP). This repository contains common mobile and edge NN architecture examples, NN sample applications and full flagged reference designs.

ingredients

□Image Classification Networks (several versions of Mobilenet V1, V2, V3 minimalistic, full V3 to come)

□kws (Google Keyword Spotting)

□Mobilenet V1 from Pytorch Model

starters

□Body Detection (SSD w/ custom CNN backbone)

□Face Detection (SSD w/ custom CNN backbone)

□People Spotting (NN from MIT Visual Wakeup Words)

□Vehicle Spotting (Customization and embedding of a deep learning pipeline for visual object spotting)

main courses

Full flagged applications (aka reference designs) running on GAPoC series boards.

□ReID (on GAPoC A)

□Occupancy Management (on GAPoC B)

Bitcraze Workshop: GAP8 Architecture Overview

Thanks for listening

More about **GreenWaves Technolgies**:
https://greenwaves-technologies.com/
https://github.com/GreenWaves-Technologies/

PULP PLATFORM
Open Source Hardware, the way it should be!

Bitcraze Workshop: AI-deck

Printed circuit board overview & GAP8 SDK

Lorenzo Lamberti, *Hanna Müller*, Vlad Niculescu, Manuele Rusci, Daniele Palossi

http://pulp-platform.org @pulp_platform https://www.youtube.com/pulp_platform

How to bring intelligence to nano-drones?

We have:

- Crazyflie
  - STM32F405
    - (Flight controller)
  - NRF51822
    - (radio)

We need:

- Information about surroundings
  - Camera

  (ULP, greyscale/RGB, QVGA)

- Processing power for image processing (parallel)
  - PULP

- One QVGA greyscale image ~ 80kB

→need more memory
  - HyperMem Flash/RAM

Extra:

- WiFi Streaming

History – from the PULP-shield to the AI-deck

PULP-shield

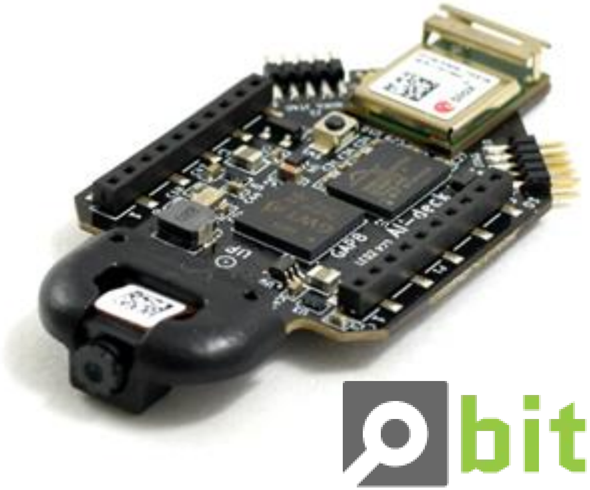
→

AI-deck

Pluggable PCB:

- ~ 5 g – 30x28 mm
- PULP **GAP8** SoC
- DRAM/Flash
- QVGA ULP HiMax
- Open source

Copyright 2019 © ETH zürich



Pluggable PCB:

- ~ 8 g – 40x28 mm
- PULP **GAP8** SoC
- 8/64 MB DRAM/Flash
- QVGA ULP HiMax
- WiFi module

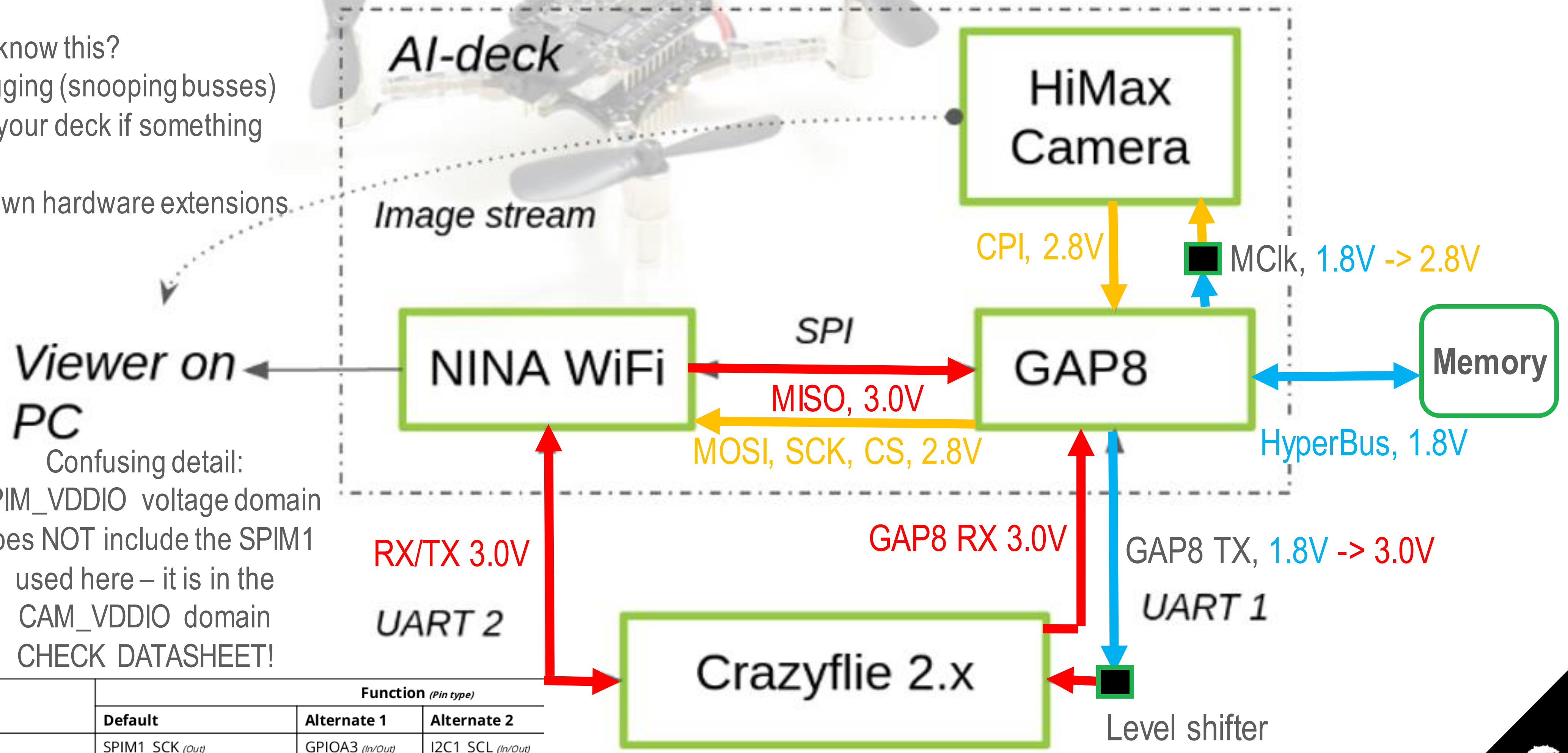
bitcraze

The AI-deck – logical connections

GAP8 has multiple voltage domains!

Why should I know this?

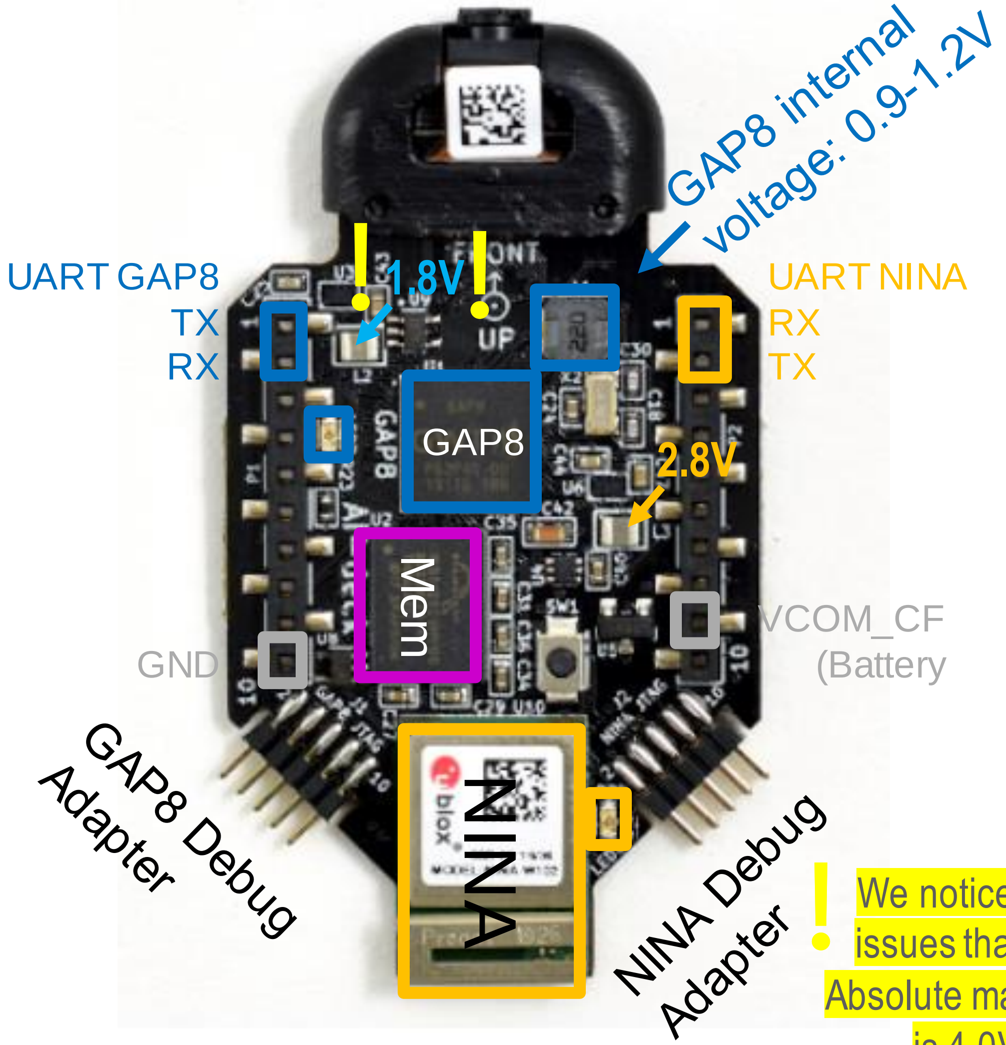
- For debugging (snooping busses)
- For fixing your deck if something broke
- For your own hardware extensions



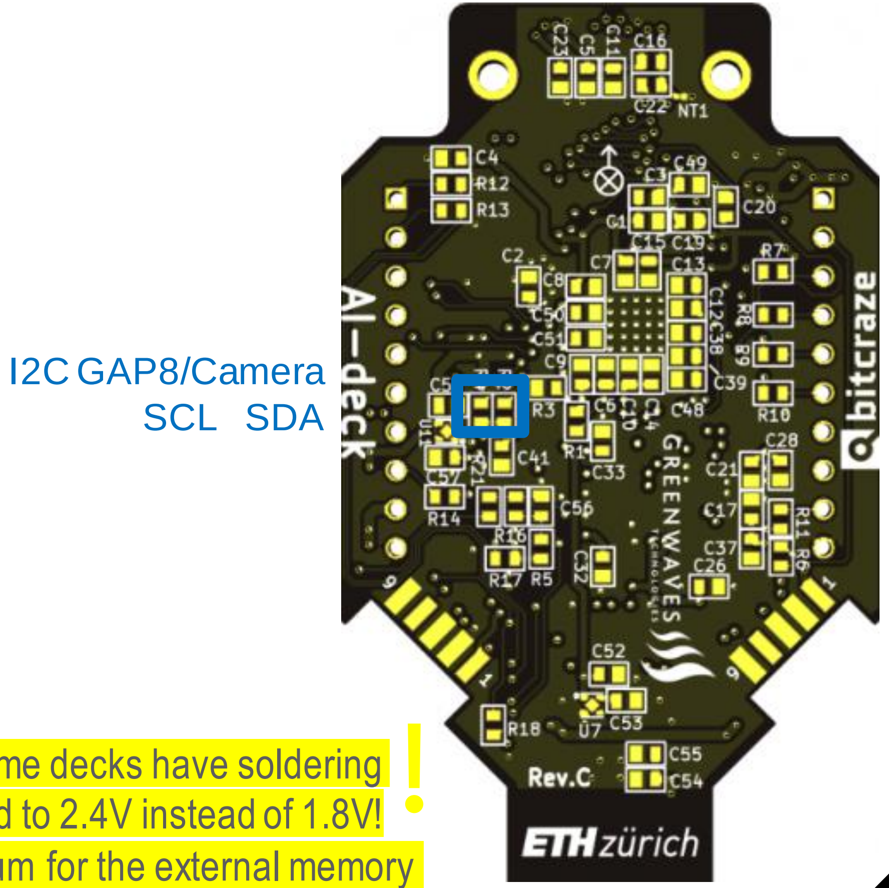
Confusing detail:
SPIM_VDDIO voltage domain does NOT include the SPIM1 used here – it is in the CAM_VDDIO domain
CHECK DATASHEET!

Position	Voltage Ref	Function *(Pin type)*		
		Default	Alternate 1	Alternate 2
B4	CAM_VDDIO	SPIM1_SCK *(Out)*	GPIOA3 *(In/Out)*	I2C1_SCL *(In/Out)*
A3	CAM_VDDIO	ORCA_TXSYNC *(In)*	GPIOA0 *(In/Out)*	SPIM1_CS0 *(Out)*
B2	CAM_VDDIO	ORCA_RXSYNC *(In)*	GPIOA1 *(In/Out)*	SPIM1_CS1 *(Out)*

The AI-deck

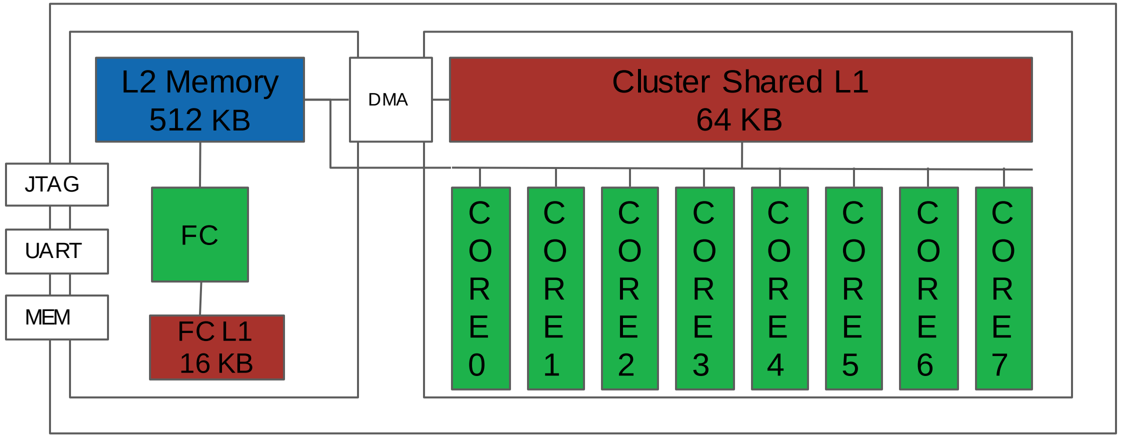


Capacitors – a lot of capacitors and some resistors



We noticed some decks have soldering issues that lead to 2.4V instead of 1.8V! Absolute maximum for the external memory is 4.0V, supply range up to 2.0V.

How to program GAP8? GAP-SDK!



Example: to queue a buffer that receives camera samples:
In PMSIS BSP: static void pi_camera_capture _async()
Uses a function to queue a buffer that receives CPI samples:
In PMSIS API: static void pi_cpi_capture_async()
The OS is on top – you can define a callback task from your OS

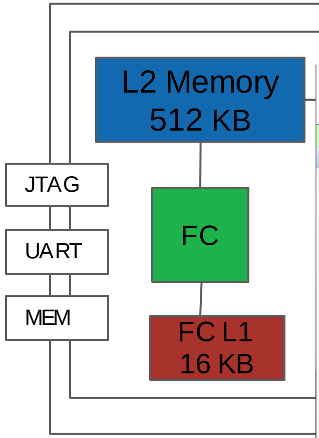
GAP-SDK provides:

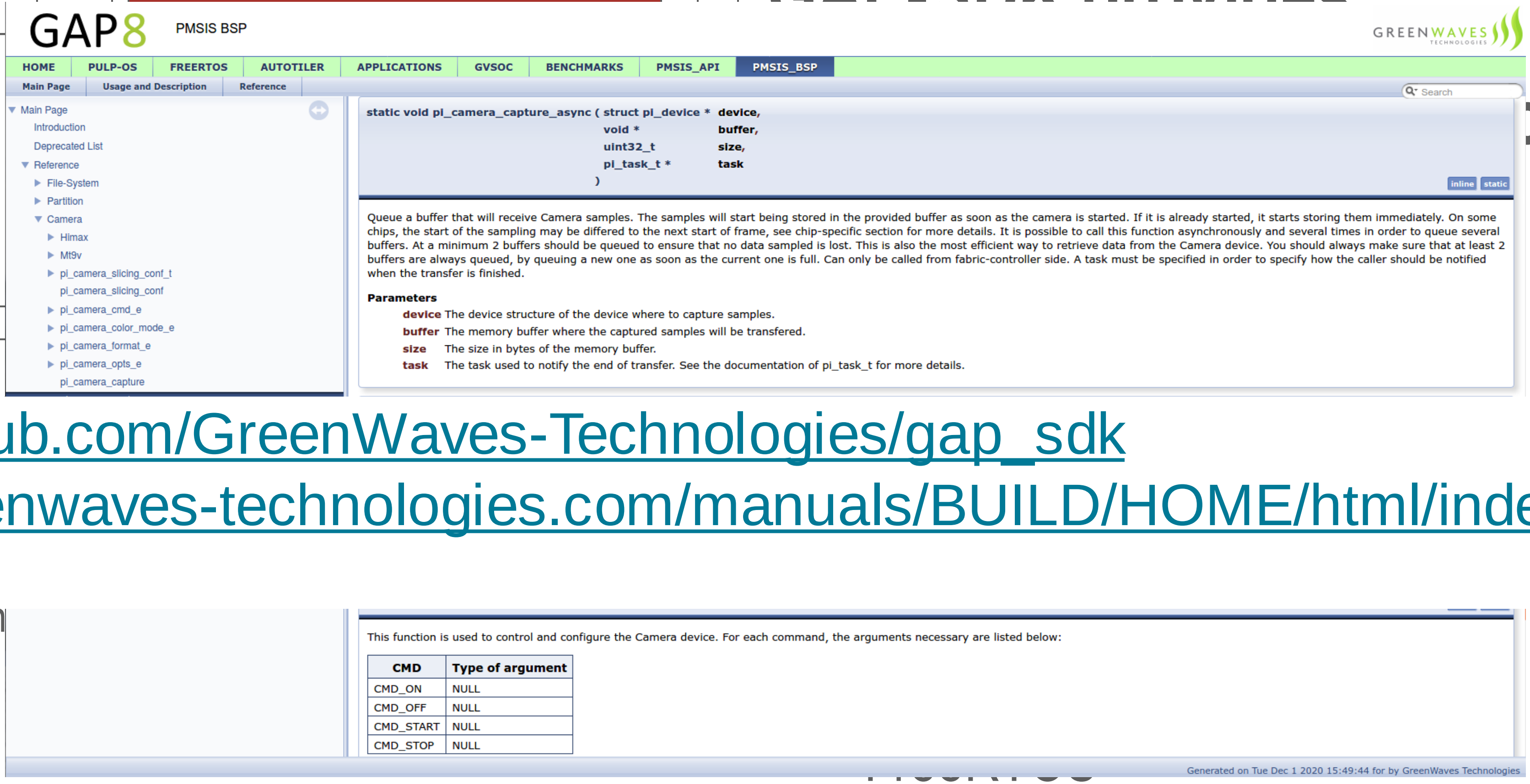
- GAP8 RISCV GNU toolchain:
  - Program/control gap8
  - Use gdb
  - Program external HyperFlash
  - Virtual platform (gvsoc)
- Operating Systems
  - PulpOS
  - FreeRTOS
  - PMSIS API/BSP (common driver)

How to program GAP8? GAP-SDK!

GAP-SDK provides:

chain:





```
static void pi_camera_capture_async ( struct pi_device *  device,
                                      void *              buffer,
                                      uint32_t            size,
                                      pi_task_t *         task
                                    )
```

Queue a buffer that will receive Camera samples. The samples will start being stored in the provided buffer as soon as the camera is started. If it is already started, it starts storing them immediately. On some chips, the start of the sampling may be differed to the next start of frame, see chip-specific section for more details. It is possible to call this function asynchronously and several times in order to queue several buffers. At a minimum 2 buffers should be queued to ensure that no data sampled is lost. This is also the most efficient way to retrieve data from the Camera device. You should always make sure that at least 2 buffers are always queued, by queuing a new one as soon as the current one is full. Can only be called from fabric-controller side. A task must be specified in order to specify how the caller should be notified when the transfer is finished.

Parameters

device	The device structure of the device where to capture samples.
buffer	The memory buffer where the captured samples will be transfered.
size	The size in bytes of the memory buffer.
task	The task used to notify the end of transfer. See the documentation of pi_task_t for more details.

https://github.com/GreenWaves-Technologies/gap_sdk

https://greenwaves-technologies.com/manuals/BUILD/HOME/html/index.html

The OS is on

This function is used to control and configure the Camera device. For each command, the arguments necessary are listed below:

CMD	Type of argument
CMD_ON	NULL
CMD_OFF	NULL
CMD_START	NULL
CMD_STOP	NULL

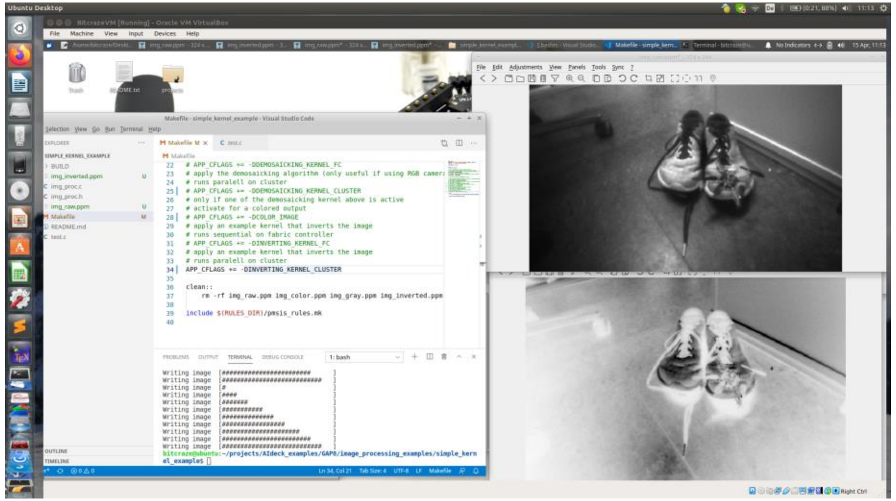
Generated on Tue Dec 1 2020 15:49:44 for by GreenWaves Technologies

- PMSIS API/BSP (common driver)

How to program GAP8?

Easiest way: Bitcraze VM!

- Gap-sdk is installed! Open a terminal and get started :)
- Also: All tools installed to compile for and flash the STM32 and nRF on the Crazyflie (Ubuntu, gnu-arm-none-eabi toolchain, python dependencies, KiCad, and many more)
- **Update your Crazyflie 2.x** to the most recent firmware before trying to program GAP8!



```
# APP_CFLAGS += -DDEMOSAICKING_KERNEL_FC
# apply the demosaicking algorithm (only useful if using RGB camer
# runs paralell on cluster
# APP_CFLAGS += -DDEMOSAICKING_KERNEL_CLUSTER
# only if one of the demosaicking kernel above is active
# activate for a colored output
# APP_CFLAGS += -DCOLOR_IMAGE
# apply an example kernel that inverts the image
# runs sequential on fabric controller
# APP_CFLAGS += -DINVERTING_KERNEL_FC
# apply an example kernel that inverts the image
# runs paralell on cluster
APP_CFLAGS += -DINVERTING_KERNEL_CLUSTER

clean::
    rm -rf img_raw.ppm img_color.ppm img_gray.ppm img_inverted.ppm

include $(RULES_DIR)/pmsis_rules.mk
```

```
Writing image  [########################        ]
Writing image  [############################    ]
Writing image  [#                               ]
Writing image  [####                            ]
Writing image  [#######                         ]
Writing image  [###########                     ]
Writing image  [##############                  ]
Writing image  [#################               ]
Writing image  [#####################           ]
Writing image  [########################        ]
Writing image  [############################    ]
bitcraze@ubuntu:~/projects/AIdeck_examples/GAP8/image_processing_examples/simple_kern
el_example$ 
```

Ubuntu Desktop
BitcrazeVM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Makefile - simple_kernel_example - Visual Studio Code
Selection View Go Run Terminal Help
EXPLORER
SIMPLE_KERNEL_EXAMPLE
BUILD
img_inverted.ppm
img_proc.c
img_proc.h
img_raw.ppm
Makefile
README.md
test.c
Makefile M
test.c
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: bash
OUTLINE
TIMELINE
Ln 34, Col 21 Tab Size: 4 UTF-8 LF Makefile
File Edit Adjustments View Panels Tools Sync ?
Trash
README.txt
projects

Important: in the VM you need to use docker!
Some commands are preconfigured in the .bashrc file
Just typing "make clean all run" like on a native install will not work. Type "gap_run" instead

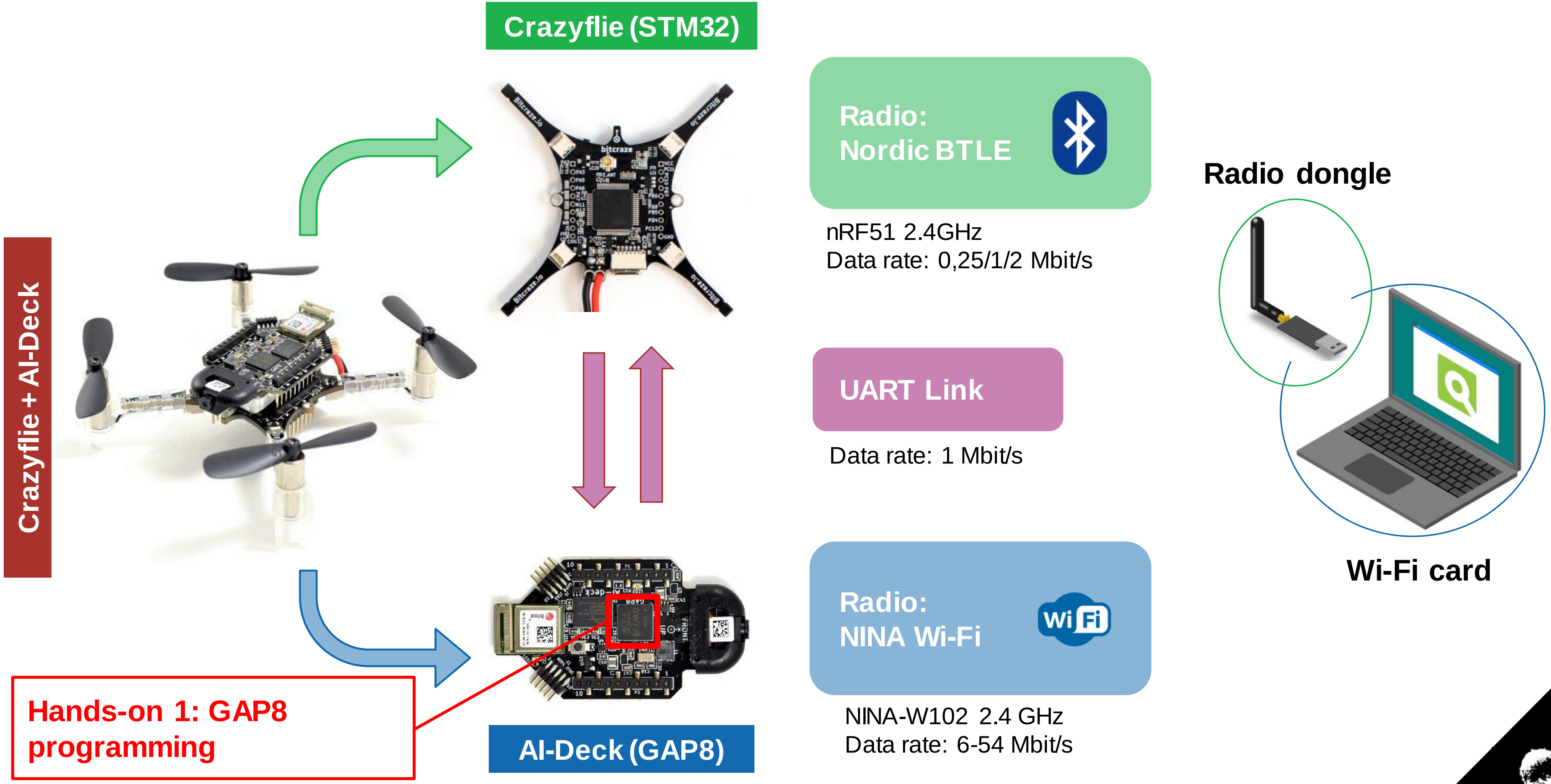
PULP PLATFORM
Open Source Hardware, the way it should be!

Bitcraze Workshop: *Hands-on Session 1*
'Hello World' on the AI-deck

Lorenzo Lamberti, *Hanna Müller*, Vlad Niculescu, Manuele Rusci,
Daniele Palossi

http://pulp-platform.org @pulp_platform https://www.youtube.com/pulp_platform

The AI-Deck



Crazyflie (STM32)

Crazyflie + AI-Deck

Radio: Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

Radio: NINA Wi-Fi

NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

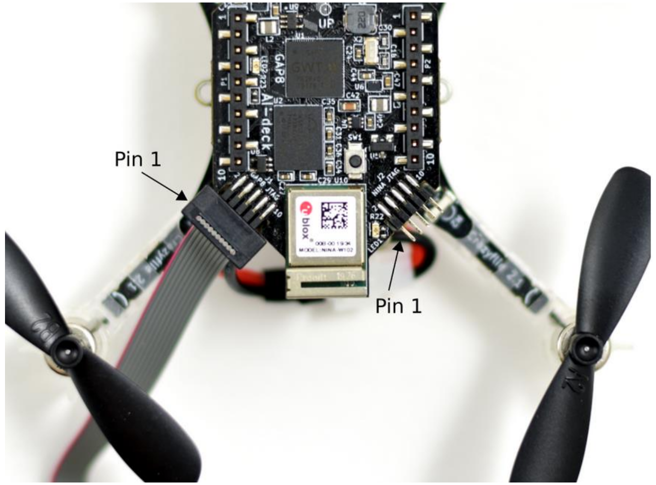
AI-Deck (GAP8)

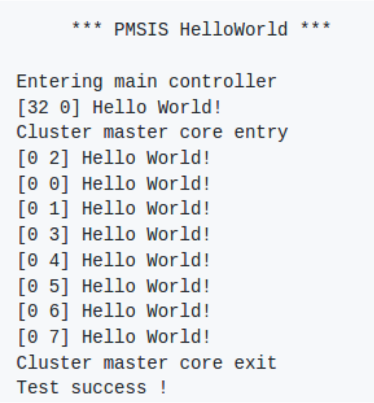
Hands-on 1: GAP8 programming

Radio dongle

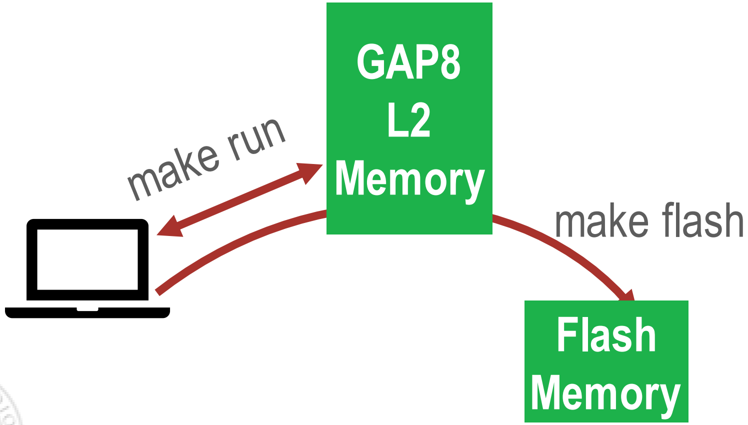
Wi-Fi card

Hands-on: Hello World!





```
        *** PMSIS HelloWorld ***

Entering main controller
[32 0] Hello World!
Cluster master core entry
[0 2] Hello World!
[0 0] Hello World!
[0 1] Hello World!
[0 3] Hello World!
[0 4] Hello World!
[0 5] Hello World!
[0 6] Hello World!
[0 7] Hello World!
Cluster master core exit
Test success !
```



Code is always executed from L2!
(Volatile memory – if you lose power, you lose the code)
You can store your code in flash, then the bootloader loads the code on startup

"gap_run" in the VM, no command configured for gvsoc, you can add it yourself to the .bashrc script

Open a terminal

1. cd $GAP_SDK_HOME

   Env variable set by step 2

2. source configs/ai_deck.sh

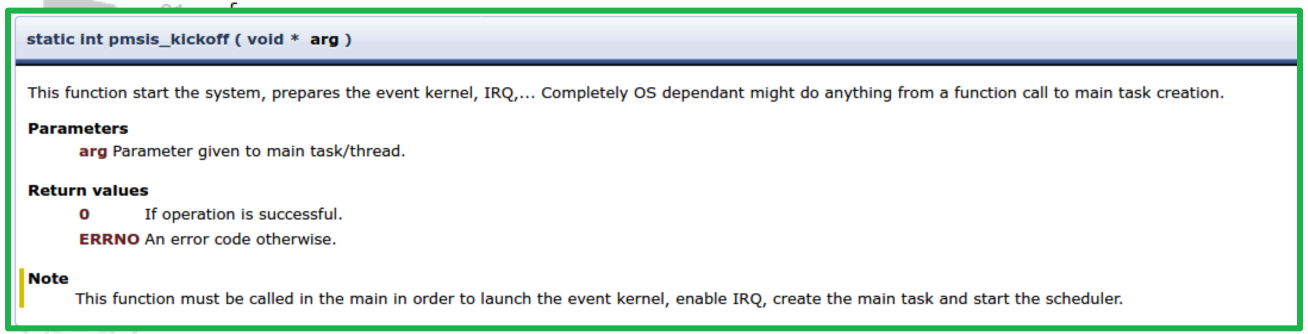
   Is done already in VM

1. cd examples/pmsis/helloworld
2. Connect JTAG
3. Power on drone/AI-deck
4. Compile and run

```
# Run on GVSoC
make clean all run platform=gvsoc

# Run on real board
make clean all run platform=board
```

Hands-on: Hello World!

```c
/* PMSIS includes */
#include "pmsis.h"

/* Task executed by cluster cores. */
void cluster_helloworld(void *arg)
{
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    printf("[%d %d] Hello World!\n", cluster_id, core_id);
}

/* Cluster main entry, executed by core 0. */
void cluster_delegate(void *arg)
{
    printf("Cluster master core entry\n");
    /* Task dispatch to cluster cores. */
    pi_cl_team_fork(pi_cl_cluster_nb_cores(), cluster_helloworld, arg);
    printf("Cluster master core exit\n");
}

void helloworld(void)
{
```



static int pmsis_kickoff (void * arg)

This function start the system, prepares the event kernel, IRQ,... Completely OS dependant might do anything from a function call to main task creation.

Parameters

arg Parameter given to main task/thread.

Return values

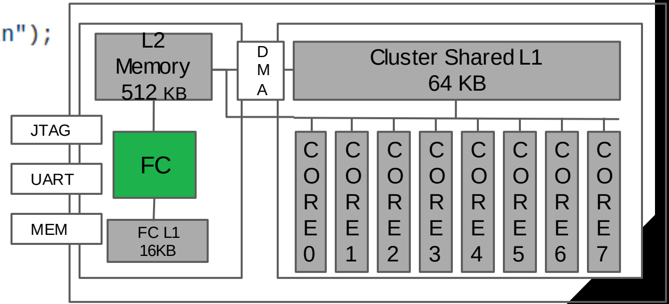
0 If operation is successful.

ERRNO An error code otherwise.

Note

This function must be called in the main in order to launch the event kernel, enable IRQ, create the main task and start the scheduler.

```c
    /* Init cluster configuration structure. */
    pi_cluster_conf_init(&cl_conf);
    cl_conf.id = 0;                /* Set cluster ID. */
    /* Configure & open cluster. */
    pi_open_from_conf(&cluster_dev, &cl_conf);
    if (pi_cluster_open(&cluster_dev))
    {
        printf("Cluster open failed !\n");
        pmsis_exit(-1);
    }

    /* Prepare cluster task and send it to cluster. */
    struct pi_cluster_task cl_task = {0};
    cl_task.entry = cluster_delegate;
    cl_task.arg = NULL;

    pi_cluster_send_task_to_cl(&cluster_dev, &cl_task);

    pi_cluster_close(&cluster_dev);

    printf("Test success !\n");

    pmsis_exit(errors);
}

/* Program Entry. */
int main(void)
{
    printf("\n\n\t *** PMSIS HelloWorld ***\n\n");
    return pmsis_kickoff((void *) helloworld);
}
```



Hands-on: Hello World!

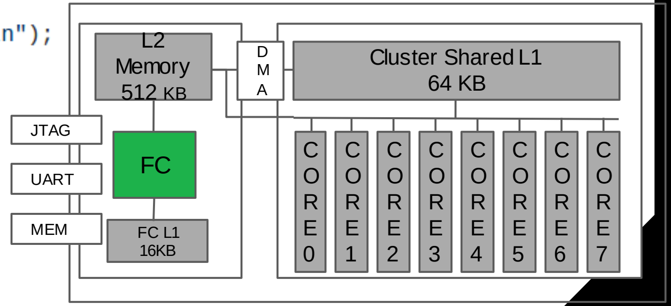
```c
/* PMSIS includes */
#include "pmsis.h"

/* Task executed by cluster cores. */
void cluster_helloworld(void *arg)
{
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    printf("[%d %d] Hello World!\n", cluster_id, core_id);
}

/* Cluster main entry, executed by core 0. */
void cluster_delegate(void *arg)
{
    printf("Cluster master core entry\n");
    /* Task dispatch to cluster cores. */
    pi_cl_team_fork(pi_cl_cluster_nb_cores(), cluster_helloworld, arg);
    printf("Cluster master core exit\n");
}

void helloworld(void)
{
    printf("Entering main contro

    uint32_t errors = 0;
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    printf("[%d %d] Hello World!\n", cluster_id, core_id);

    struct pi_device cluster_dev = {0};
    struct pi_cluster_conf cl_conf = {0};
```

We are on the Fabric controller
Cluster ID is 32 per default.
We only have core 0.

Init cluster config to default values
Set id manually
Point cluster device to your config
Open cluster (power up), blocking

```c
    /* Init cluster configuration structure. */
    pi_cluster_conf_init(&cl_conf);
    cl_conf.id = 0;                /* Set cluster ID. */
    /* Configure & open cluster. */
    pi_open_from_conf(&cluster_dev, &cl_conf);
    if (pi_cluster_open(&cluster_dev))
    {
        printf("Cluster open failed !\n");
        pmsis_exit(-1);
    }

    /* Prepare cluster task and send it to c
    struct pi_cluster_task cl_task = {0};
    cl_task.entry = cluster_delegate;
    cl_task.arg = NULL;

    pi_cluster_send_task_to_cl(&cluster_dev, &cl_task);

    pi_cluster_close(&cluster_dev);

    printf("Test success !\n");

    pmsis_exit(errors);
}

/* Program Entry. */
int main(void)
{
    printf("\n\n\t *** PMSIS HelloWorld ***\n\n");
    return pmsis_kickoff((void *) helloworld);
}
```

Configure cluster task
Send task to cluster
(blocking, also exists
in async)



Hands-on: Hello World!

```c
/* PMSIS includes */
#include "pmsis.h"

/* Task executed by cluster cores. */
void cluster_helloworld(void *arg)
{
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    printf("[%d %d] Hello World!\n", cluster_id, core_id);
}

/* Cluster main entry, executed by core 0. */
void cluster_delegate(void *arg)
{
    printf("Cluster master core entry\n");
    /* Task dispatch to cluster cores. */
    pi_cl_team_fork(pi_cl_cluster_nb_cores(), cluster_helloworld, arg);
    printf("Cluster master core exit\n");
}

void helloworld(void)
{
    printf("Entering main contro

    uint32_t errors = 0;
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    printf("[%d %d] Hello World!\n", cluster_id, core_id);

    struct pi_device cluster_dev = {0};
    struct pi_cluster_conf cl_conf = {0};
```

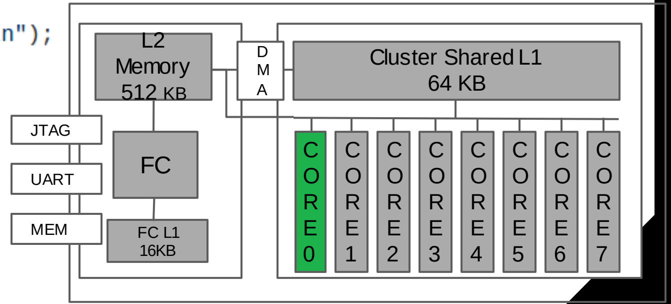
We are only on core 0 of the cluster yet

We are on the Fabric controller
Cluster ID is 32 per default.
We only have core 0.

Init cluster config to default values
Set id manually
Point cluster device to your config
Open cluster (power up), blocking

```c
    /* Init cluster configuration structure. */
    pi_cluster_conf_init(&cl_conf);
    cl_conf.id = 0;                /* Set cluster ID. */
    /* Configure & open cluster. */
    pi_open_from_conf(&cluster_dev, &cl_conf);
    if (pi_cluster_open(&cluster_dev))
    {
        printf("Cluster open failed !\n");
        pmsis_exit(-1);
    }

    /* Prepare cluster task and send it to c
    struct pi_cluster_task cl_task = {0};
    cl_task.entry = cluster_delegate;
    cl_task.arg = NULL;

    pi_cluster_send_task_to_cl(&cluster_dev, &cl_task);

    pi_cluster_close(&cluster_dev);

    printf("Test success !\n");

    pmsis_exit(errors);
}

/* Program Entry. */
int main(void)
{
    printf("\n\n\t *** PMSIS HelloWorld ***\n\n");
    return pmsis_kickoff((void *) helloworld);
}
```

Configure cluster task
Send task to cluster
(blocking, also exists in async)



Hands-on: Hello World!

```
/* PMSIS includes */
#include "pmsis.h"

uster cores. */
void cluster_helloworld(void *arg)
{
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    printf("[%d %d] Hello World!\n", cluster_id, core_id);
}

/* Cluster main entry, executed by core 0. */
void cluster_delegate(void *arg)
{
    printf("Cluster master core entry\n");
    /* Task dispatch to cluster cores. */
    pi_cl_team_fork(pi_cl_cluster_nb_cores(), cluster_helloworld, arg);
    printf("Cluster master core exit\n");
}

void helloworld(void)
{
    printf("Entering main contro

    uint32_t errors = 0;
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    printf("[%d %d] Hello World!\n", cluster_id, core_id);

    struct pi_device cluster_dev = {0};
    struct pi_cluster_conf cl_conf = {0};
```

Print cluster and core ID

We are only on core 0 of the cluster yet

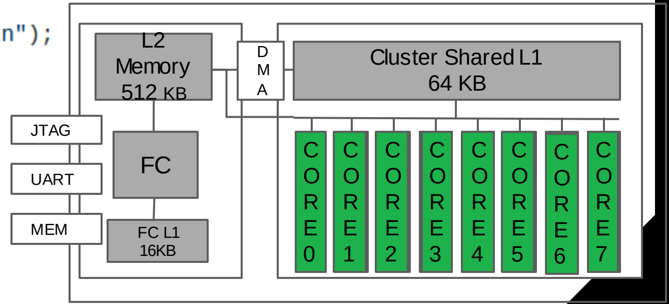
Fork to number of cluster cores available

We are on the Fabric controller
Cluster ID is 32 per default.
We only have core 0.

Init cluster config to default values
Set id manually
Point cluster device to your config
Open cluster (power up), blocking

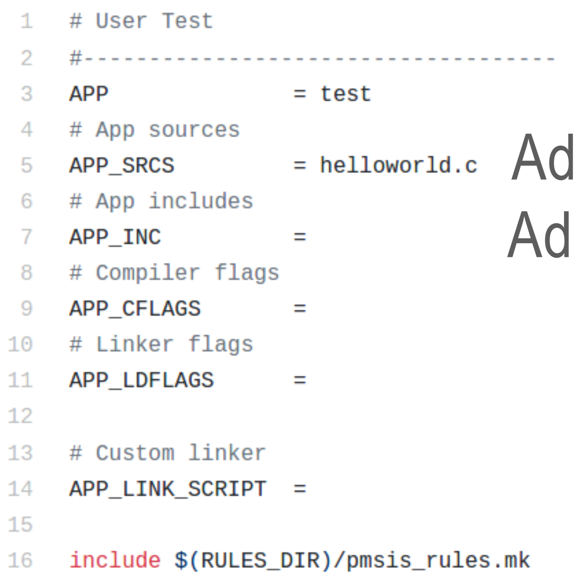
```
    /* Init cluster configuration structure. */
    pi_cluster_conf_init(&cl_conf);
    cl_conf.id = 0;                /* Set cluster ID. */
    /* Configure & open cluster. */
    pi_open_from_conf(&cluster_dev, &cl_conf);
    if (pi_cluster_open(&cluster_dev))
    {
        printf("Cluster open failed !\n");
        pmsis_exit(-1);
    }

    /* Prepare cluster task and send it to c
    struct pi_cluster_task cl_task = {0};
    cl_task.entry = cluster_delegate;
    cl_task.arg = NULL;

    pi_cluster_send_task_to_cl(&cluster_dev, &cl_task);

    pi_cluster_close(&cluster_dev);

    printf("Test success !\n");

    pmsis_exit(errors);
}

/* Program Entry. */
int main(void)
{
    printf("\n\n\t *** PMSIS HelloWorld ***\n\n");
    return pmsis_kickoff((void *) helloworld);
}
```

Configure cluster task
Send task to cluster
(blocking, also exists in async)



Hands-on: Hello World!

Makefile



```
# User Test
#------------------------------------
APP              = test
# App sources
APP_SRCS         = helloworld.c
# App includes
APP_INC          =
# Compiler flags
APP_CFLAGS       =
# Linker flags
APP_LDFLAGS      =

# Custom linker
APP_LINK_SCRIPT  =

include $(RULES_DIR)/pmsis_rules.mk
```

Add sources here

Add directories to include (header files) here

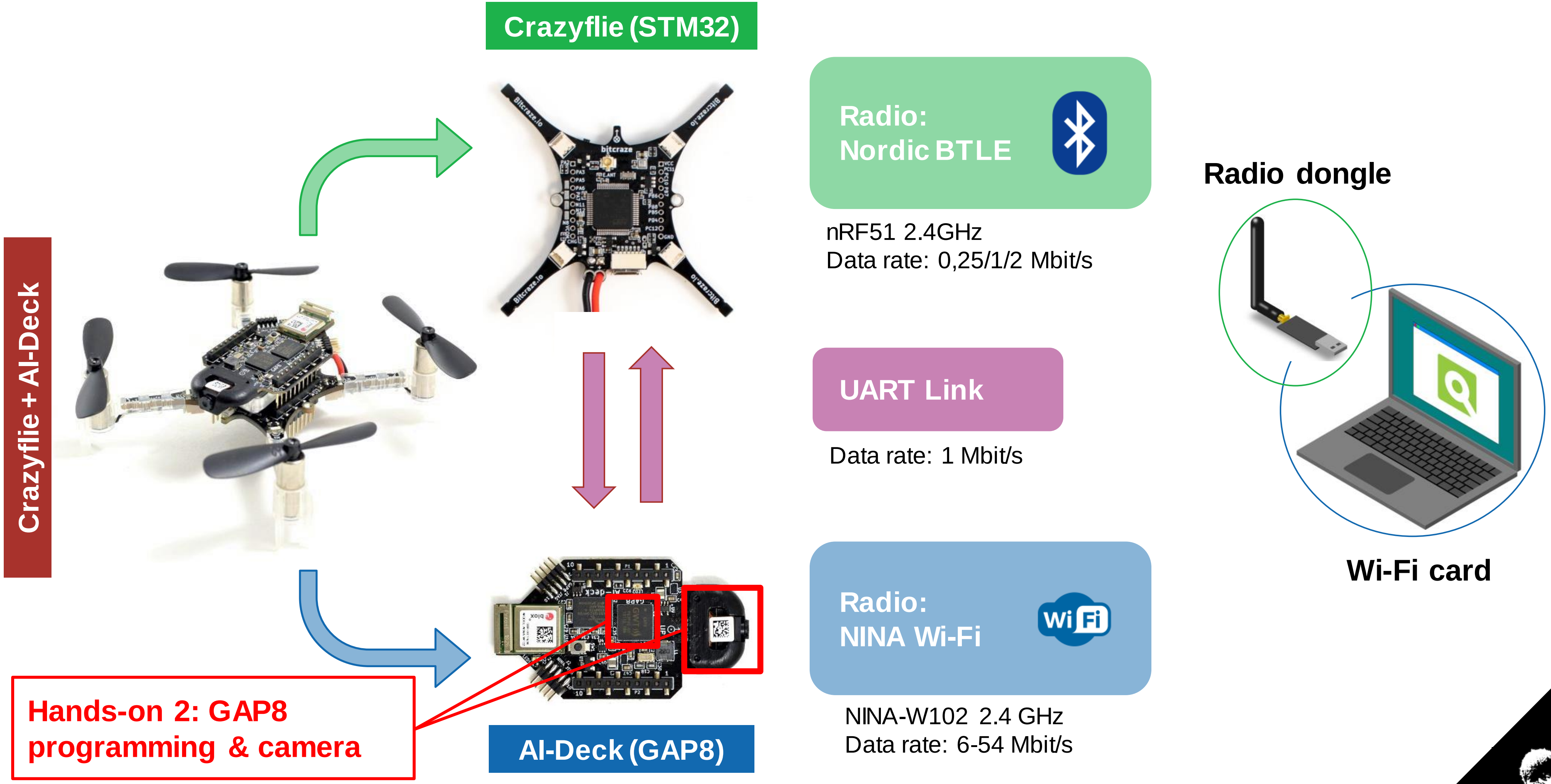
PULP PLATFORM
Open Source Hardware, the way it should be!

Bitcraze Workshop: *Hands-on Session 2*
Image acquisition and parallel image filter

Lorenzo Lamberti, *Hanna Müller*, Vlad Niculescu, Manuele Rusci, Daniele Palossi

http://pulp-platform.org @pulp_platform https://www.youtube.com/pulp_platform

The AI-Deck



Crazyflie (STM32)

Crazyflie + AI-Deck

Radio: Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

Radio: NINA Wi-Fi

NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

Radio dongle

Wi-Fi card

AI-Deck (GAP8)

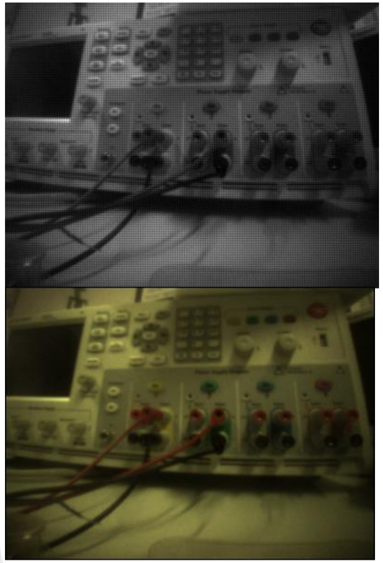
Hands-on 2: GAP8 programming & camera

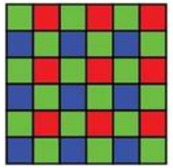
Hands-on: Image acquisition and filtering

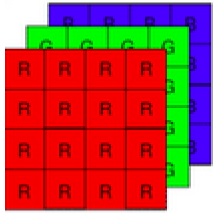
1. git clone https://github.com/bitcraze/AIdeck_examples
2. set up your gap-sdk (source configs/ai_deck.sh)
3. Go to GAP8/image_processing_examples/simple_kernel_example
4. Compile and run the code (make clean all run platform=board or gap_run in the VM)
5. You can configure some flags in the Makefile

First: execution flow using demosaicking on the fabric controller as example
Then: parallelization with inverting an image on the cluster.

The code is simplified on the slides (but functional)







Demosaicking Fabric controller

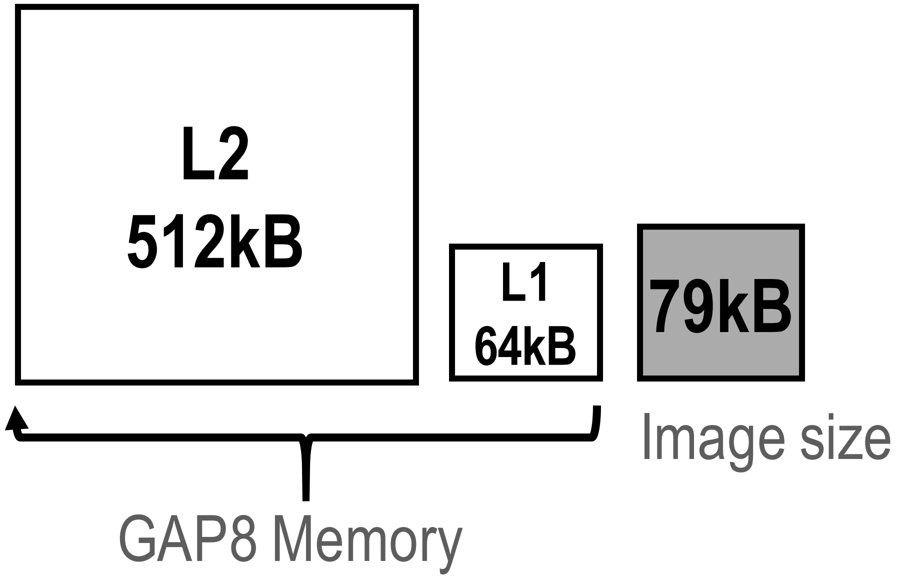
Inverting Fabric controller

Demosaicking Cluster

Inverting Cluster

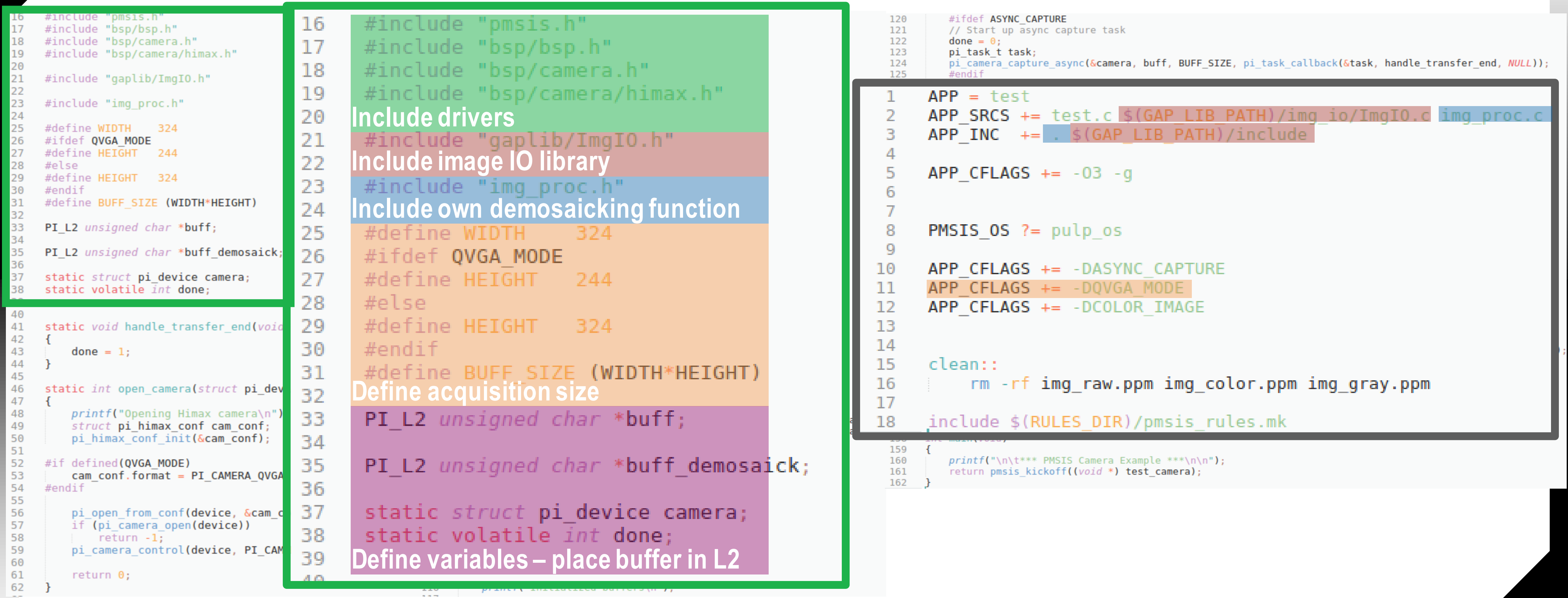
Hands-on: Image acquisition and filtering

Before we start, let's think about memory:
How many QVGA images could you have on GAP8 at the same time?
Does it matter if they are colored or grey? Hint:
GAP8 L2 Memory:512kB



Not even a single grey scale one on L1.
6 grey scale or 2 RGB in L2 – BUT do not forget, you also need space for the code in L2!

Hands-on: Image acquisition and filtering



```c
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera.h"
#include "bsp/camera/himax.h"

#include "gaplib/ImgIO.h"

#include "img_proc.h"

#define WIDTH    324
#ifdef QVGA_MODE
#define HEIGHT   244
#else
#define HEIGHT   324
#endif
#define BUFF_SIZE (WIDTH*HEIGHT)

PI_L2 unsigned char *buff;

PI_L2 unsigned char *buff_demosaick;

static struct pi_device camera;
static volatile int done;

static void handle_transfer_end(void
{
    done = 1;
}

static int open_camera(struct pi_dev
{
    printf("Opening Himax camera\n")
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA
#endif

    pi_open_from_conf(device, &cam_c
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAM

    return 0;
}
```

```c
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera.h"
#include "bsp/camera/himax.h"
```

Include drivers

```c
#include "gaplib/ImgIO.h"
```

Include image IO library

```c
#include "img_proc.h"
```

Include own demosaicking function

```c
#define WIDTH    324
#ifdef QVGA_MODE
#define HEIGHT   244
#else
#define HEIGHT   324
#endif
#define BUFF_SIZE (WIDTH*HEIGHT)
```

Define acquisition size

```c
PI_L2 unsigned char *buff;

PI_L2 unsigned char *buff_demosaick;

static struct pi_device camera;
static volatile int done;
```

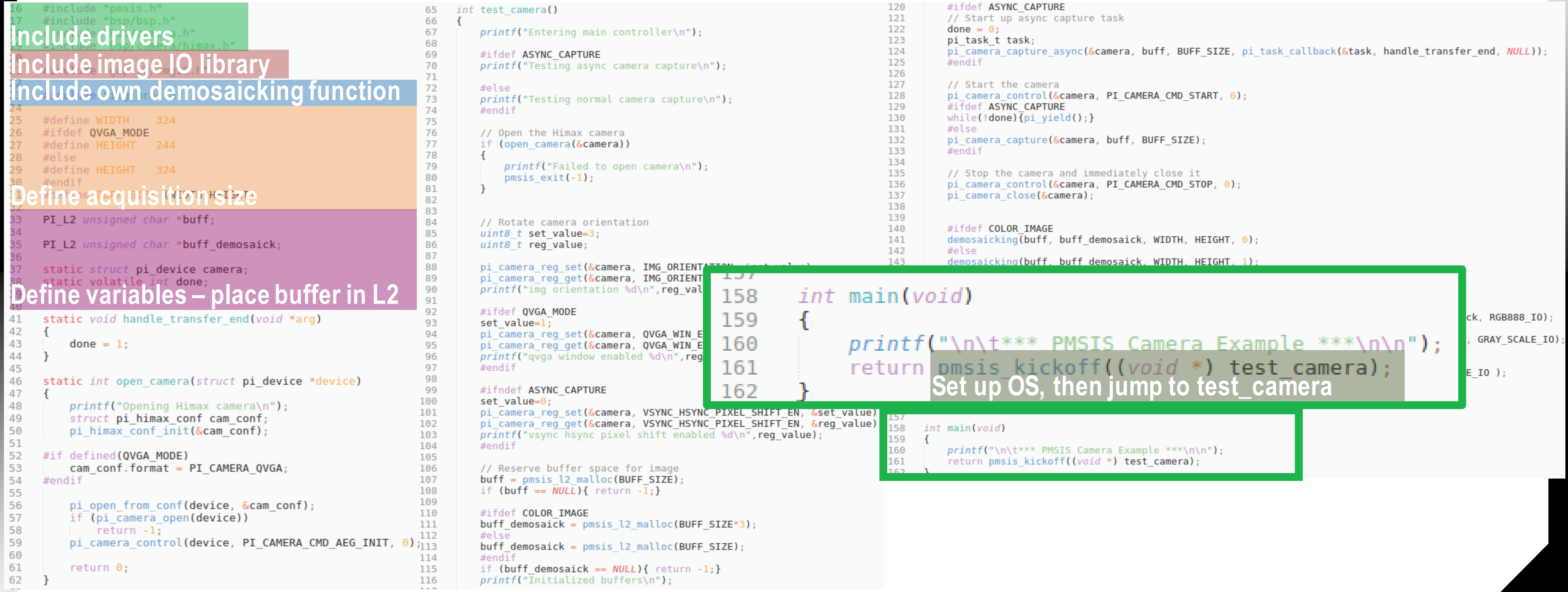
Define variables – place buffer in L2

```c
    #ifdef ASYNC_CAPTURE
    // Start up async capture task
    done = 0;
    pi_task_t task;
    pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, NULL));
    #endif
```

```makefile
APP = test
APP_SRCS += test.c $(GAP_LIB_PATH)/img_io/ImgIO.c img_proc.c
APP_INC  += . $(GAP_LIB_PATH)/include

APP_CFLAGS += -O3 -g


PMSIS_OS ?= pulp_os

APP_CFLAGS += -DASYNC_CAPTURE
APP_CFLAGS += -DQVGA_MODE
APP_CFLAGS += -DCOLOR_IMAGE


clean::
	rm -rf img_raw.ppm img_color.ppm img_gray.ppm

include $(RULES_DIR)/pmsis_rules.mk
```

```c
{
    printf("\n\t*** PMSIS Camera Example ***\n\n");
    return pmsis_kickoff((void *) test_camera);
}
```

Hands-on: Image acquisition and filtering



Include drivers

Include image IO library

Include own demosaicking function

Define acquisition size

Define variables – place buffer in L2

Set up OS, then jump to test_camera

```c
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera/himax.h"

#define WIDTH    324
#ifdef QVGA_MODE
#define HEIGHT   244
#else
#define HEIGHT   324
#endif

PI_L2 unsigned char *buff;

PI_L2 unsigned char *buff_demosaick;

static struct pi_device camera;
static volatile int done;

static void handle_transfer_end(void *arg)
{
    done = 1;
}

static int open_camera(struct pi_device *device)
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf);
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0);

    return 0;
}
```

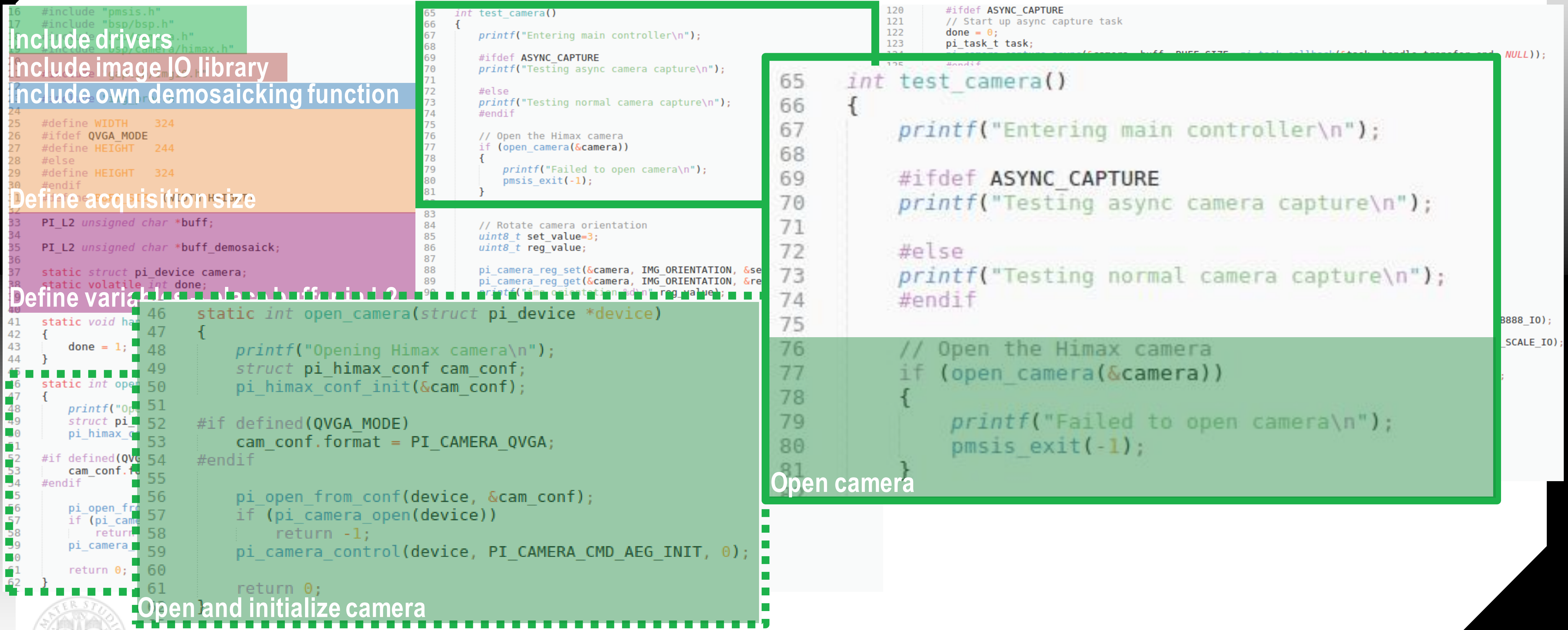
```c
int test_camera()
{
    printf("Entering main controller\n");

    #ifdef ASYNC_CAPTURE
    printf("Testing async camera capture\n");

    #else
    printf("Testing normal camera capture\n");
    #endif

    // Open the Himax camera
    if (open_camera(&camera))
    {
        printf("Failed to open camera\n");
        pmsis_exit(-1);
    }


    // Rotate camera orientation
    uint8_t set_value=3;
    uint8_t reg_value;

    pi_camera_reg_set(&camera, IMG_ORIENT
    pi_camera_reg_get(&camera, IMG_ORIENT
    printf("img orientation %d\n",reg_val

    #ifdef QVGA_MODE
    set_value=1;
    pi_camera_reg_set(&camera, QVGA_WIN_E
    pi_camera_reg_get(&camera, QVGA_WIN_E
    printf("qvga window enabled %d\n",reg
    #endif

    #ifndef ASYNC_CAPTURE
    set_value=0;
    pi_camera_reg_set(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &set_value);
    pi_camera_reg_get(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &reg_value);
    printf("vsync hsync pixel shift enabled %d\n",reg_value);
    #endif

    // Reserve buffer space for image
    buff = pmsis_l2_malloc(BUFF_SIZE);
    if (buff == NULL){ return -1;}

    #ifdef COLOR_IMAGE
    buff_demosaick = pmsis_l2_malloc(BUFF_SIZE*3);
    #else
    buff_demosaick = pmsis_l2_malloc(BUFF_SIZE);
    #endif
    if (buff_demosaick == NULL){ return -1;}
    printf("Initialized buffers\n");
```

```c
    #ifdef ASYNC_CAPTURE
    // Start up async capture task
    done = 0;
    pi_task_t task;
    pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, NULL));
    #endif

    // Start the camera
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
    #ifdef ASYNC_CAPTURE
    while(!done){pi_yield();}
    #else
    pi_camera_capture(&camera, buff, BUFF_SIZE);
    #endif

    // Stop the camera and immediately close it
    pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
    pi_camera_close(&camera);


    #ifdef COLOR_IMAGE
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
    #else
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
```

```c
int main(void)
{
    printf("\n\t*** PMSIS Camera Example ***\n\n");
    return pmsis_kickoff((void *) test_camera);
}
```

```c
int main(void)
{
    printf("\n\t*** PMSIS Camera Example ***\n\n");
    return pmsis_kickoff((void *) test_camera);
```

Hands-on: Image acquisition and filtering



Include drivers

```c
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera/himax.h"
```

Include image IO library

Include own demosaicking function

```c
#define WIDTH    324
#ifdef QVGA_MODE
#define HEIGHT   244
#else
#define HEIGHT   324
#endif
```

Define acquisition size

```c
PI_L2 unsigned char *buff;

PI_L2 unsigned char *buff_demosaick;

static struct pi_device camera;
static volatile int done;
```

Define variables – place buffer in L2

```c
static void ha
{
    done = 1;
}
```

```c
static int open_camera(struct pi_device *device)
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf);
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0);

    return 0;
}
```

Open and initialize camera

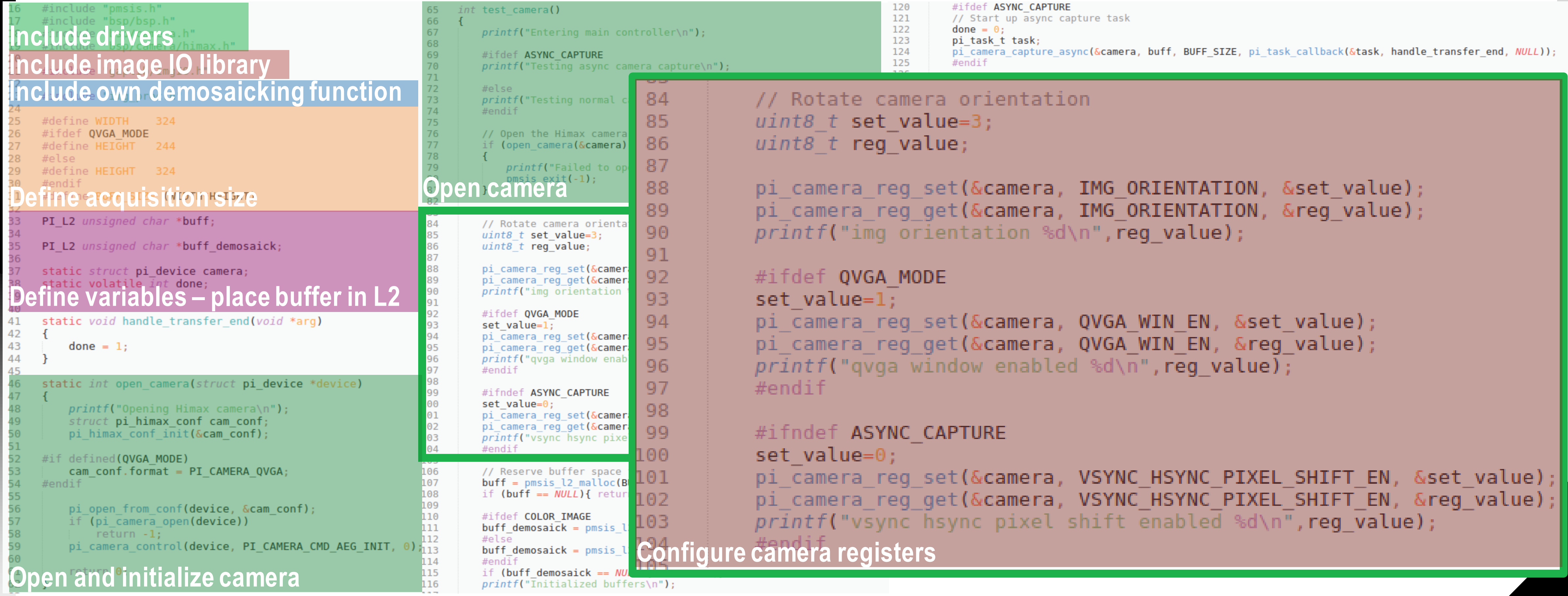
```c
int test_camera()
{
    printf("Entering main controller\n");

    #ifdef ASYNC_CAPTURE
    printf("Testing async camera capture\n");

    #else
    printf("Testing normal camera capture\n");
    #endif

    // Open the Himax camera
    if (open_camera(&camera))
    {
        printf("Failed to open camera\n");
        pmsis_exit(-1);
    }
```

Open camera

```c

    // Rotate camera orientation
    uint8_t set_value=3;
    uint8_t reg_value;

    pi_camera_reg_set(&camera, IMG_ORIENTATION, &se
    pi_camera_reg_get(&camera, IMG_ORIENTATION, &re
```

```c
    #ifdef ASYNC_CAPTURE
    // Start up async capture task
    done = 0;
    pi_task_t task;
                                                              NULL));
```

Hands-on: Image acquisition and filtering



Include drivers

```c
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera.h"
#include "bsp/camera/himax.h"
```

Include image IO library

```c
#include "gaplib/ImgIO.h"
```

Include own demosaicking function

```c
#include "img_proc.h"
```

Define acquisition size

```c
#define WIDTH    324
#ifdef QVGA_MODE
#define HEIGHT   244
#else
#define HEIGHT   324
#endif
#define BUFF_SIZE (WIDTH*HEIGHT)
```

Define variables – place buffer in L2

```c
PI_L2 unsigned char *buff;

PI_L2 unsigned char *buff_demosaick;

static struct pi_device camera;
static volatile int done;
```

```c
static void handle_transfer_end(void *arg)
{
    done = 1;
}
```

Open and initialize camera

```c
static int open_camera(struct pi_device *device)
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf);
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0);

    return 0;
}
```

Open camera

```c
int test_camera()
{
    printf("Entering main controller\n");

    #ifdef ASYNC_CAPTURE
    printf("Testing async camera capture\n");

    #else
    printf("Testing normal c
    #endif

    // Open the Himax camera
    if (open_camera(&camera)
    {
        printf("Failed to op
        pmsis_exit(-1);
    }
```

```c
    // Rotate camera orienta
    uint8_t set_value=3;
    uint8_t reg_value;

    pi_camera_reg_set(&camer
    pi_camera_reg_get(&camer
    printf("img orientation 

    #ifdef QVGA_MODE
    set_value=1;
    pi_camera_reg_set(&camer
    pi_camera_reg_get(&camer
    printf("qvga window enab
    #endif

    #ifndef ASYNC_CAPTURE
    set_value=0;
    pi_camera_reg_set(&camer
    pi_camera_reg_get(&camer
    printf("vsync hsync pixe
    #endif
```

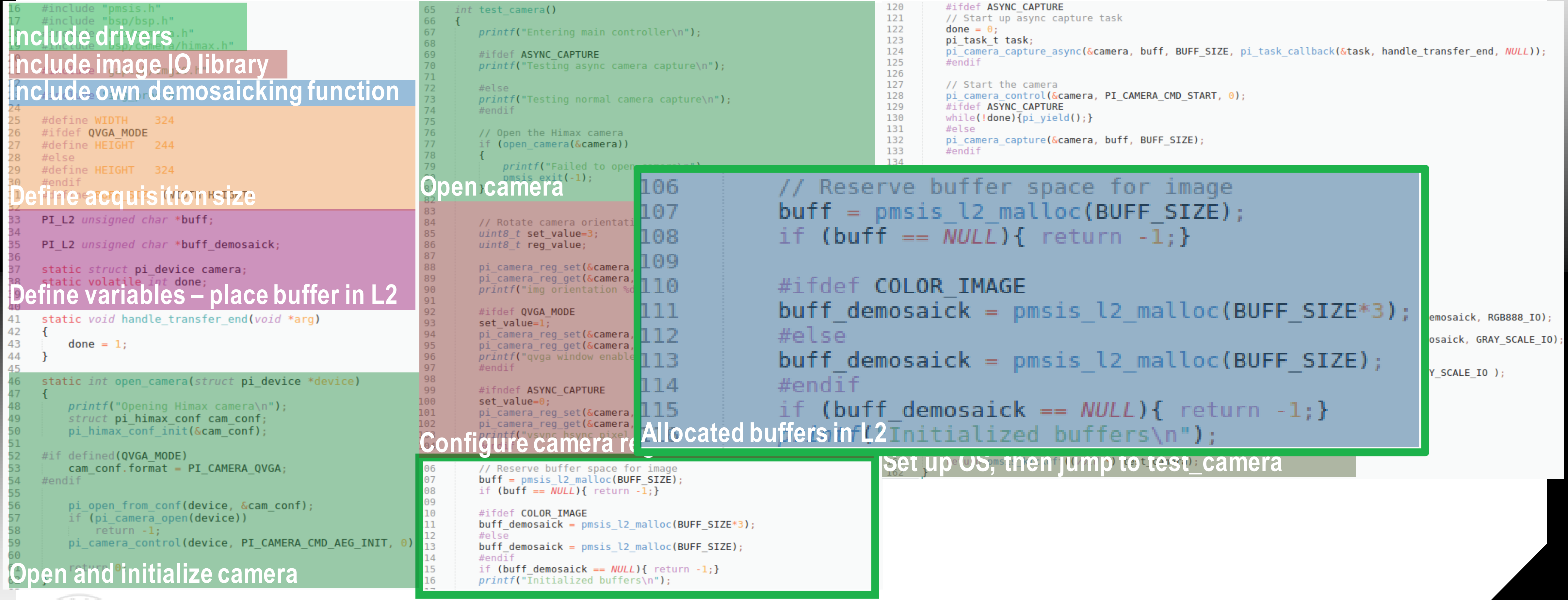
```c
    // Reserve buffer space
    buff = pmsis_l2_malloc(BU
    if (buff == NULL){ retur

    #ifdef COLOR_IMAGE
    buff_demosaick = pmsis_l
    #else
    buff_demosaick = pmsis_l
    #endif
    if (buff_demosaick == NU
    printf("Initialized buffers\n");
```

```c
    #ifdef ASYNC_CAPTURE
    // Start up async capture task
    done = 0;
    pi_task_t task;
    pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, NULL));
    #endif
```

```c
    // Rotate camera orientation
    uint8_t set_value=3;
    uint8_t reg_value;

    pi_camera_reg_set(&camera, IMG_ORIENTATION, &set_value);
    pi_camera_reg_get(&camera, IMG_ORIENTATION, &reg_value);
    printf("img orientation %d\n",reg_value);

    #ifdef QVGA_MODE
    set_value=1;
    pi_camera_reg_set(&camera, QVGA_WIN_EN, &set_value);
    pi_camera_reg_get(&camera, QVGA_WIN_EN, &reg_value);
    printf("qvga window enabled %d\n",reg_value);
    #endif

    #ifndef ASYNC_CAPTURE
    set_value=0;
    pi_camera_reg_set(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &set_value);
    pi_camera_reg_get(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &reg_value);
    printf("vsync hsync pixel shift enabled %d\n",reg_value);
    #endif
```

Configure camera registers

Hands-on: Image acquisition and filtering



Include drivers

```
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera/himax.h"
```

Include image IO library

Include own demosaicking function

```
#define WIDTH    324
#ifdef QVGA_MODE
#define HEIGHT   244
#else
#define HEIGHT   324
#endif
```

Define acquisition size

```
PI_L2 unsigned char *buff;

PI_L2 unsigned char *buff_demosaick;

static struct pi_device camera;
static volatile int done;
```

Define variables – place buffer in L2

```
static void handle_transfer_end(void *arg)
{
    done = 1;
}

static int open_camera(struct pi_device *device)
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf);
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0)
```

Open and initialize camera

```
int test_camera()
{
    printf("Entering main controller\n");

    #ifdef ASYNC_CAPTURE
    printf("Testing async camera capture\n");

    #else
    printf("Testing normal camera capture\n");
    #endif

    // Open the Himax camera
    if (open_camera(&camera))
    {
        printf("Failed to open
        pmsis_exit(-1);
```

Open camera

```
    // Rotate camera orientati
    uint8_t set_value=3;
    uint8_t reg_value;

    pi_camera_reg_set(&camera,
    pi_camera_reg_get(&camera,
    printf("img orientation %d

    #ifdef QVGA_MODE
    set_value=1;
    pi_camera_reg_set(&camera,
    pi_camera_reg_get(&camera,
    printf("qvga window enable
    #endif

    #ifndef ASYNC_CAPTURE
    set_value=0;
    pi_camera_reg_set(&camera,
    pi_camera_reg_get(&camera,
    printf("vsync hsync pixel
```

Configure camera re

```
    // Reserve buffer space for image
    buff = pmsis_l2_malloc(BUFF_SIZE);
    if (buff == NULL){ return -1;}

    #ifdef COLOR_IMAGE
    buff_demosaick = pmsis_l2_malloc(BUFF_SIZE*3);
    #else
    buff_demosaick = pmsis_l2_malloc(BUFF_SIZE);
    #endif
    if (buff_demosaick == NULL){ return -1;}
    printf("Initialized buffers\n");
```

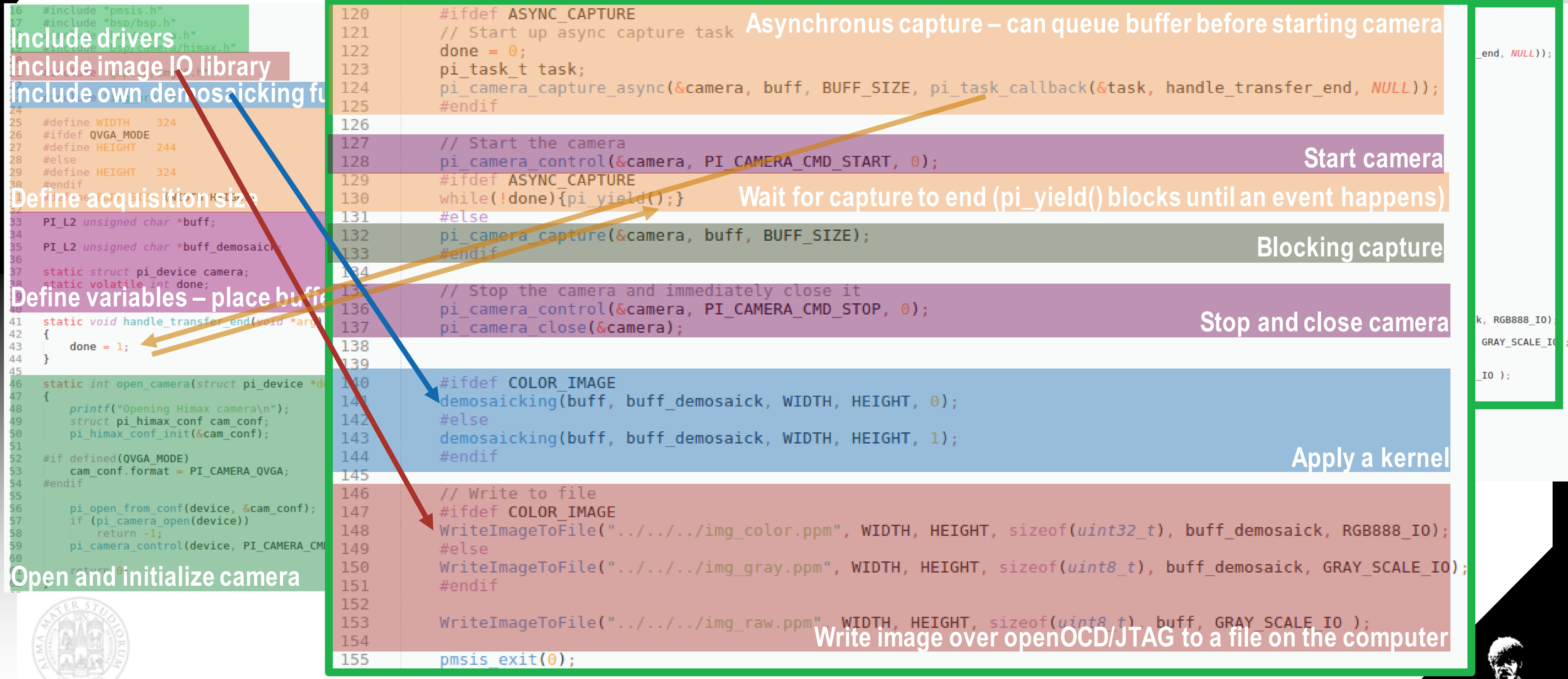
Allocated buffers in L2

```
    #ifdef ASYNC_CAPTURE
    // Start up async capture task
    done = 0;
    pi_task_t task;
    pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, NULL));
    #endif

    // Start the camera
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
    #ifdef ASYNC_CAPTURE
    while(!done){pi_yield();}
    #else
    pi_camera_capture(&camera, buff, BUFF_SIZE);
    #endif
```

```
emosaick, RGB888_IO);
osaick, GRAY_SCALE_IO);
Y_SCALE_IO );
```

Set up OS, then jump to test_camera

Hands-on: Image acquisition and filtering



```c
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera.h"
#include "bsp/camera/himax.h"

#include "gaplib/ImgIO.h"

#include "img_proc.h"

#define WIDTH    324
#ifdef QVGA_MODE
#define HEIGHT   244
#else
#define HEIGHT   324
#endif
#define BUFF_SIZE (WIDTH*HEIGHT)

PI_L2 unsigned char *buff;

PI_L2 unsigned char *buff_demosaick;

static struct pi_device camera;
static volatile int done;


static void handle_transfer_end(void *arg)
{
    done = 1;
}

static int open_camera(struct pi_device *de
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf);
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA_CMI

    return 0;
}
```

Include drivers

Include image IO library

Include own demosaicking fu

Define acquisition size

Define variables – place buffe

Open and initialize camera

```c
    #ifdef ASYNC_CAPTURE
    // Start up async capture task
    done = 0;
    pi_task_t task;
    pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, NULL));
    #endif

    // Start the camera
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
    #ifdef ASYNC_CAPTURE
    while(!done){pi_yield();}
    #else
    pi_camera_capture(&camera, buff, BUFF_SIZE);
    #endif

    // Stop the camera and immediately close it
    pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
    pi_camera_close(&camera);


    #ifdef COLOR_IMAGE
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
    #else
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
    #endif

    // Write to file
    #ifdef COLOR_IMAGE
    WriteImageToFile("../../../img_color.ppm", WIDTH, HEIGHT, sizeof(uint32_t), buff_demosaick, RGB888_IO);
    #else
    WriteImageToFile("../../../img_gray.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff_demosaick, GRAY_SCALE_IO);
    #endif

    WriteImageToFile("../../../img_raw.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff, GRAY_SCALE_IO );

    pmsis_exit(0);
```

Asynchronus capture – can queue buffer before starting camera

Start camera

Wait for capture to end (pi_yield() blocks until an event happens)

Blocking capture

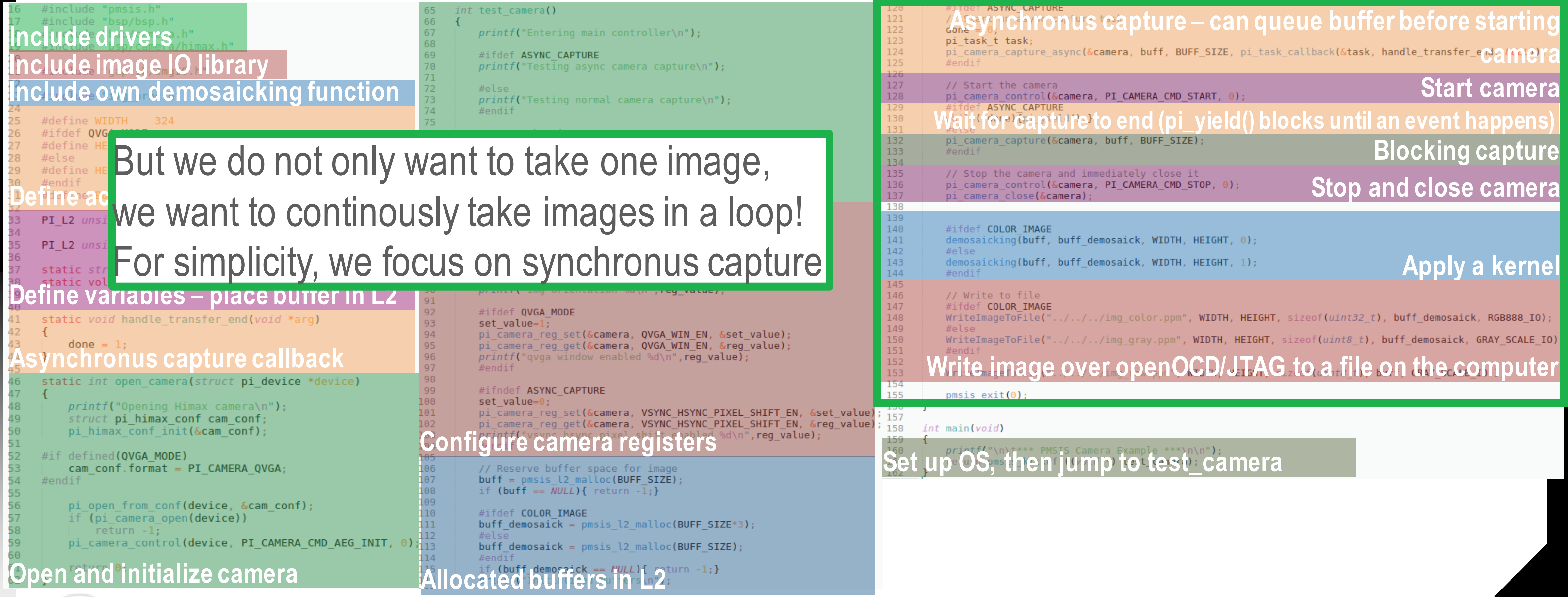
Stop and close camera

Apply a kernel

Write image over openOCD/JTAG to a file on the computer

```c
_end, NULL));
k, RGB888_IO);
 GRAY_SCALE_IO
_IO );
```

Hands-on: Image acquisition and filtering



```c
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera/himax.h"

#define WIDTH    324
#ifdef QVG
#define HE
#else
#define HE
#endif

PI_L2 unsi

PI_L2 unsi

static str
static vol
```

Include drivers

Include image IO library

Include own demosaicking function

Define ac

Define variables – place buffer in L2

```c
static void handle_transfer_end(void *arg)
{
    done = 1;
}
```

Asynchronus capture callback

```c
static int open_camera(struct pi_device *device)
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf);
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0);

    return 0;
}
```

Open and initialize camera

```c
int test_camera()
{
    printf("Entering main controller\n");

    #ifdef ASYNC_CAPTURE
    printf("Testing async camera capture\n");

    #else
    printf("Testing normal camera capture\n");
    #endif

    #ifdef QVGA_MODE
    set_value=1;
    pi_camera_reg_set(&camera, QVGA_WIN_EN, &set_value);
    pi_camera_reg_get(&camera, QVGA_WIN_EN, &reg_value);
    printf("qvga window enabled %d\n",reg_value);
    #endif

    #ifndef ASYNC_CAPTURE
    set_value=0;
    pi_camera_reg_set(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &set_value);
    pi_camera_reg_get(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &reg_value);
    printf("vsync hsync pixel shift enabled %d\n",reg_value);
```

Configure camera registers

```c
    // Reserve buffer space for image
    buff = pmsis_l2_malloc(BUFF_SIZE);
    if (buff == NULL){ return -1;}

    #ifdef COLOR_IMAGE
    buff_demosaick = pmsis_l2_malloc(BUFF_SIZE*3);
    #else
    buff_demosaick = pmsis_l2_malloc(BUFF_SIZE);
    #endif
    if (buff_demosaick == NULL){ return -1;}
```

Allocated buffers in L2

```c
    #ifdef ASYNC_CAPTURE
    done = 0;
    pi_task_t task;
    pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_e
    #endif
```

Asynchronus capture – can queue buffer before starting camera

```c
    // Start the camera
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
    #ifdef ASYNC_CAPTURE
```

Start camera

Wait for capture to end (pi_yield() blocks until an event happens)

```c
    pi_camera_capture(&camera, buff, BUFF_SIZE);
    #endif
```

Blocking capture

```c
    // Stop the camera and immediately close it
    pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
    pi_camera_close(&camera);


    #ifdef COLOR_IMAGE
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
    #else
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
    #endif
```

Stop and close camera

Apply a kernel

```c
    // Write to file
    #ifdef COLOR_IMAGE
    WriteImageToFile("../../../img_color.ppm", WIDTH, HEIGHT, sizeof(uint32_t), buff_demosaick, RGB888_IO);
    #else
    WriteImageToFile("../../../img_gray.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff_demosaick, GRAY_SCALE_IO)
    #endif
```

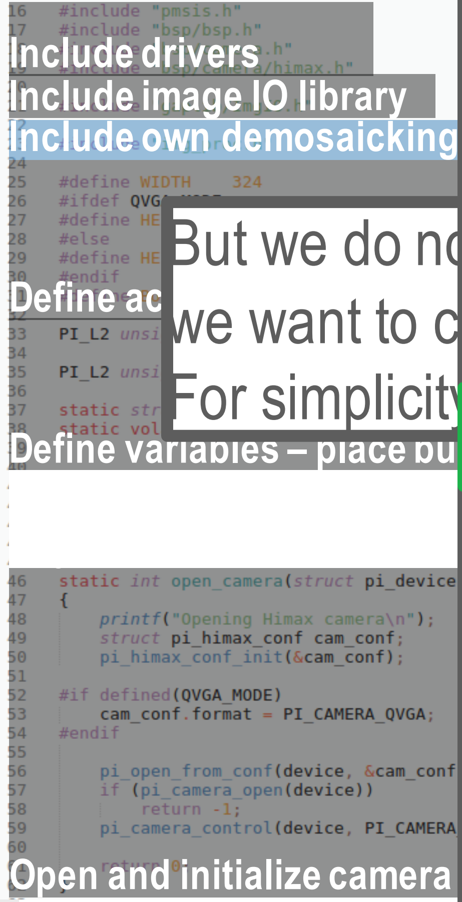
Write image over openOCD/JTAG to a file on the computer

```c
    pmsis_exit(0);
}

int main(void)
{
    printf("\n\t*** PMSIS Camera Example ***\n\n");
```

Set up OS, then jump to test_camera

But we do not only want to take one image, we want to continously take images in a loop! For simplicity, we focus on synchronus capture

Hands-on: Image acquisition and filtering



```
#include "pmsis.h"
#include "bsp/bsp.h"
#include "bsp/camera.h"
#include "bsp/camera/himax.h"

#include "gaplib/ImgIO.h"

#include "img_proc..."

#define WIDTH    324
#ifdef QVGA_MODE
#define HE
#else
#define HE
#endif

PI_L2 unsi

PI_L2 unsi

static str
static vol
```

Include drivers

Include image IO library

Include own demosaicking

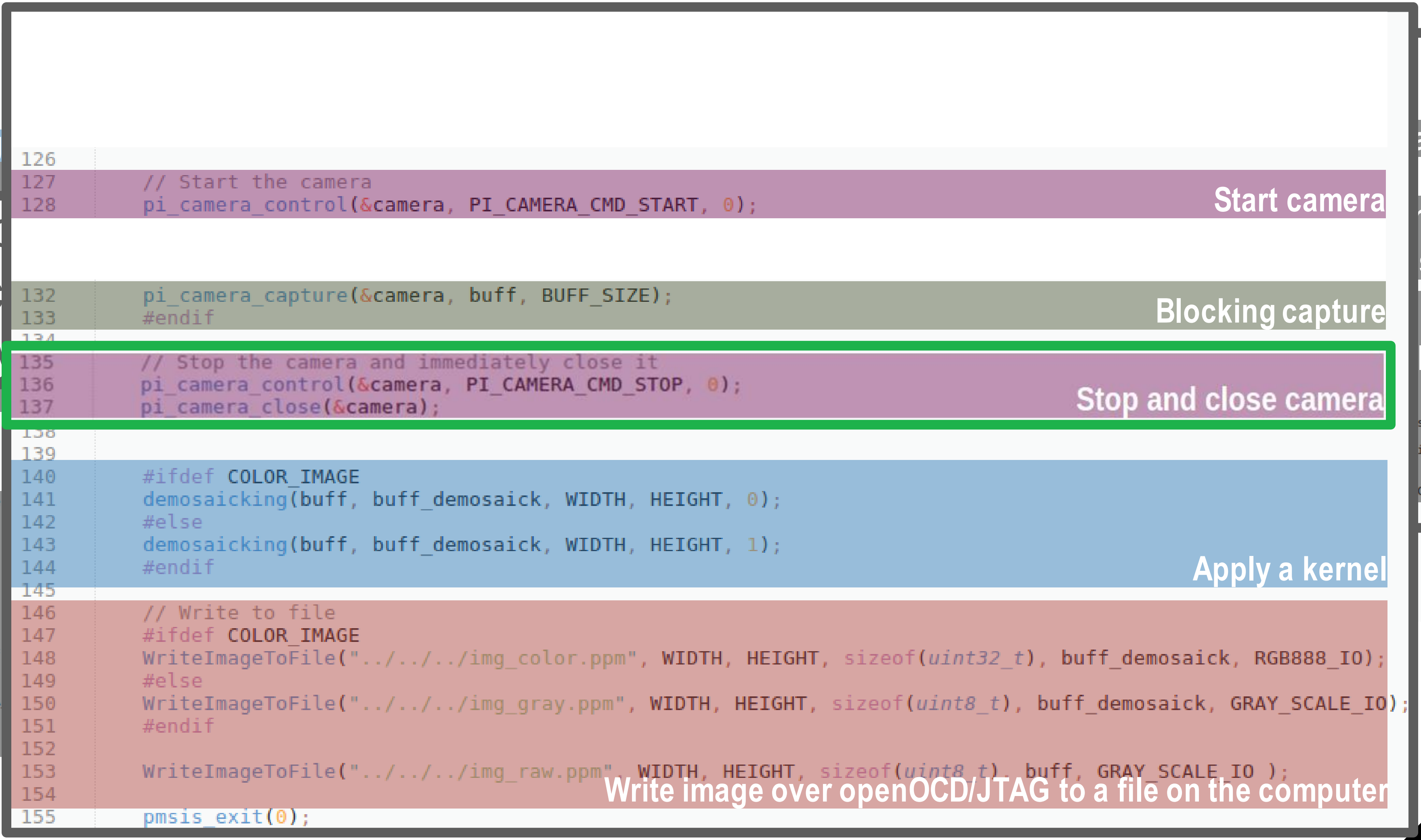
Define ac

But we do no
we want to c
For simplicity

Define variables – place bu

```
static int open_camera(struct pi_device
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA

    return 0;
}
```

Open and initialize camera



```
    // Start the camera
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);

    pi_camera_capture(&camera, buff, BUFF_SIZE);
    #endif

    // Stop the camera and immediately close it
    pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
    pi_camera_close(&camera);


    #ifdef COLOR_IMAGE
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
    #else
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
    #endif

    // Write to file
    #ifdef COLOR_IMAGE
    WriteImageToFile("../../../img_color.ppm", WIDTH, HEIGHT, sizeof(uint32_t), buff_demosaick, RGB888_IO);
    #else
    WriteImageToFile("../../../img_gray.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff_demosaick, GRAY_SCALE_IO);
    #endif

    WriteImageToFile("../../../img_raw.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff, GRAY_SCALE_IO );

    pmsis_exit(0);
```

Start camera

Blocking capture

Stop and close camera

Apply a kernel

Write image over openOCD/JTAG to a file on the computer

art camera

ng capture

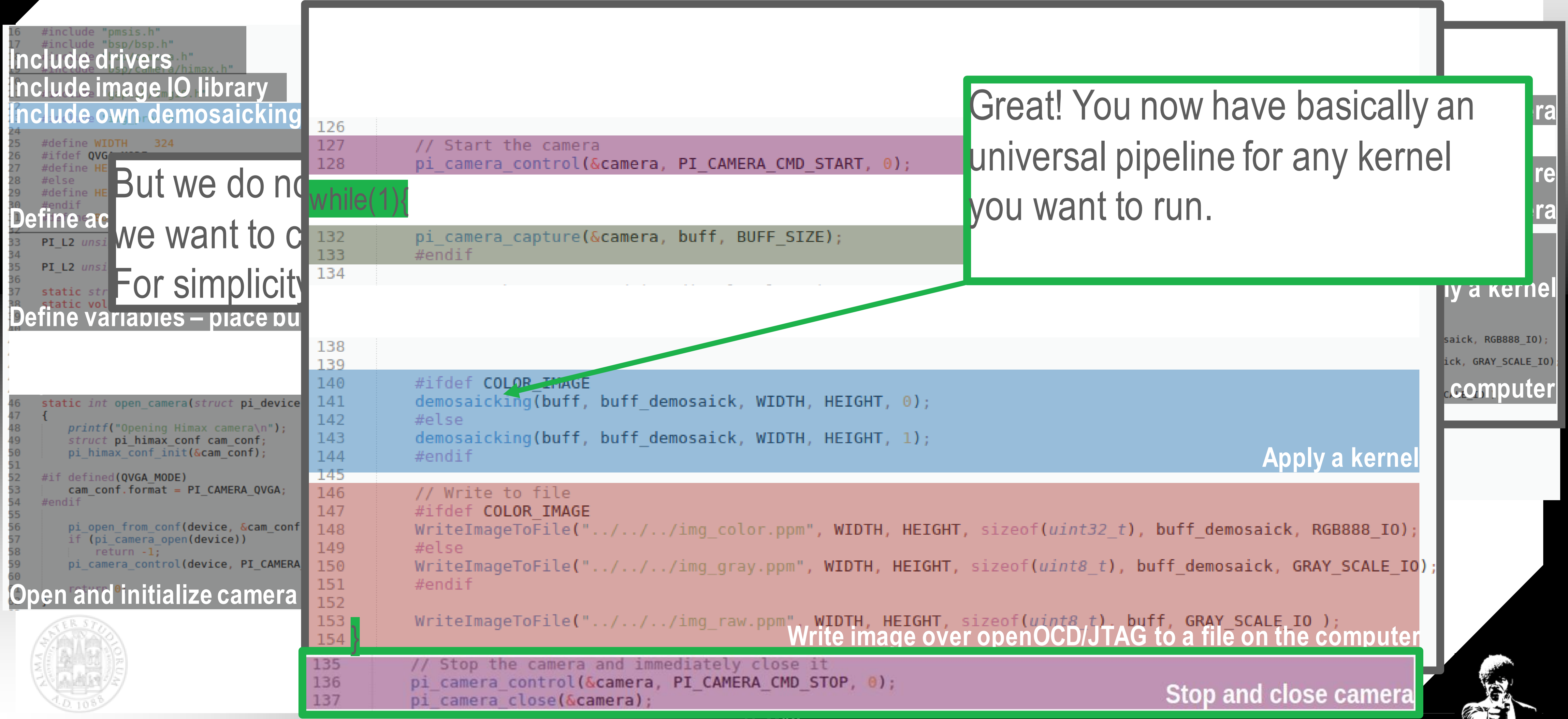
se camera

ly a kernel

```
saick, RGB888_IO);
ick, GRAY_SCALE_IO)
```

computer

Hands-on: Image acquisition and filtering



```
#include "pmsis.h"
#include "bsp/bsp.h"
```

Include drivers

Include image IO library

Include own demosaicking

```
#define WIDTH    324
#ifdef QVG
#define HE
#else
#define HE
#endif
```

Define ac

```
PI_L2 unsi

PI_L2 unsi

static str
static vol
```

Define variables – place bu

But we do no
we want to c
For simplicity

```
static int open_camera(struct pi_device
{
    printf("Opening Himax camera\n");
    struct pi_himax_conf cam_conf;
    pi_himax_conf_init(&cam_conf);

#if defined(QVGA_MODE)
    cam_conf.format = PI_CAMERA_QVGA;
#endif

    pi_open_from_conf(device, &cam_conf
    if (pi_camera_open(device))
        return -1;
    pi_camera_control(device, PI_CAMERA
```

Open and initialize camera

```
    // Start the camera
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
while(1){
    pi_camera_capture(&camera, buff, BUFF_SIZE);
    #endif

    #ifdef COLOR_IMAGE
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
    #else
    demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
    #endif
```

Apply a kernel

```
    // Write to file
    #ifdef COLOR_IMAGE
    WriteImageToFile("../../../img_color.ppm", WIDTH, HEIGHT, sizeof(uint32_t), buff_demosaick, RGB888_IO);
    #else
    WriteImageToFile("../../../img_gray.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff_demosaick, GRAY_SCALE_IO);
    #endif

    WriteImageToFile("../../../img_raw.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff, GRAY_SCALE_IO );
}
```

Write image over openOCD/JTAG to a file on the computer

```
    // Stop the camera and immediately close it
    pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
    pi_camera_close(&camera);
```

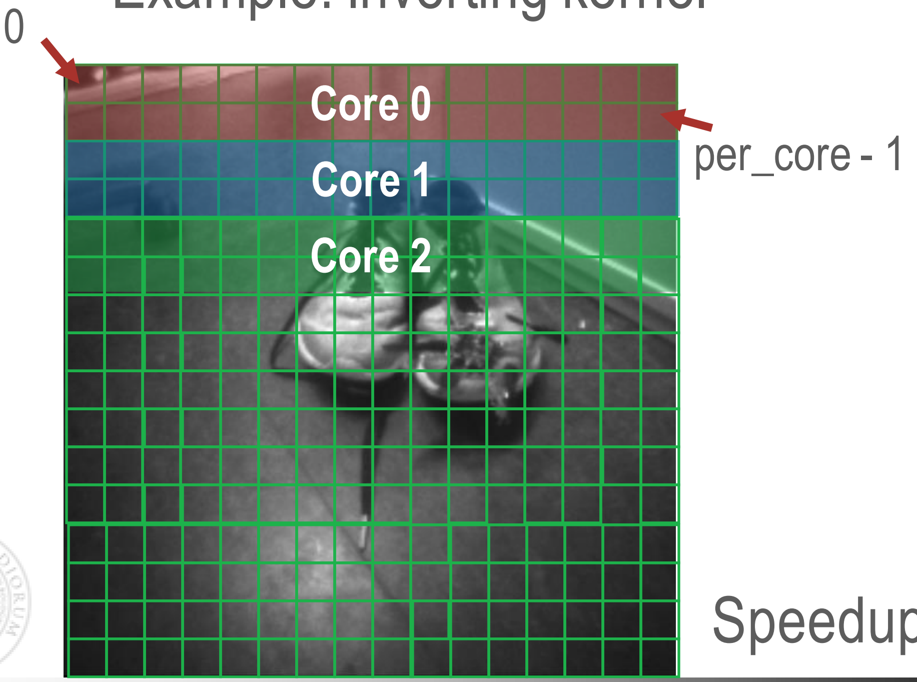
Stop and close camera

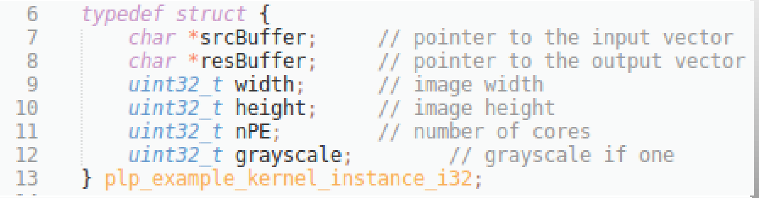
Great! You now have basically an universal pipeline for any kernel you want to run.

Hands-on: Image acquisition and filtering

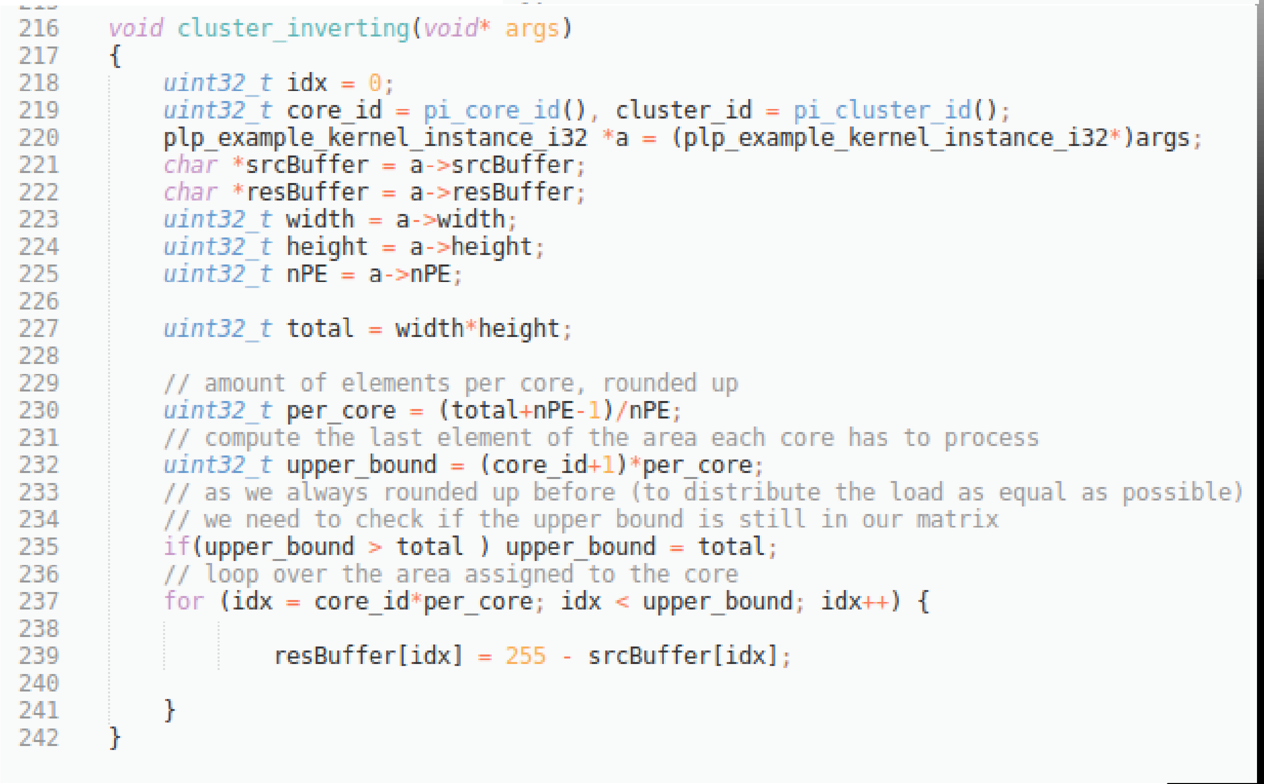
How do we improve performance?

- Avoid float operations
- Parallelize code
  - All cores should execute similar code on different data
- Example: Inverting kernel





```c
typedef struct {
    char *srcBuffer;      // pointer to the input vector
    char *resBuffer;      // pointer to the output vector
    uint32_t width;       // image width
    uint32_t height;      // image height
    uint32_t nPE;         // number of cores
    uint32_t grayscale;        // grayscale if one
} plp_example_kernel_instance_i32;
```



```c
void cluster_inverting(void* args)
{
    uint32_t idx = 0;
    uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
    plp_example_kernel_instance_i32 *a = (plp_example_kernel_instance_i32*)args;
    char *srcBuffer = a->srcBuffer;
    char *resBuffer = a->resBuffer;
    uint32_t width = a->width;
    uint32_t height = a->height;
    uint32_t nPE = a->nPE;

    uint32_t total = width*height;

    // amount of elements per core, rounded up
    uint32_t per_core = (total+nPE-1)/nPE;
    // compute the last element of the area each core has to process
    uint32_t upper_bound = (core_id+1)*per_core;
    // as we always rounded up before (to distribute the load as equal as possible)
    // we need to check if the upper bound is still in our matrix
    if(upper_bound > total ) upper_bound = total;
    // loop over the area assigned to the core
    for (idx = core_id*per_core; idx < upper_bound; idx++) {

        resBuffer[idx] = 255 - srcBuffer[idx];

    }
}
```

Speedup: @50MHz FC and Cluster from 8ms ->1.5ms

PULP PLATFORM
Open Source Hardware, the way it should be!

Bitcraze Workshop: AI-deck
The Application Layer

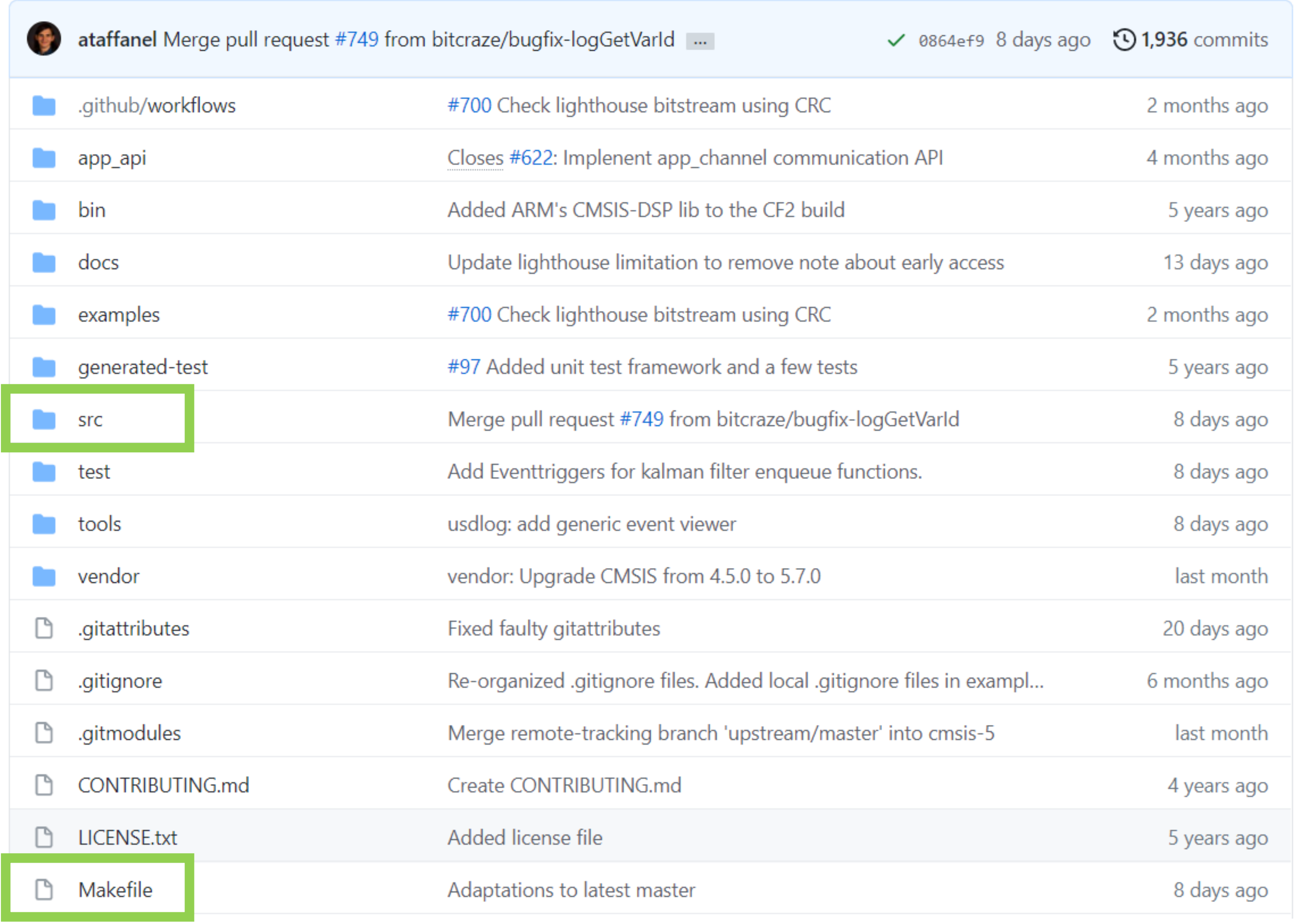
Lorenzo Lamberti, Hanna Müller, *Vlad Niculescu*, Manuele Rusci, Daniele Palossi

Firmware Overview

- Open-source, available at: https://github.com/bitcraze/crazyflie-firmware.
- Based on FreeRTOS.
- The firmware implements solutions for: state estimation, control, logging, trajectory planning, etc.
- It implements the sensor drivers and deck drivers.
  Deck: a plug-in PCB that is attached to the Crazyflie.
- The user can add new functionalities.

Firmware Overview

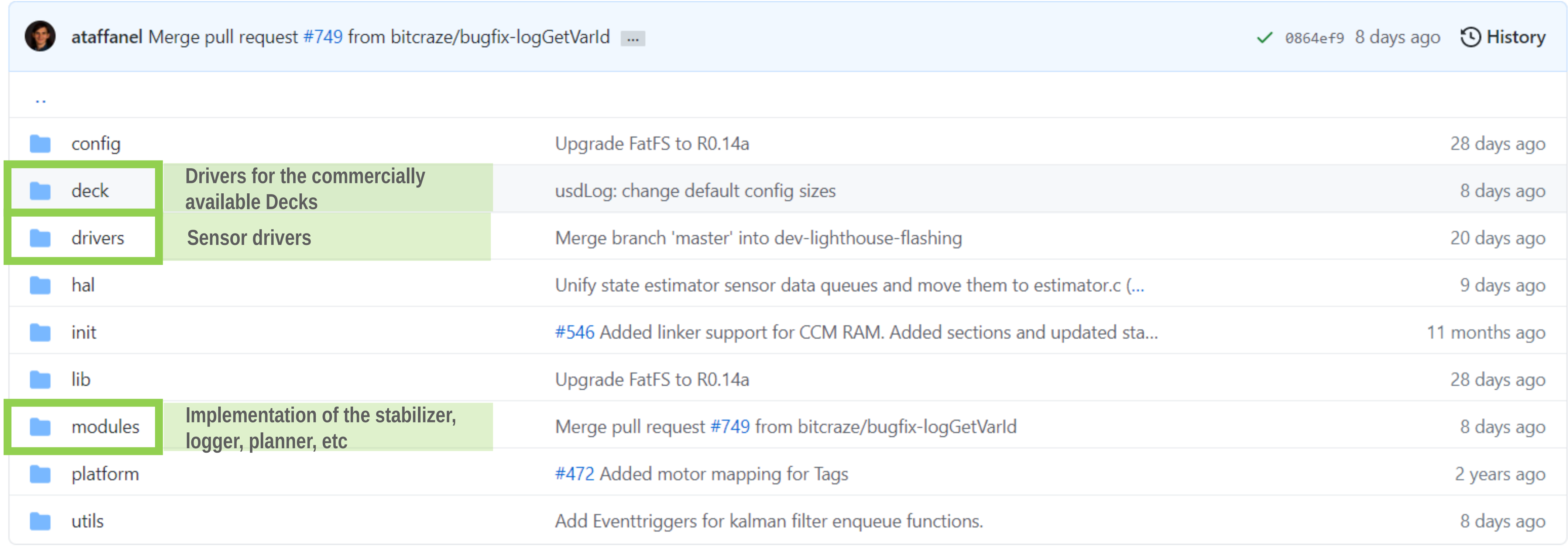
Firmware source files



ataffanel Merge pull request #749 from bitcraze/bugfix-logGetVarId … ✓ 0864ef9 8 days ago 1,936 commits

.github/workflows	#700 Check lighthouse bitstream using CRC	2 months ago
app_api	Closes #622: Implenent app_channel communication API	4 months ago
bin	Added ARM's CMSIS-DSP lib to the CF2 build	5 years ago
docs	Update lighthouse limitation to remove note about early access	13 days ago
examples	#700 Check lighthouse bitstream using CRC	2 months ago
generated-test	#97 Added unit test framework and a few tests	5 years ago
src	Merge pull request #749 from bitcraze/bugfix-logGetVarId	8 days ago
test	Add Eventtriggers for kalman filter enqueue functions.	8 days ago
tools	usdlog: add generic event viewer	8 days ago
vendor	vendor: Upgrade CMSIS from 4.5.0 to 5.7.0	last month
.gitattributes	Fixed faulty gitattributes	20 days ago
.gitignore	Re-organized .gitignore files. Added local .gitignore files in exampl…	6 months ago
.gitmodules	Merge remote-tracking branch 'upstream/master' into cmsis-5	last month
CONTRIBUTING.md	Create CONTRIBUTING.md	4 years ago
LICENSE.txt	Added license file	5 years ago
Makefile	Adaptations to latest master	8 days ago

Firmware Overview – Source Files



ataffanel Merge pull request #749 from bitcraze/bugfix-logGetVarId ... ✓ 0864ef9 8 days ago History

..

config	Upgrade FatFS to R0.14a	28 days ago
deck	usdLog: change default config sizes	8 days ago
drivers	Merge branch 'master' into dev-lighthouse-flashing	20 days ago
hal	Unify state estimator sensor data queues and move them to estimator.c (...	9 days ago
init	#546 Added linker support for CCM RAM. Added sections and updated sta...	11 months ago
lib	Upgrade FatFS to R0.14a	28 days ago
modules	Merge pull request #749 from bitcraze/bugfix-logGetVarId	8 days ago
platform	#472 Added motor mapping for Tags	2 years ago
utils	Add Eventtriggers for kalman filter enqueue functions.	8 days ago

deck: Drivers for the commercially available Decks

drivers: Sensor drivers

modules: Implementation of the stabilizer, logger, planner, etc

Developping Your Own Application

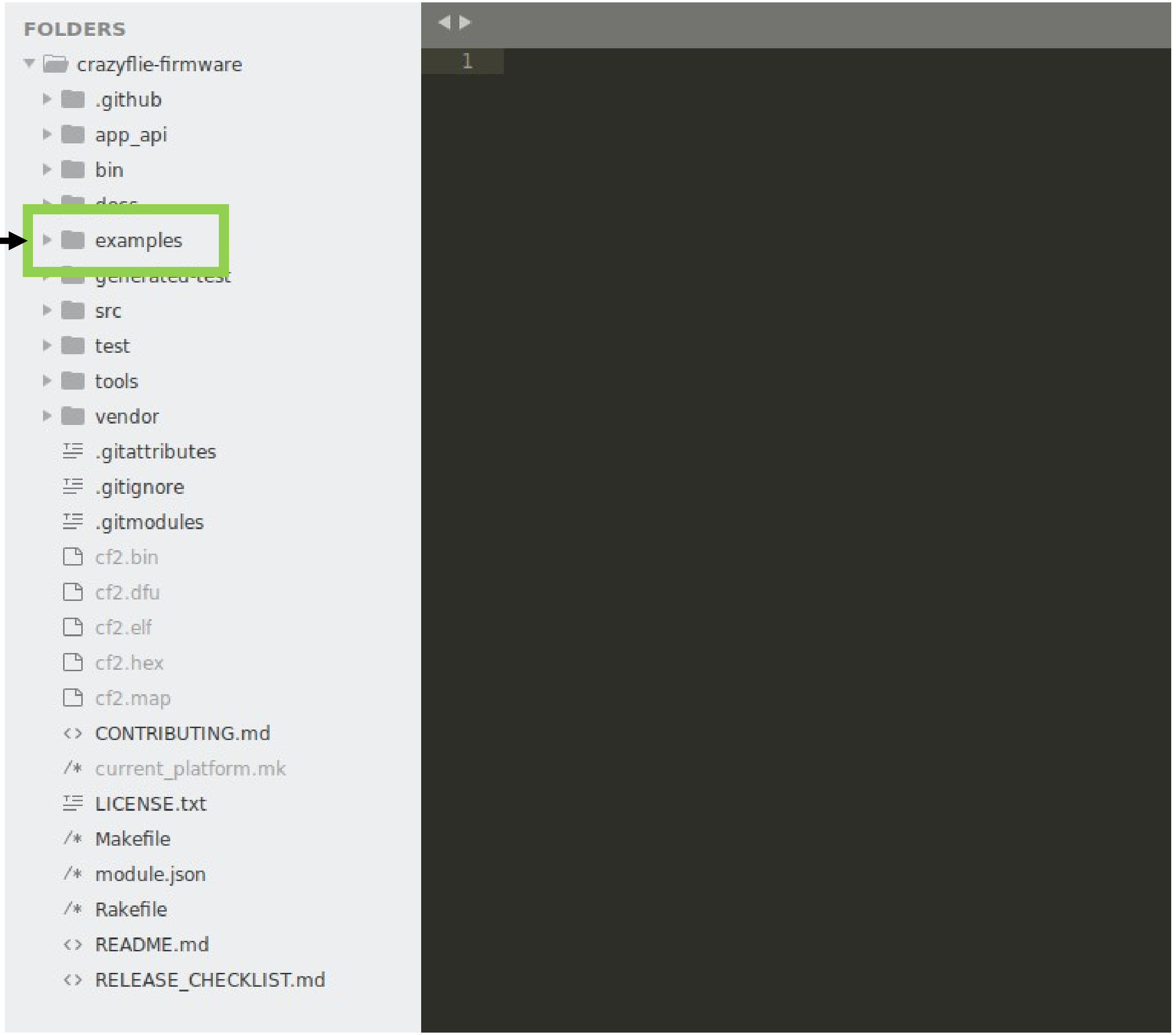
- One option for developing with Crazyflie, is to add the new source files to the *modules* or as a new *deck*.
- Not the best practice, since it alters the firmware and could cause conflicts with future updates (i.e., git pull conflicts).

Developping Your Own Application

- The *Application Layer* feature of the firmware allows the user to develop an application without changing the firmware.
- The code written within an application, is integrated as a new task and executed by the scheduler of the main firmware.

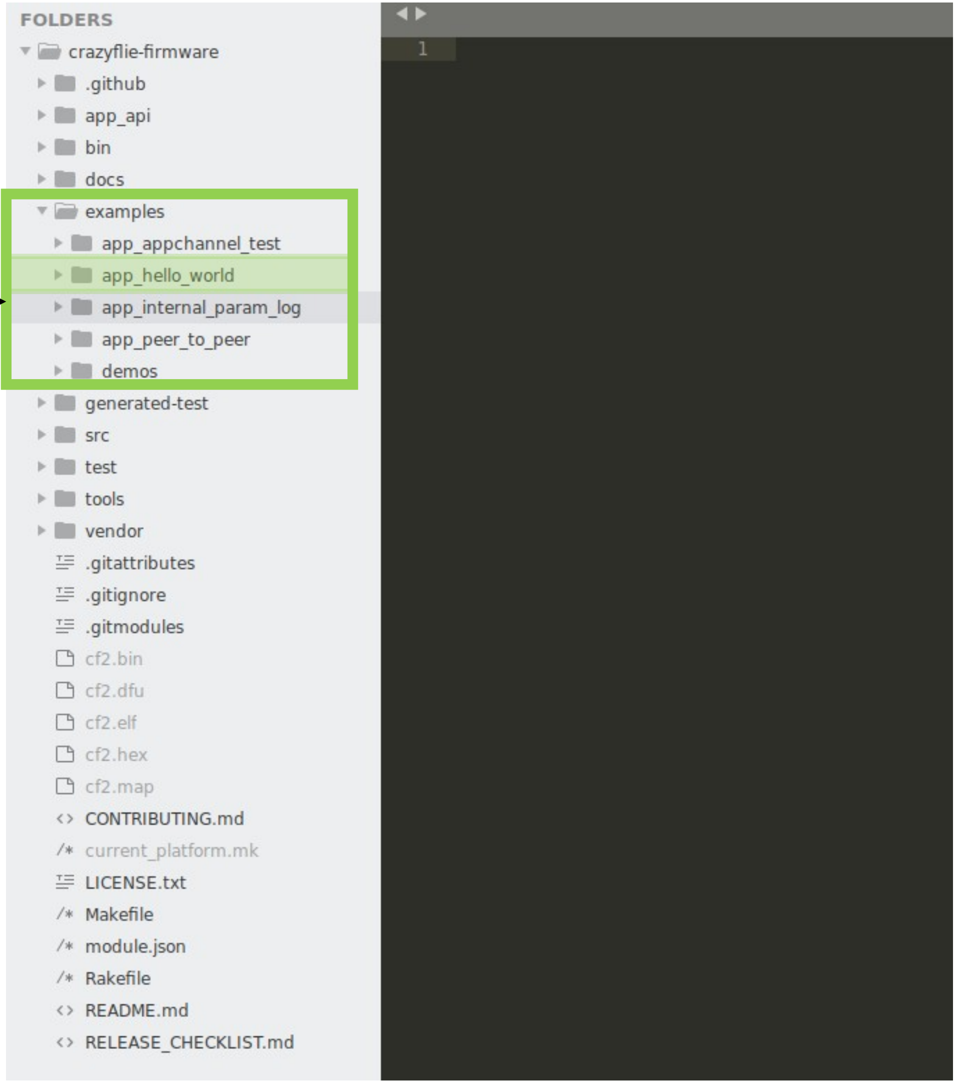
Firmware Overview

Examples on developing using the Application Layer



Example Applications

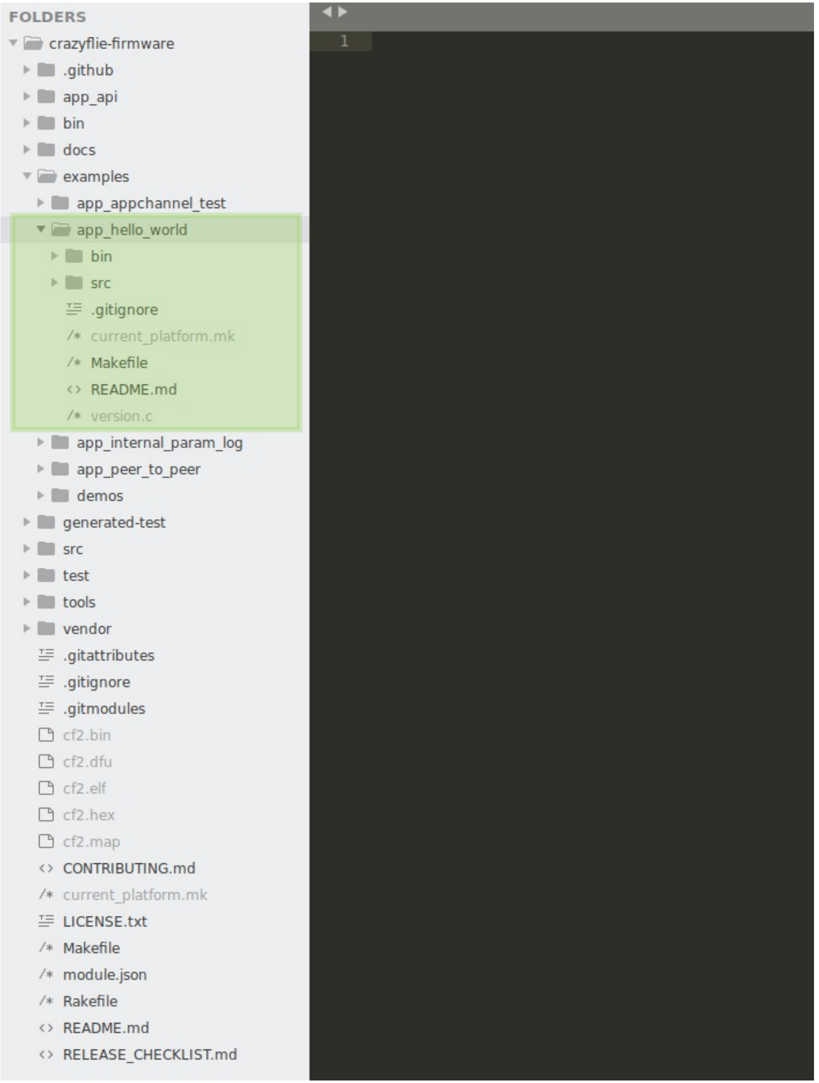
Examples on developing using the Application Layer



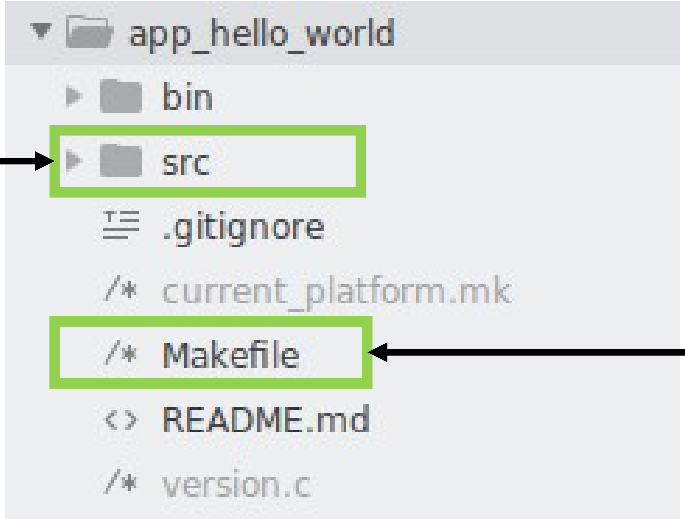
FOLDERS

- crazyflie-firmware
  - .github
  - app_api
  - bin
  - docs
  - examples
    - app_appchannel_test
    - app_hello_world
    - app_internal_param_log
    - app_peer_to_peer
    - demos
  - generated-test
  - src
  - test
  - tools
  - vendor
  - .gitattributes
  - .gitignore
  - .gitmodules
  - cf2.bin
  - cf2.dfu
  - cf2.elf
  - cf2.hex
  - cf2.map
  - CONTRIBUTING.md
  - current_platform.mk
  - LICENSE.txt
  - Makefile
  - module.json
  - Rakefile
  - README.md
  - RELEASE_CHECKLIST.md

Example Applications



Example Application – Hello World

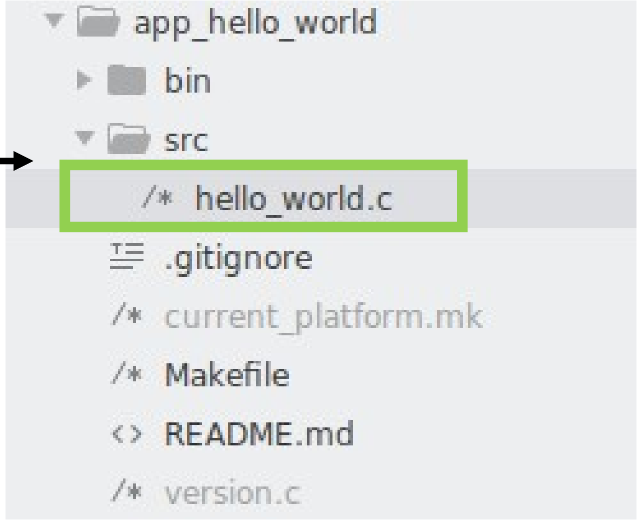


```
▾ app_hello_world
  ▸ bin
  ▸ src
    .gitignore
    current_platform.mk
    Makefile
    README.md
    version.c
```

Project source files. Contains the new code developed by the user.

Project's Makefile. It is appended to the firmware's Makefile. At compilation time, both the firmware and the application get compiled.

Example Application – Hello World

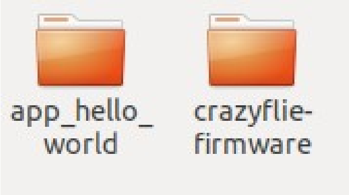


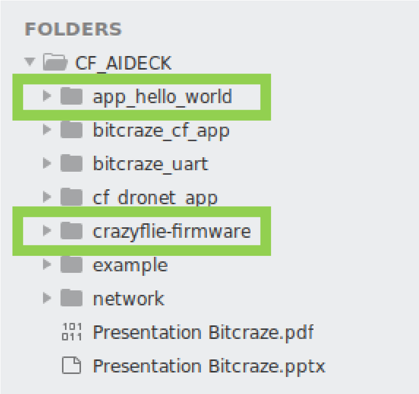
```
app_hello_world
  bin
  src
    hello_world.c
  .gitignore
  current_platform.mk
  Makefile
  README.md
  version.c
```

Source file that contains the application’s code

Moving the application outside the firmware

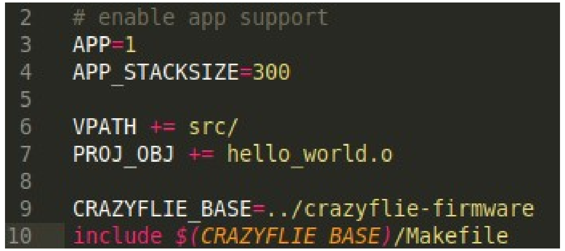
- The application code can be kept outside the main firmware.
- The *app_hello_world* project can be moved at the same level with the *crazyflie-firmware* folder.





```
FOLDERS
▾ CF_AIDECK
    ▸ app_hello_world
    ▸ bitcraze_cf_app
    ▸ bitcraze_uart
    ▸ cf_dronet_app
    ▸ crazyflie-firmware
    ▸ example
    ▸ network
      Presentation Bitcraze.pdf
      Presentation Bitcraze.pptx
```

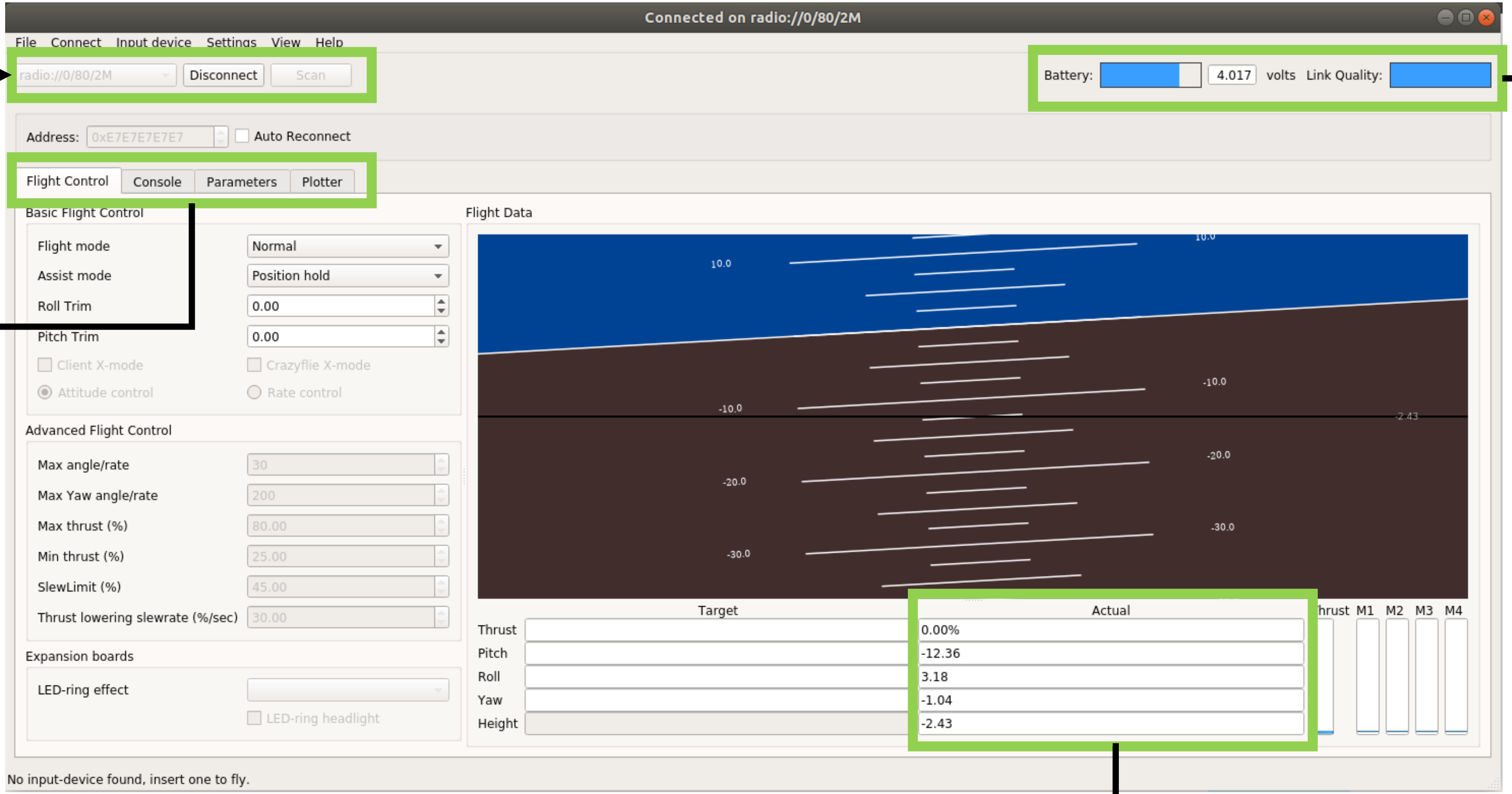
- It is required to inform the application where the firmware folder is located, by modifying its Makefile.



```
# enable app support
APP=1
APP_STACKSIZE=300

VPATH += src/
PROJ_OBJ += hello_world.o

CRAZYFLIE_BASE=../crazyflie-firmware
include $(CRAZYFLIE_BASE)/Makefile
```

The Crazyflie Client - Overview

- Allows the user to interact with the Crazyflie via USB or Radio



Battery: 4.017 volts

Address: 0xE7E7E7E7E7 ☐ Auto Reconnect

Basic Flight Control

Flight mode: Normal

Assist mode: Position hold

Roll Trim: 0.00

Pitch Trim: 0.00

☐ Client X-mode ☐ Crazyflie X-mode

● Attitude control ○ Rate control

Advanced Flight Control

Max angle/rate	30
Max Yaw angle/rate	200
Max thrust (%)	80.00
Min thrust (%)	25.00
SlewLimit (%)	45.00
Thrust lowering slewrate (%/sec)	30.00

Expansion boards

LED-ring effect

☐ LED-ring headlight

	Target	Actual
Thrust		0.00%
Pitch		-12.36
Roll		3.18
Yaw		-1.04
Height		-2.43

No input-device found, insert one to fly.

Connect to the desired drone.

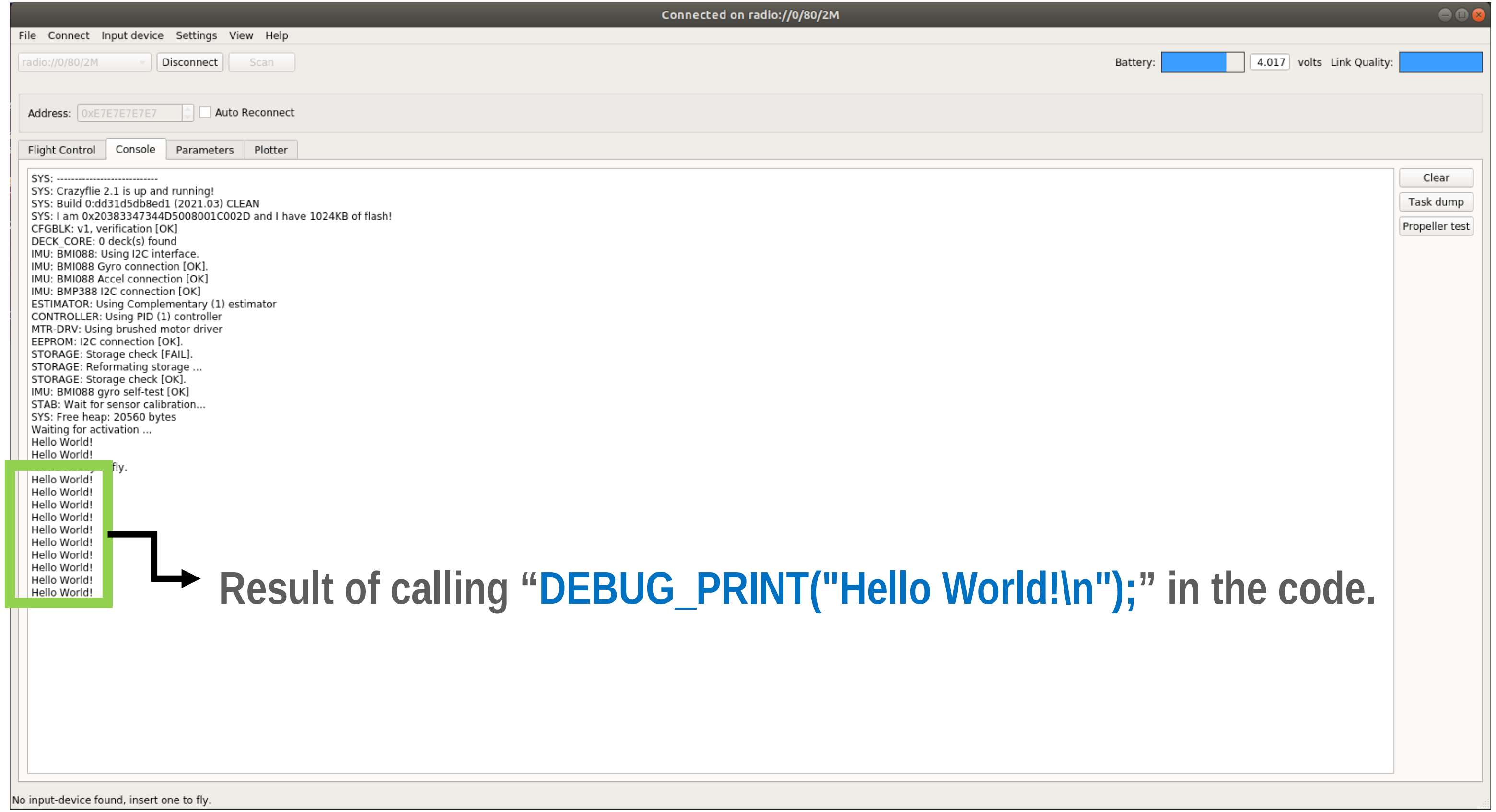
Each tab represent a functionality of the Client. More can be added via the View menu.

Check drone's battery level and Crazyradio's signal strength.

Observe the attitude

The Crazyflie Client - Console

- The console displays what is printed in the firmware via the DEBUG_PRINT function: strings and variables’ values



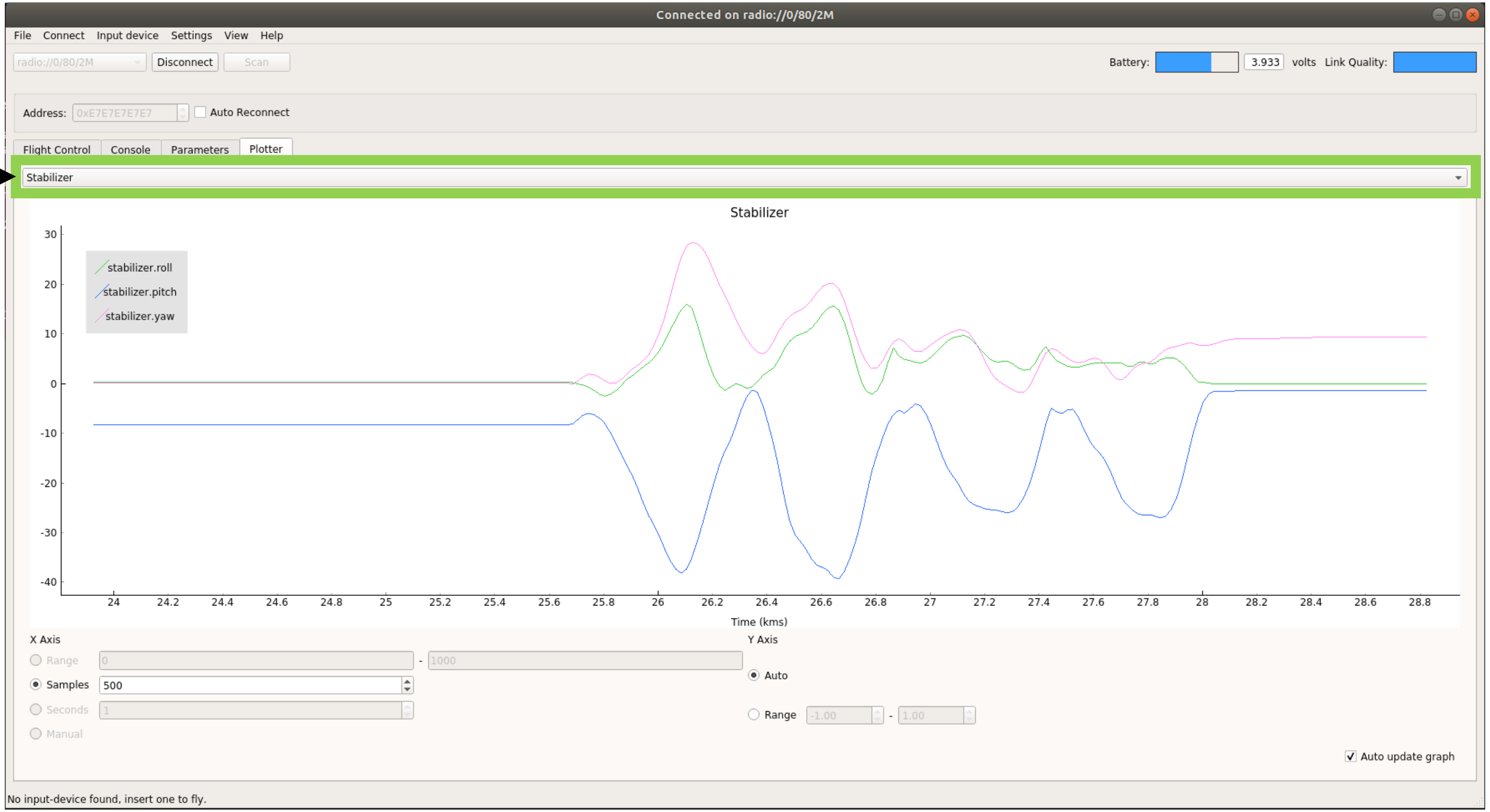
```
SYS: ---------------------------
SYS: Crazyflie 2.1 is up and running!
SYS: Build 0:dd31d5db8ed1 (2021.03) CLEAN
SYS: I am 0x20383347344D5008001C002D and I have 1024KB of flash!
CFGBLK: v1, verification [OK]
DECK_CORE: 0 deck(s) found
IMU: BMI088: Using I2C interface.
IMU: BMI088 Gyro connection [OK].
IMU: BMI088 Accel connection [OK]
IMU: BMP388 I2C connection [OK]
ESTIMATOR: Using Complementary (1) estimator
CONTROLLER: Using PID (1) controller
MTR-DRV: Using brushed motor driver
EEPROM: I2C connection [OK].
STORAGE: Storage check [FAIL].
STORAGE: Reformating storage ...
STORAGE: Storage check [OK].
IMU: BMI088 gyro self-test [OK]
STAB: Wait for sensor calibration...
SYS: Free heap: 20560 bytes
Waiting for activation ...
Hello World!
Hello World!
STAB: Ready to fly.
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

No input-device found, insert one to fly.

Result of calling “DEBUG_PRINT("Hello World!\n");” in the code.

The Crazyflie Client - Plotter

- Allows plotting the logged variables and monitor their evolution in time.



Connected on radio://0/80/2M

radio://0/80/2M

Battery: 3.933 volts Link Quality:

Address: 0xE7E7E7E7E7 □ Auto Reconnect

Stabilizer

X Axis

○ Range 0 - 1000

● Samples 500

○ Seconds 1

○ Manual

Y Axis

● Auto

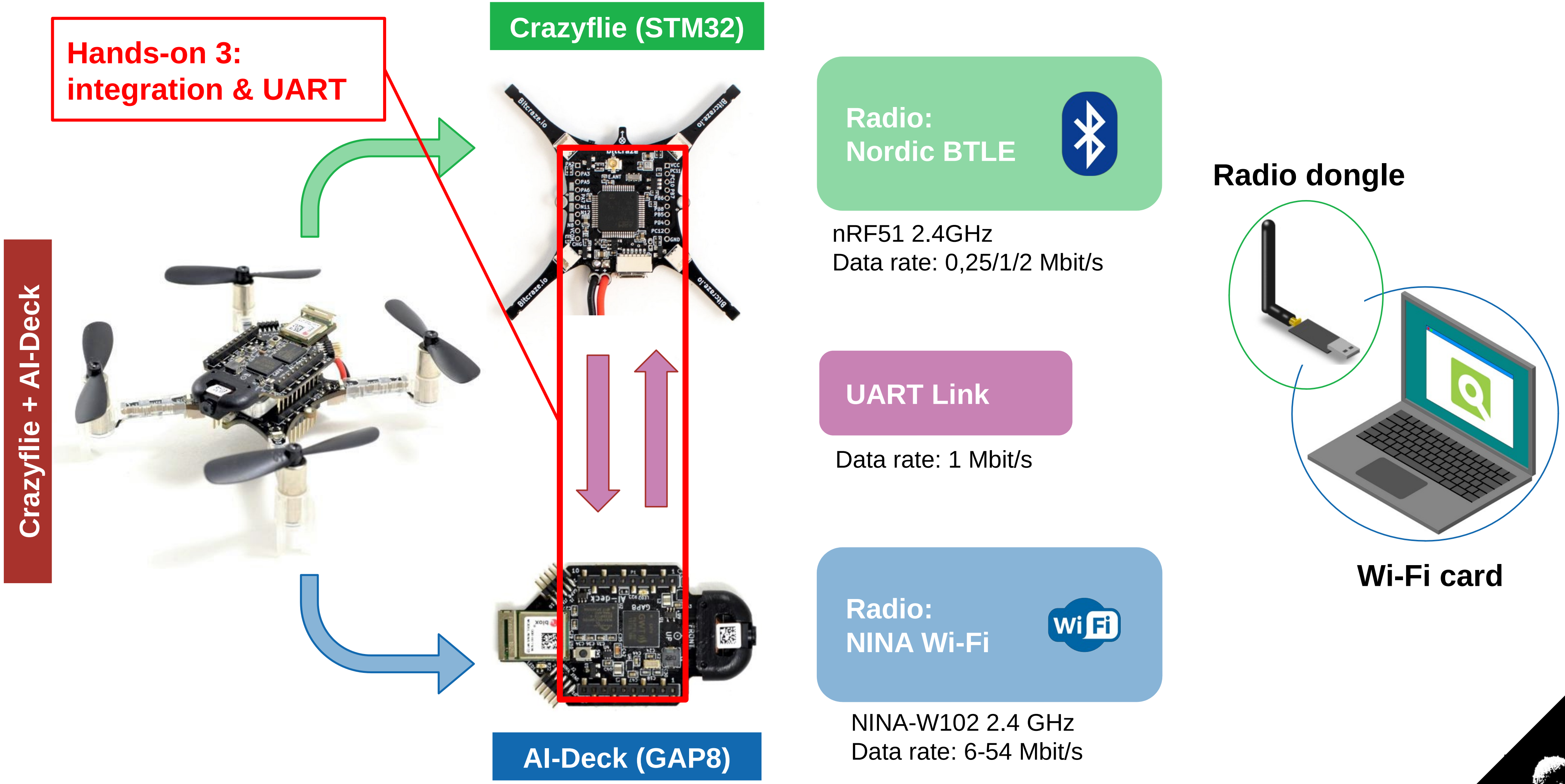
○ Range -1.00 - 1.00

☑ Auto update graph

No input-device found, insert one to fly.

Select variables to plot.

The AI-Deck



Hands-on 3:
integration & UART

Crazyflie (STM32)

Crazyflie + AI-Deck

AI-Deck (GAP8)

Radio:
Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

Radio:
NINA Wi-Fi

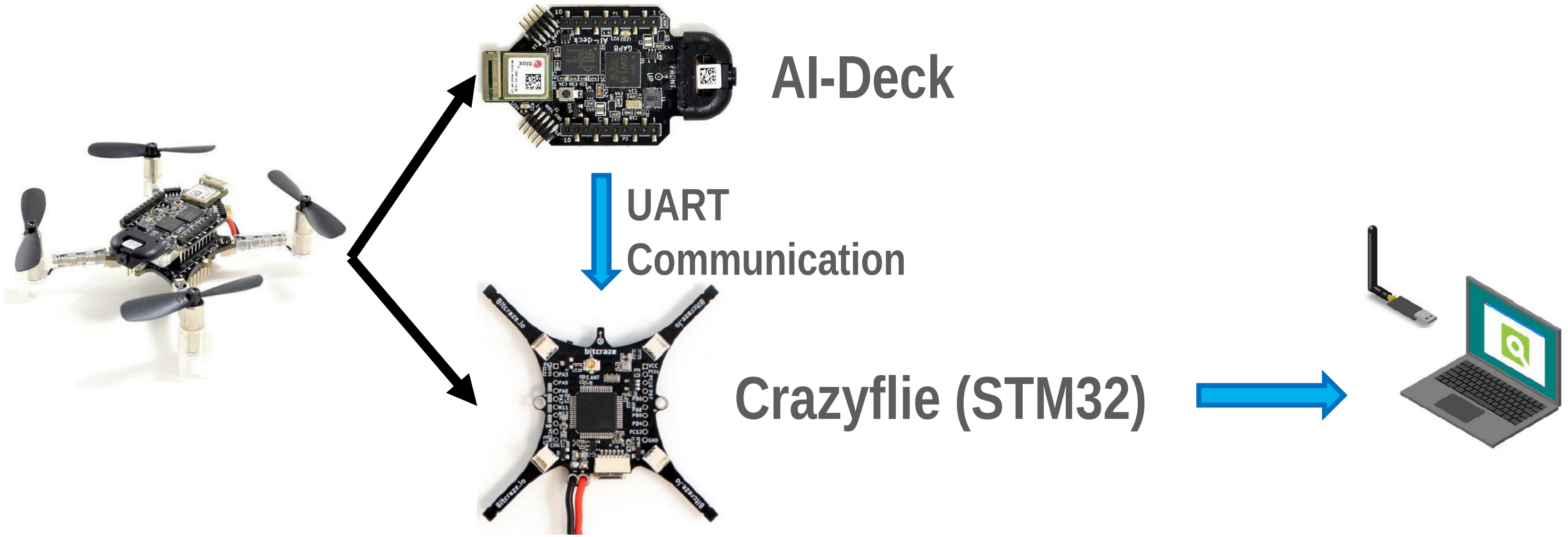
NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

Radio dongle

Wi-Fi card

Application Example

- Example: AI-Deck is sending the value of a counter every 0.5s.
- The Crazyflie prints every value that it receives.
- The Crazyflie uses the UART with DMA, which triggers an interrupt whenever a certain amount of bytes was received.

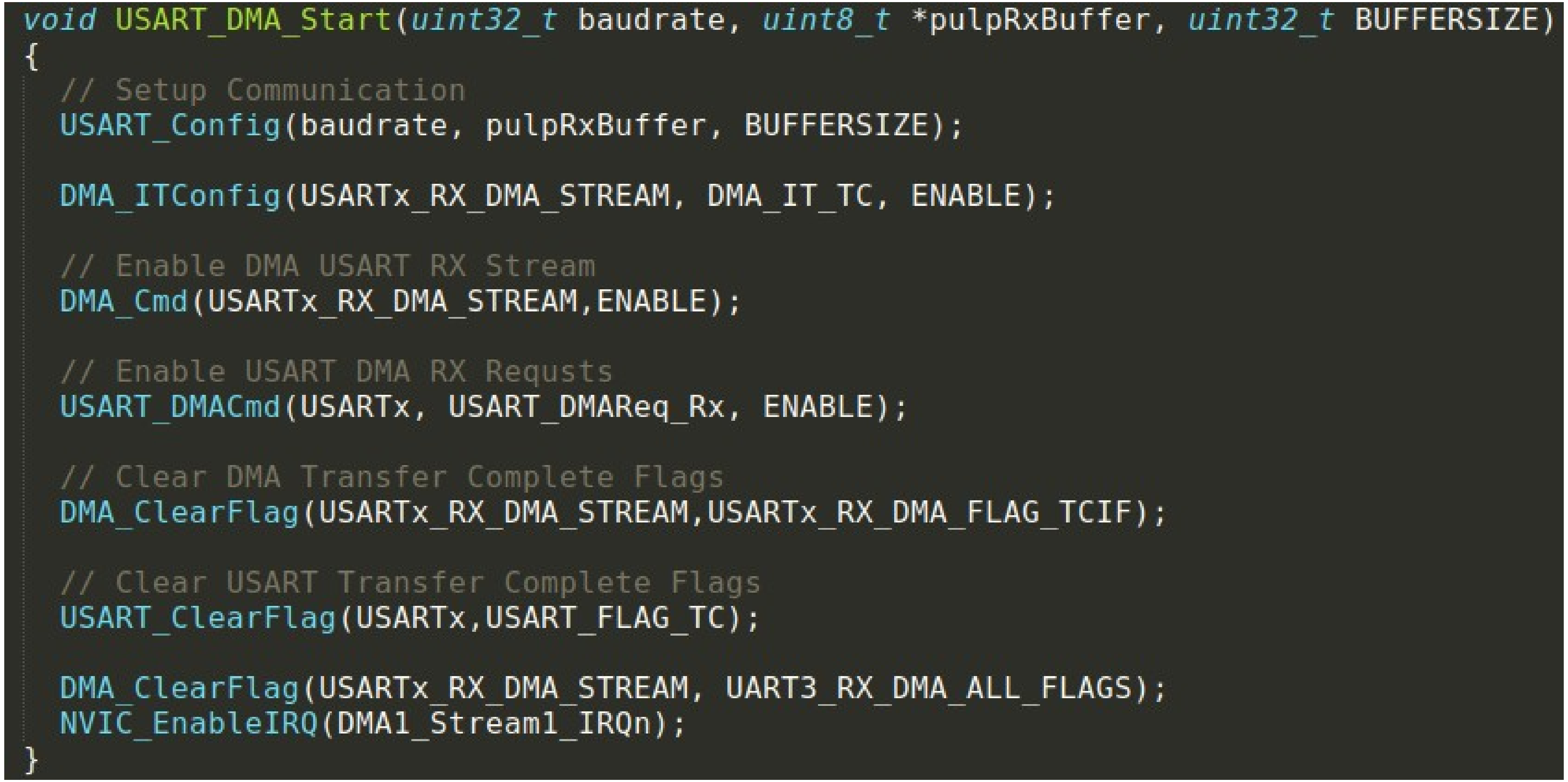


AI-Deck

UART Communication

Crazyflie (STM32)

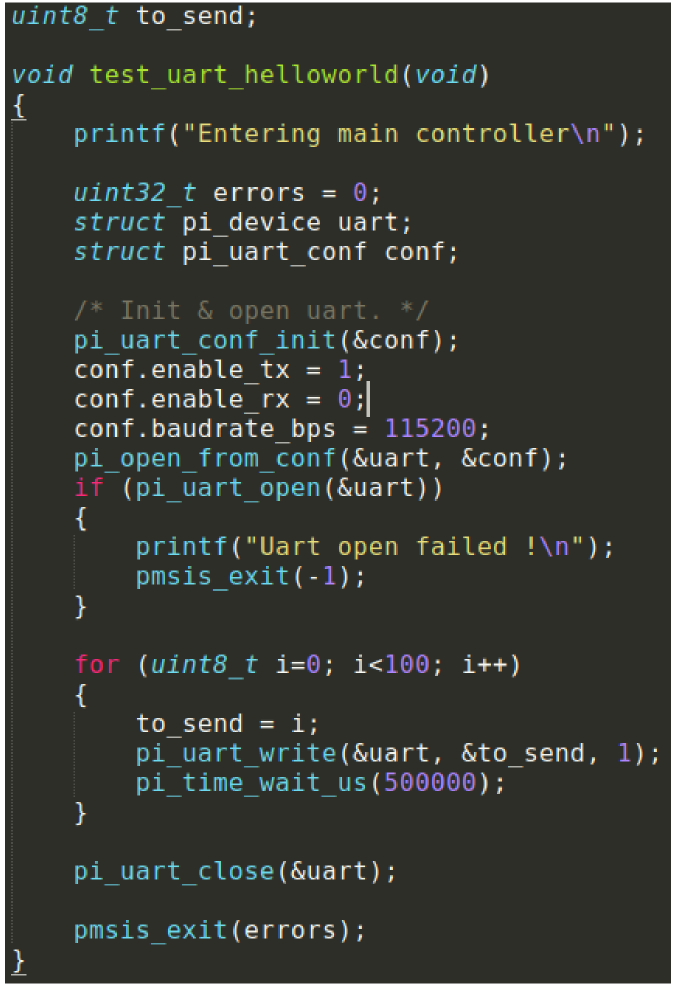
Application Example: UART and DMA



```
void USART_DMA_Start(uint32_t baudrate, uint8_t *pulpRxBuffer, uint32_t BUFFERSIZE)
{
  // Setup Communication
  USART_Config(baudrate, pulpRxBuffer, BUFFERSIZE);

  DMA_ITConfig(USARTx_RX_DMA_STREAM, DMA_IT_TC, ENABLE);

  // Enable DMA USART RX Stream
  DMA_Cmd(USARTx_RX_DMA_STREAM,ENABLE);

  // Enable USART DMA RX Requsts
  USART_DMACmd(USARTx, USART_DMAReq_Rx, ENABLE);

  // Clear DMA Transfer Complete Flags
  DMA_ClearFlag(USARTx_RX_DMA_STREAM,USARTx_RX_DMA_FLAG_TCIF);

  // Clear USART Transfer Complete Flags
  USART_ClearFlag(USARTx,USART_FLAG_TC);

  DMA_ClearFlag(USARTx_RX_DMA_STREAM, UART3_RX_DMA_ALL_FLAGS);
  NVIC_EnableIRQ(DMA1_Stream1_IRQn);
}
```

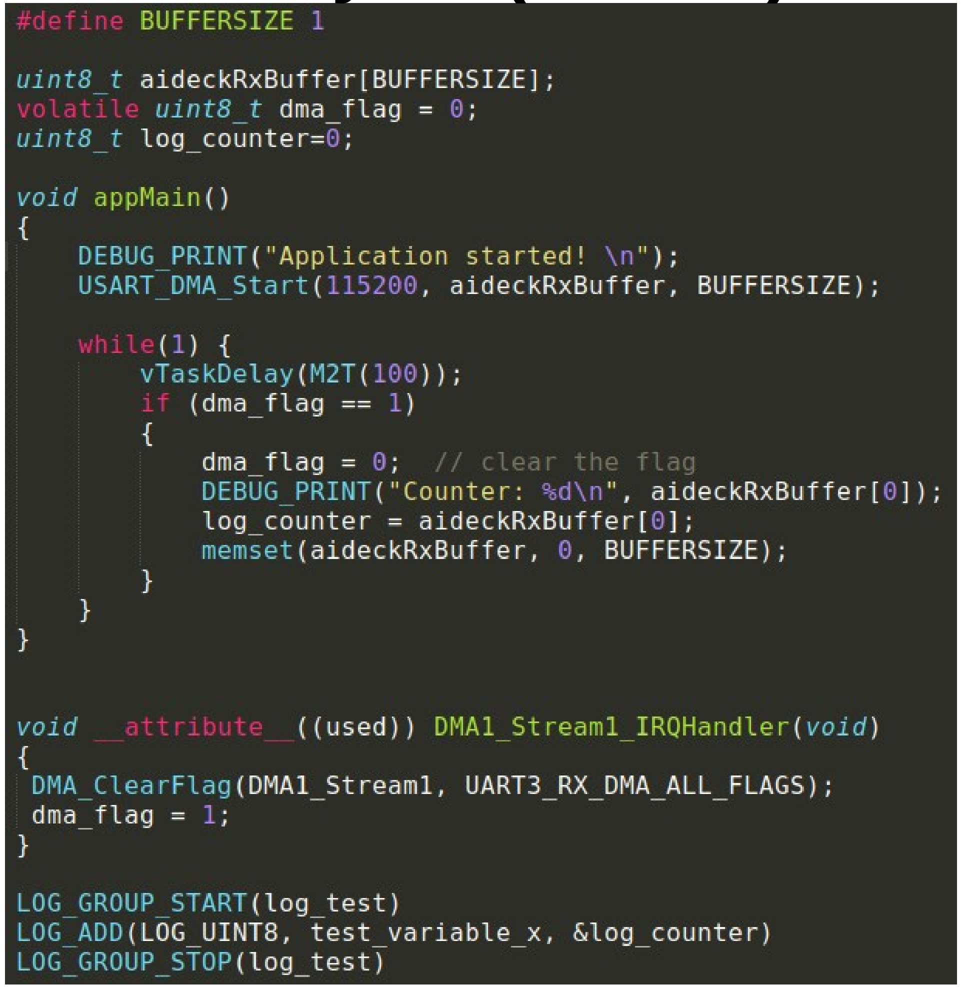
Application Example: Main

AI-Deck



```
uint8_t to_send;

void test_uart_helloworld(void)
{
    printf("Entering main controller\n");

    uint32_t errors = 0;
    struct pi_device uart;
    struct pi_uart_conf conf;

    /* Init & open uart. */
    pi_uart_conf_init(&conf);
    conf.enable_tx = 1;
    conf.enable_rx = 0;
    conf.baudrate_bps = 115200;
    pi_open_from_conf(&uart, &conf);
    if (pi_uart_open(&uart))
    {
        printf("Uart open failed !\n");
        pmsis_exit(-1);
    }

    for (uint8_t i=0; i<100; i++)
    {
        to_send = i;
        pi_uart_write(&uart, &to_send, 1);
        pi_time_wait_us(500000);
    }

    pi_uart_close(&uart);

    pmsis_exit(errors);
}
```

Crazyflie (STM32)

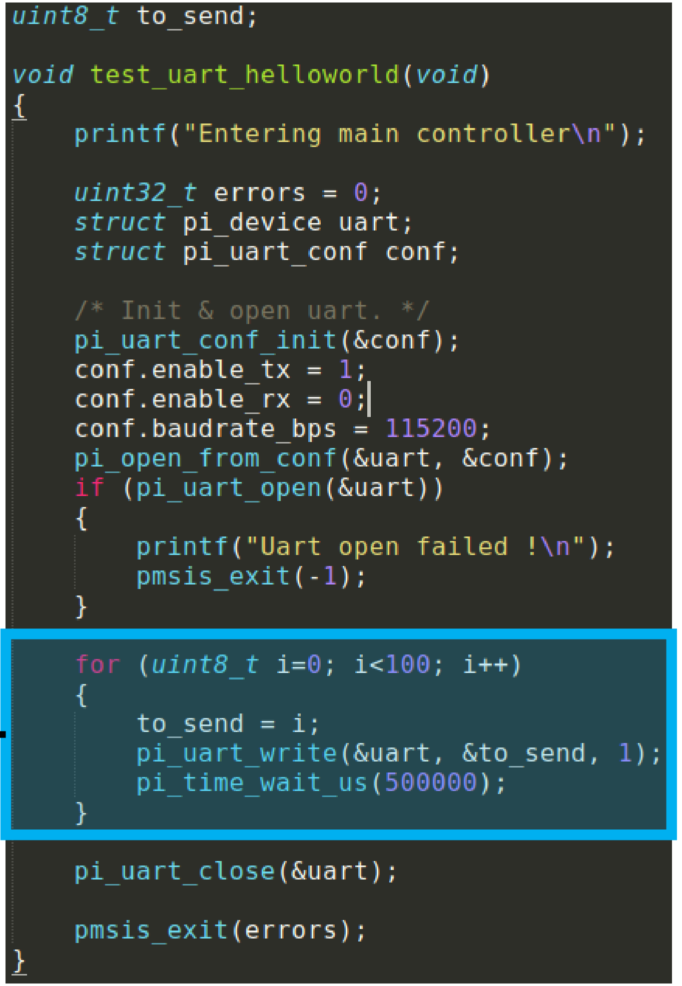


```
#define BUFFERSIZE 1

uint8_t aideckRxBuffer[BUFFERSIZE];
volatile uint8_t dma_flag = 0;
uint8_t log_counter=0;

void appMain()
{
    DEBUG_PRINT("Application started! \n");
    USART_DMA_Start(115200, aideckRxBuffer, BUFFERSIZE);

    while(1) {
        vTaskDelay(M2T(100));
        if (dma_flag == 1)
        {
            dma_flag = 0;  // clear the flag
            DEBUG_PRINT("Counter: %d\n", aideckRxBuffer[0]);
            log_counter = aideckRxBuffer[0];
            memset(aideckRxBuffer, 0, BUFFERSIZE);
        }
    }
}


void __attribute__((used)) DMA1_Stream1_IRQHandler(void)
{
 DMA_ClearFlag(DMA1_Stream1, UART3_RX_DMA_ALL_FLAGS);
 dma_flag = 1;
}

LOG_GROUP_START(log_test)
LOG_ADD(LOG_UINT8, test_variable_x, &log_counter)
LOG_GROUP_STOP(log_test)
```

Application Example: Main

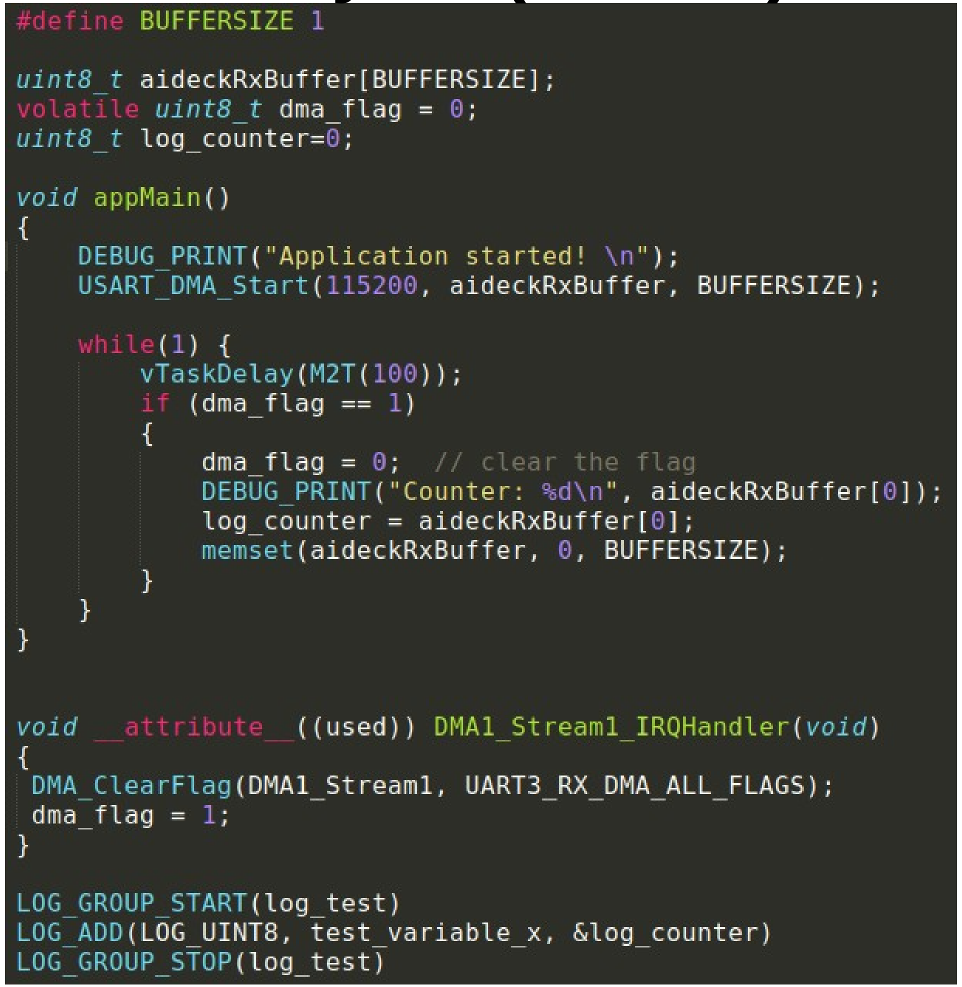
Every 0.5s: increment the counter and send its value via UART

AI-Deck



```c
uint8_t to_send;

void test_uart_helloworld(void)
{
    printf("Entering main controller\n");

    uint32_t errors = 0;
    struct pi_device uart;
    struct pi_uart_conf conf;

    /* Init & open uart. */
    pi_uart_conf_init(&conf);
    conf.enable_tx = 1;
    conf.enable_rx = 0;
    conf.baudrate_bps = 115200;
    pi_open_from_conf(&uart, &conf);
    if (pi_uart_open(&uart))
    {
        printf("Uart open failed !\n");
        pmsis_exit(-1);
    }

    for (uint8_t i=0; i<100; i++)
    {
        to_send = i;
        pi_uart_write(&uart, &to_send, 1);
        pi_time_wait_us(500000);
    }

    pi_uart_close(&uart);

    pmsis_exit(errors);
}
```

Crazyflie (STM32)

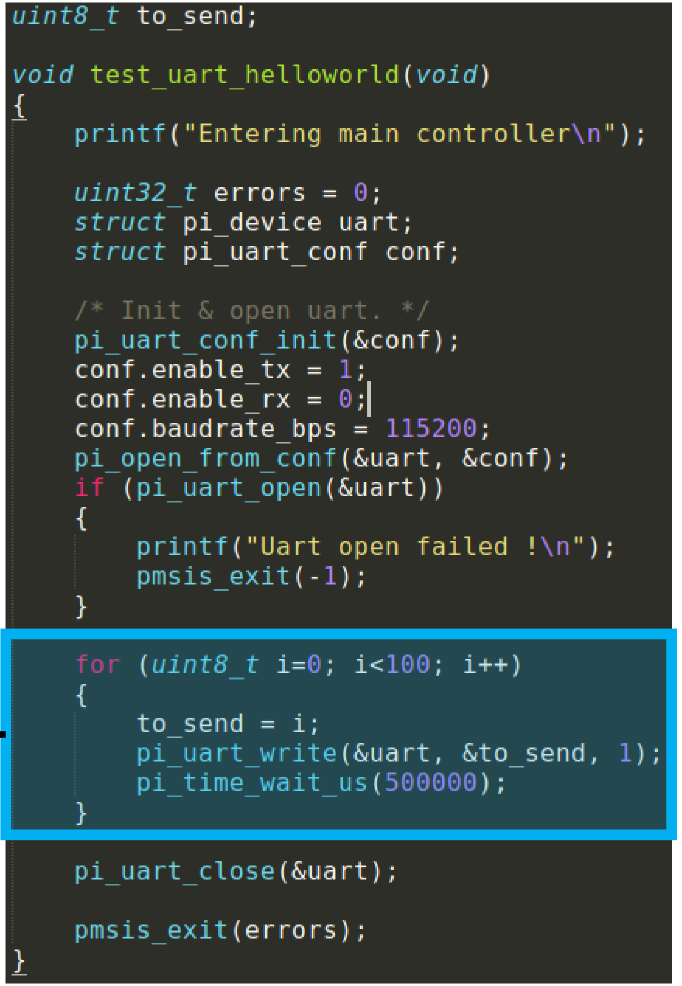


```c
#define BUFFERSIZE 1

uint8_t aideckRxBuffer[BUFFERSIZE];
volatile uint8_t dma_flag = 0;
uint8_t log_counter=0;

void appMain()
{
    DEBUG_PRINT("Application started! \n");
    USART_DMA_Start(115200, aideckRxBuffer, BUFFERSIZE);

    while(1) {
        vTaskDelay(M2T(100));
        if (dma_flag == 1)
        {
            dma_flag = 0;  // clear the flag
            DEBUG_PRINT("Counter: %d\n", aideckRxBuffer[0]);
            log_counter = aideckRxBuffer[0];
            memset(aideckRxBuffer, 0, BUFFERSIZE);
        }
    }
}


void __attribute__((used)) DMA1_Stream1_IRQHandler(void)
{
 DMA_ClearFlag(DMA1_Stream1, UART3_RX_DMA_ALL_FLAGS);
 dma_flag = 1;
}

LOG_GROUP_START(log_test)
LOG_ADD(LOG_UINT8, test_variable_x, &log_counter)
LOG_GROUP_STOP(log_test)
```

Application Example: Main

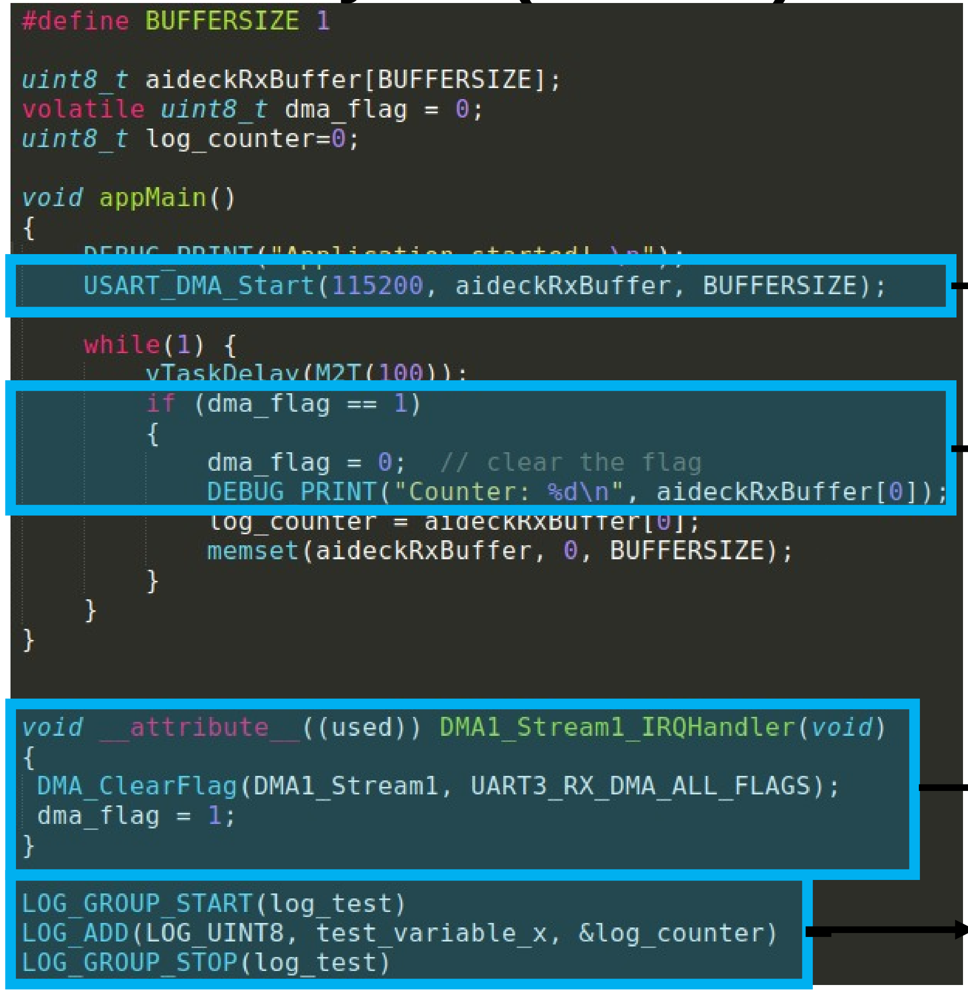
AI-Deck

Every 0.5s: increment the counter and send its value via UART



```
uint8_t to_send;

void test_uart_helloworld(void)
{
    printf("Entering main controller\n");

    uint32_t errors = 0;
    struct pi_device uart;
    struct pi_uart_conf conf;

    /* Init & open uart. */
    pi_uart_conf_init(&conf);
    conf.enable_tx = 1;
    conf.enable_rx = 0;
    conf.baudrate_bps = 115200;
    pi_open_from_conf(&uart, &conf);
    if (pi_uart_open(&uart))
    {
        printf("Uart open failed !\n");
        pmsis_exit(-1);
    }

    for (uint8_t i=0; i<100; i++)
    {
        to_send = i;
        pi_uart_write(&uart, &to_send, 1);
        pi_time_wait_us(500000);
    }

    pi_uart_close(&uart);

    pmsis_exit(errors);
}
```

Crazyflie (STM32)



```
#define BUFFERSIZE 1

uint8_t aideckRxBuffer[BUFFERSIZE];
volatile uint8_t dma_flag = 0;
uint8_t log_counter=0;

void appMain()
{
    DEBUG_PRINT("Application started! \n");
    USART_DMA_Start(115200, aideckRxBuffer, BUFFERSIZE);

    while(1) {
        vTaskDelay(M2T(100));
        if (dma_flag == 1)
        {
            dma_flag = 0;  // clear the flag
            DEBUG_PRINT("Counter: %d\n", aideckRxBuffer[0]);
            log_counter = aideckRxBuffer[0];
            memset(aideckRxBuffer, 0, BUFFERSIZE);
        }
    }
}


void __attribute__((used)) DMA1_Stream1_IRQHandler(void)
{
 DMA_ClearFlag(DMA1_Stream1, UART3_RX_DMA_ALL_FLAGS);
 dma_flag = 1;
}

LOG_GROUP_START(log_test)
LOG_ADD(LOG_UINT8, test_variable_x, &log_counter)
LOG_GROUP_STOP(log_test)
```

Init DMA and UART

If the flag is set, print the received value

DMA "full buffer" interrupt

Define log

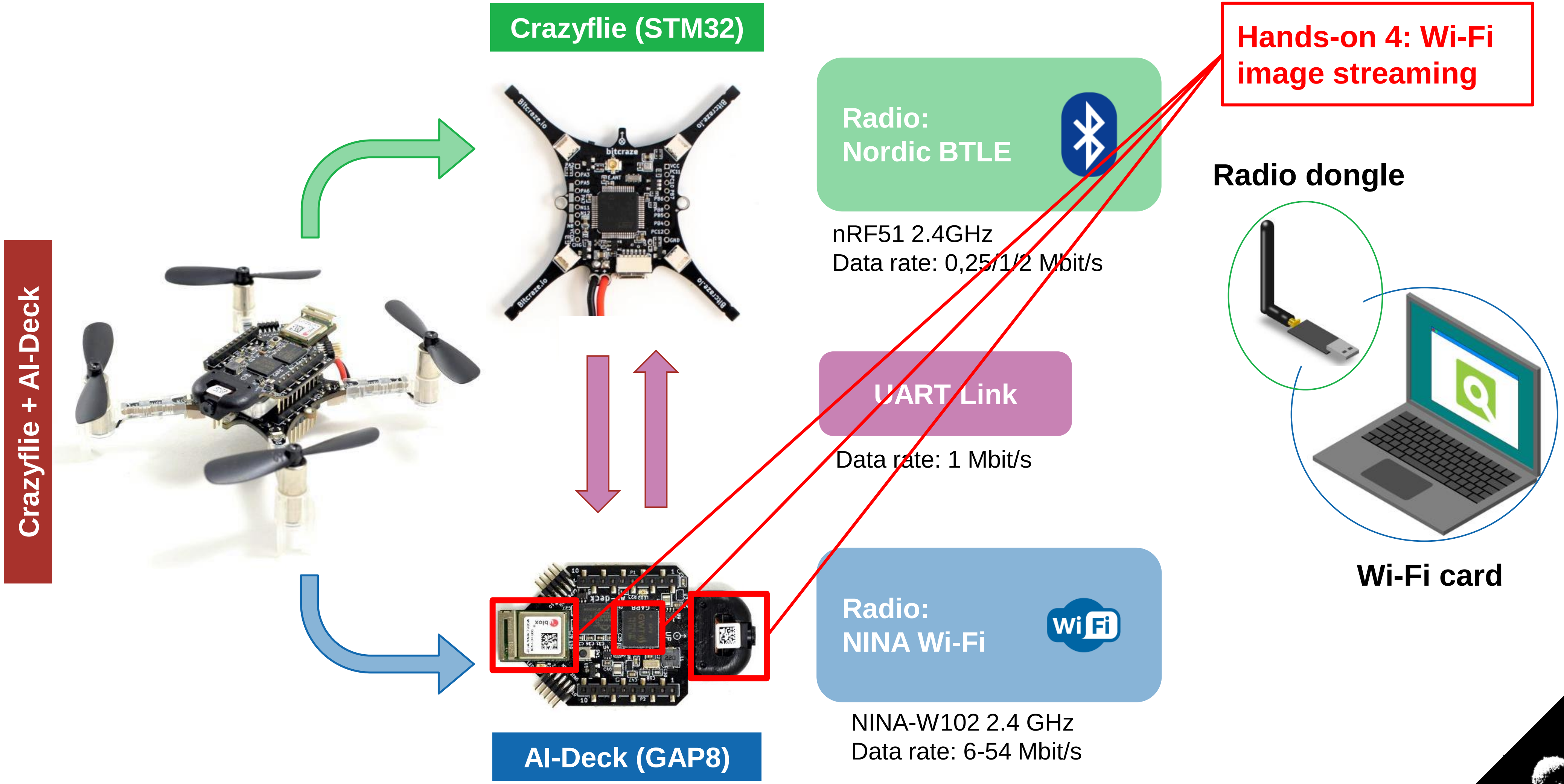
Hands-on

Hands-on demonstration of the system’s functionality

Bitcraze Workshop: Hands-on Session 4
Wi-Fi image streaming with AI-Deck

Lorenzo Lamberti, Hanna Müller, Vlad Niculescu, Manuele Rusci, Daniele Palossi

Hands-on session 4



Crazyflie (STM32)

Crazyflie + AI-Deck

Radio: Nordic BTLE

nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

Radio: NINA Wi-Fi

NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

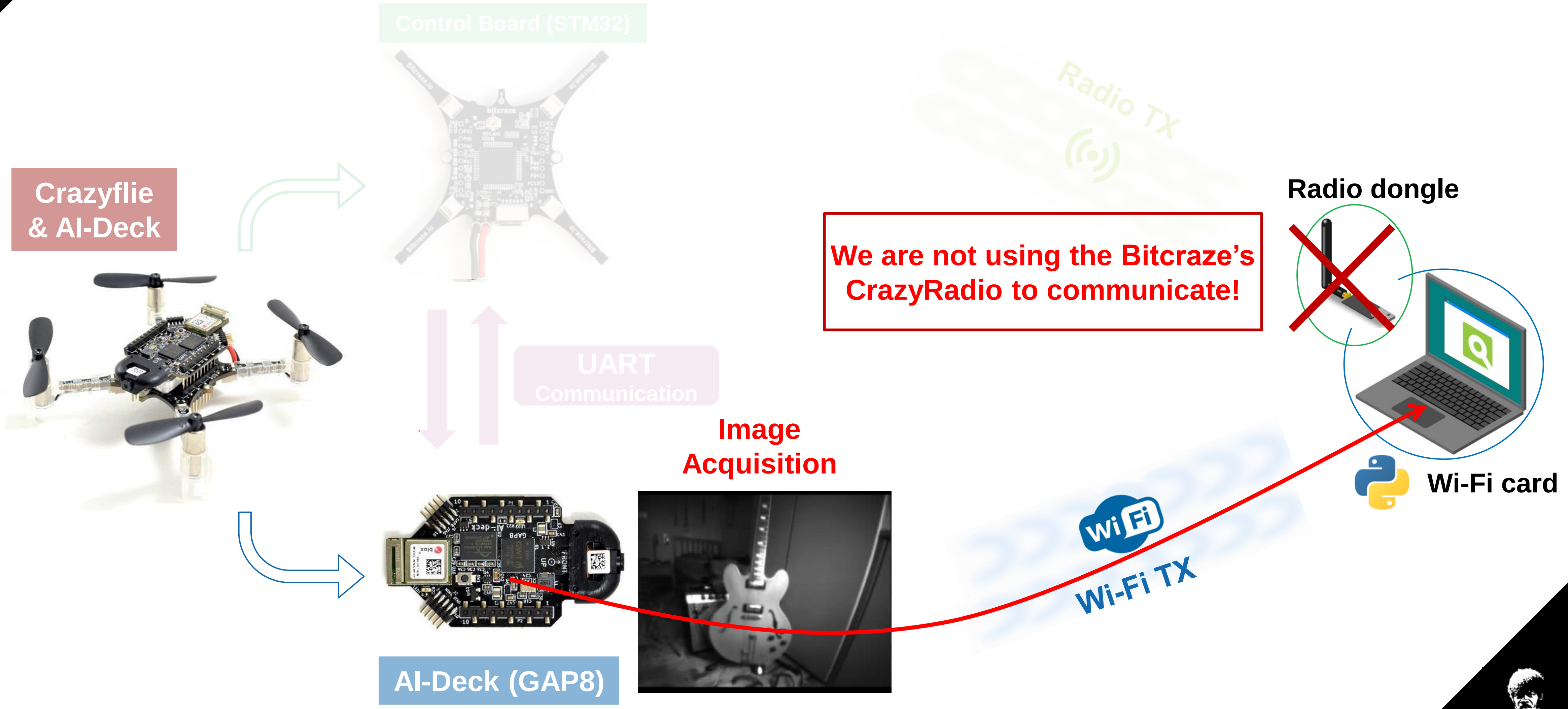
AI-Deck (GAP8)

Hands-on 4: Wi-Fi image streaming

Radio dongle

Wi-Fi card

Image streaming via Wi-Fi



Crazyflie & AI-Deck

Control Board (STM32)

UART Communication

AI-Deck (GAP8)

Image Acquisition

Radio TX

Radio dongle

We are not using the Bitcraze's CrazyRadio to communicate!

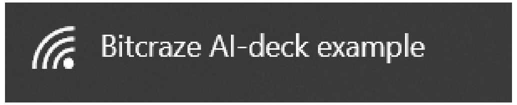
Wi-Fi TX

Wi-Fi card

Hands-on overview

The example is inside the Bitcraze GitHub repository, and it is called wifi_jpeg_streamer
Code: https://github.com/bitcraze/AIdeck_examples/blob/master/GAP8/test_functionalities/wifi_jpeg_streamer

Default Network SSID:

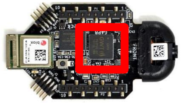


Bitcraze AI-deck example

- Create a Wi-Fi access-point with the NINA Wi-Fi module
- Establish a **point-to-point** Wi-Fi connection between laptop and AI-Deck

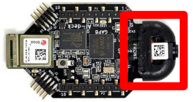
- **Acquisition of an image**
- **Compression** (JPEG)
- **Wi-Fi transmission** of the image
- **Bonus task:** pre-processing the image before transmission

Wi-Fi Image streaming: Initial setup



FC: Fabric Controller
CL: Cluster (8-cores)

Input image buffer:
L2 memory allocation



QVGA format (320x240 pixel)
AEG: Auto-Exposure Gain

AI-DeckV1: RGB Bayer
AI-DeckV2: Grayscale

Initial Setup

Main loop

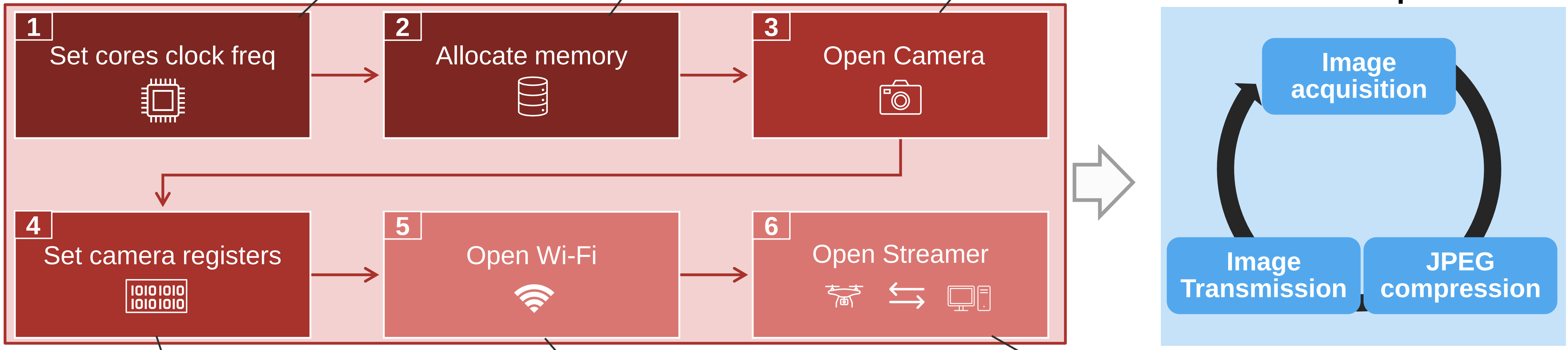
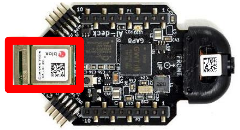


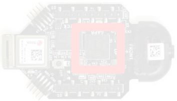
Image orientation
(rotate 180°)

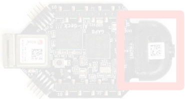
Tells **NINA** to open Wi-Fi access-point;
Set Wi-Fi **SSID** and **port**

Sets Wi-Fi as **TX channel**;
Starts **JPEG encoder**;



Wi-Fi Image streaming: Initial setup





AI-DeckV1: RGB Bayer
AI-DeckV2: Grayscale

FC: Fabric Controller
CL: Cluster (8-cores)

Image: **Static L2 memory** allocation

QVGA format (320x240 pixel)
AEG: Auto-Exposure Gain

Initial Setup

1 Set cores clock freq

2 Allocate memory

3 Open Camera

4 Set camera registers

5 Open Wi-Fi

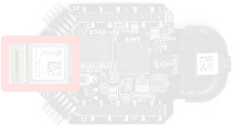
6 Open Streamer

Image orientation
(rotate 180°)

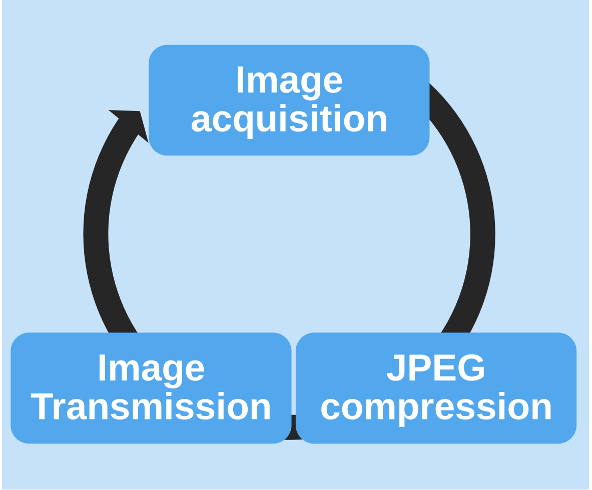
Tells **NINA** to open Wi-Fi connection;
Set Wi-Fi **SSID and port**

Sets Wi-Fi as **TX channel**;
Starts **JPEG encoder**;
Defines the **transmission chunk size**;

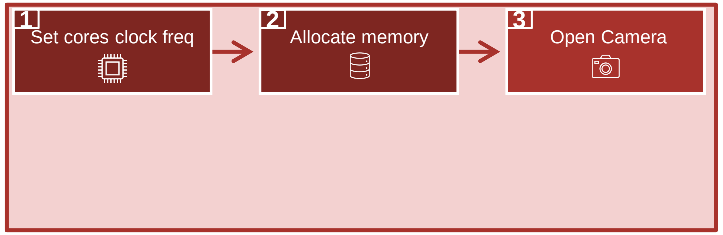
```
#define JPEG_BITSTREAM_SIZE (1024)
```

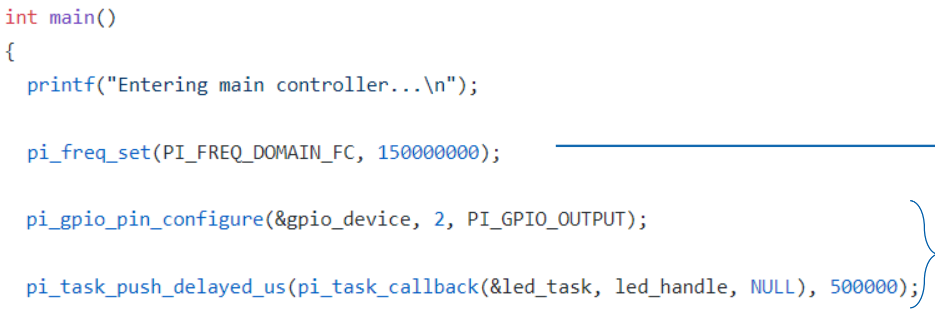


Main loop



Code inspection: Initial setup

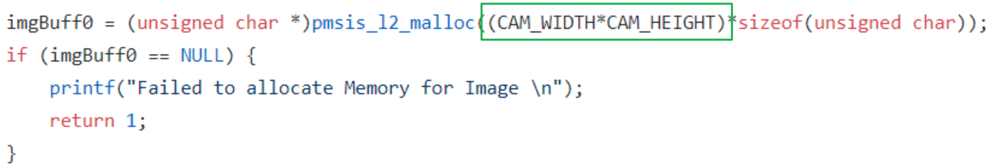




```
int main()
{
  printf("Entering main controller...\n");

  pi_freq_set(PI_FREQ_DOMAIN_FC, 150000000);

  pi_gpio_pin_configure(&gpio_device, 2, PI_GPIO_OUTPUT);

  pi_task_push_delayed_us(pi_task_callback(&led_task, led_handle, NULL), 500000);
```

1. Set the core frequency of the main GAP8’s core (FC = Fabric Controller)

We configure the LED GPIO (LED#2) to “output mode” so that we can control it.
Then we start the blinking task: `led_handle()`

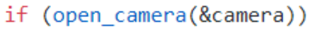


```
imgBuff0 = (unsigned char *)pmsis_l2_malloc((CAM_WIDTH*CAM_HEIGHT)*sizeof(unsigned char));
if (imgBuff0 == NULL) {
    printf("Failed to allocate Memory for Image \n");
    return 1;
}
```

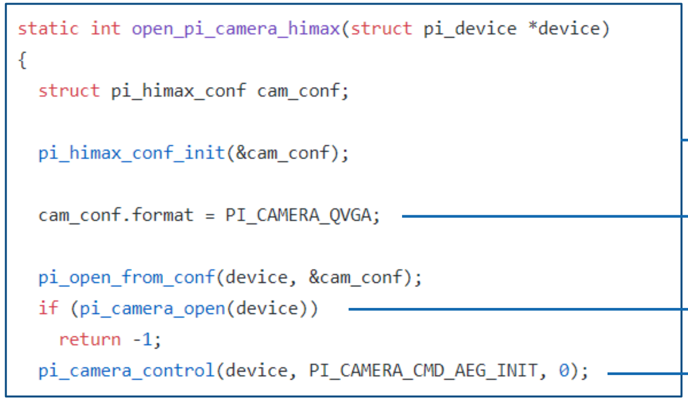
2. Allocate the memory for the image (QVGA format)

- CAM_WIDTH = 320
- CAM HEIGHT = 240

We use the L2 memory (512Kb), which is enough for storing an image.
In GAP8 you must specify the target memory for the malloc (L2 in this case).



```
if (open_camera(&camera))
```



```
static int open_pi_camera_himax(struct pi_device *device)
{
  struct pi_himax_conf cam_conf;

  pi_himax_conf_init(&cam_conf);

  cam_conf.format = PI_CAMERA_QVGA;

  pi_open_from_conf(device, &cam_conf);
  if (pi_camera_open(device))
    return -1;
  pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0);
```

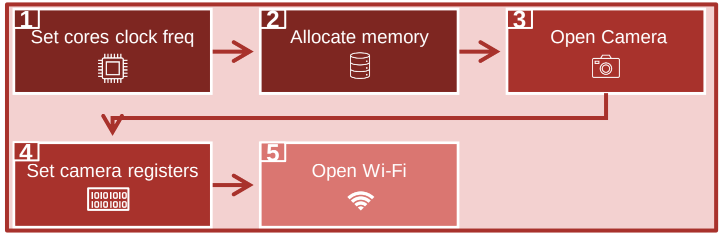
3. Open the camera

We specify the format between QVGA and QQVGA

Camera is opened

The AEG= auto-exposure-gain is activated

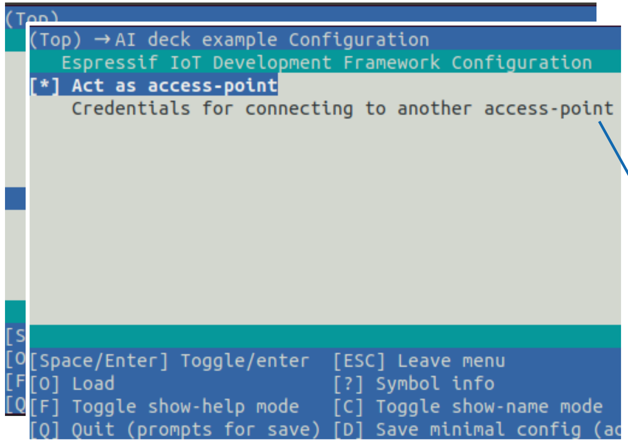
Code inspection: Initial setup



```
pi_camera_reg_set(&camera, IMG_ORIENTATION, &set_value);
```

4. Set the camera registers to rotate the image by 180° (the image is upside-down by default).

```
if (open_wifi(&wifi))
```



```
(Top) → AI deck example Configuration
    Espressif IoT Development Framework Configuration
[*] Act as access-point
    Credentials for connecting to another access-point

[Space/Enter] Toggle/enter   [ESC] Leave menu
[O] Load                     [?] Symbol info
[F] Toggle show-help mode    [C] Toggle show-name mode
[Q] Quit (prompts for save)  [D] Save minimal config (ac
```

5. Open Wi-Fi

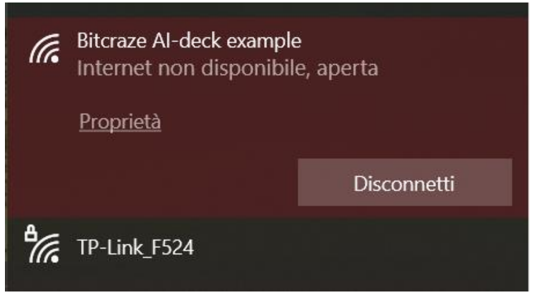
We open the Wi-Fi connection of the NINA Wi-Fi on-board module.

The configuration of NINA is loaded. To change it, you must modify the configuration and flash NINA

```
cd AIdeck_examples/NINA/firmware/
make menuconfig
(then follow instructions to flash NINA)
```

Instead of opening an access-point, you can also chose to connect to an existing one

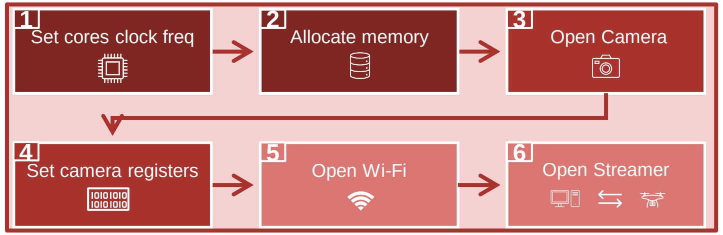
Now the "***Bitcraze AI-deck example***" SSID will appear in the Wi-Fi connections available.
We can **connect to it** with our Laptop (point-to-point).



Bitcraze AI-deck example
Internet non disponibile, aperta

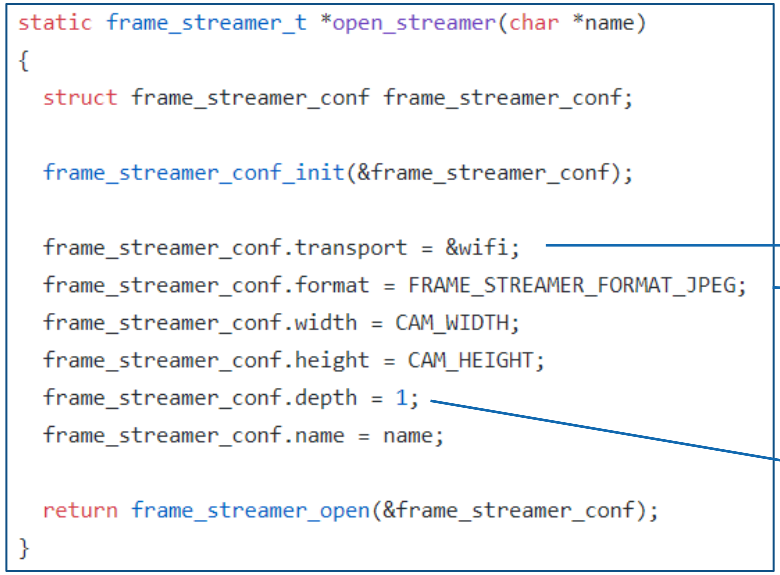
TP-Link_F524

Code inspection: Initial setup



```
streamer1 = open_streamer("camera");
```

6. Open the streamer



```
static frame_streamer_t *open_streamer(char *name)
{
  struct frame_streamer_conf frame_streamer_conf;

  frame_streamer_conf_init(&frame_streamer_conf);

  frame_streamer_conf.transport = &wifi;
  frame_streamer_conf.format = FRAME_STREAMER_FORMAT_JPEG;
  frame_streamer_conf.width = CAM_WIDTH;
  frame_streamer_conf.height = CAM_HEIGHT;
  frame_streamer_conf.depth = 1;
  frame_streamer_conf.name = name;

  return frame_streamer_open(&frame_streamer_conf);
}
```

We select Wi-Fi to stream images

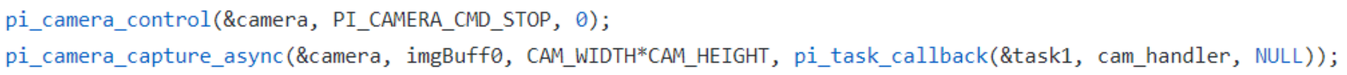
We choose the image format

- **FRAME_STREAMER_FORMAT_JPEG**: enables the JPEG encoder
- **FRAME_STREAMER_FORMAT_RAW**: does not enable the JPEG encoder and streams raw images

Image channels: Grayscale=1, RGB =3.
(But the Bayer RGB sensor AI-DeckV1 still uses one channel !)

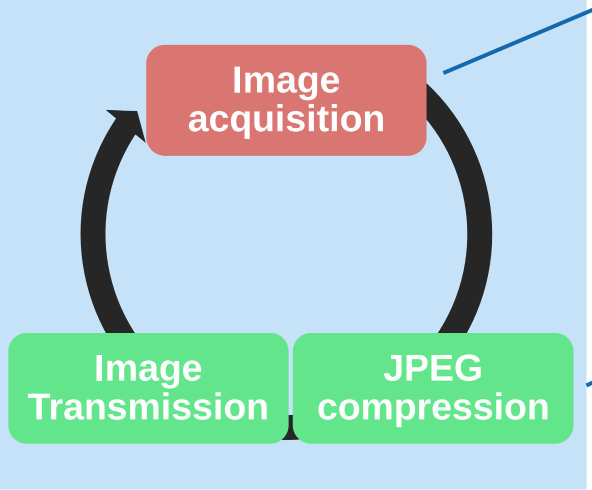
Hand-shaking between GAP8 and NINA Wi-Fi Module and the JPEG encoder is started.

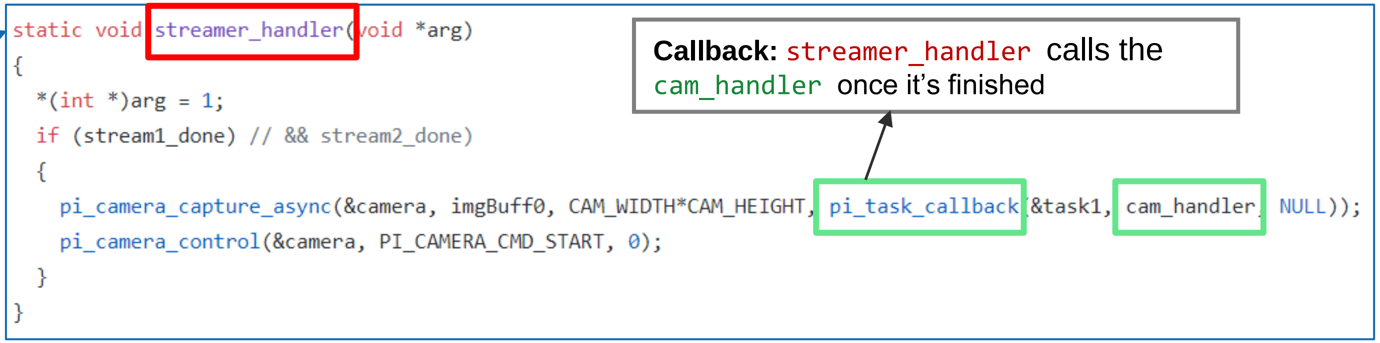
Code inspection: Wi-Fi images transmission



```
pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
pi_camera_capture_async(&camera, imgBuff0, CAM_WIDTH*CAM_HEIGHT, pi_task_callback(&task1, cam_handler, NULL));
```

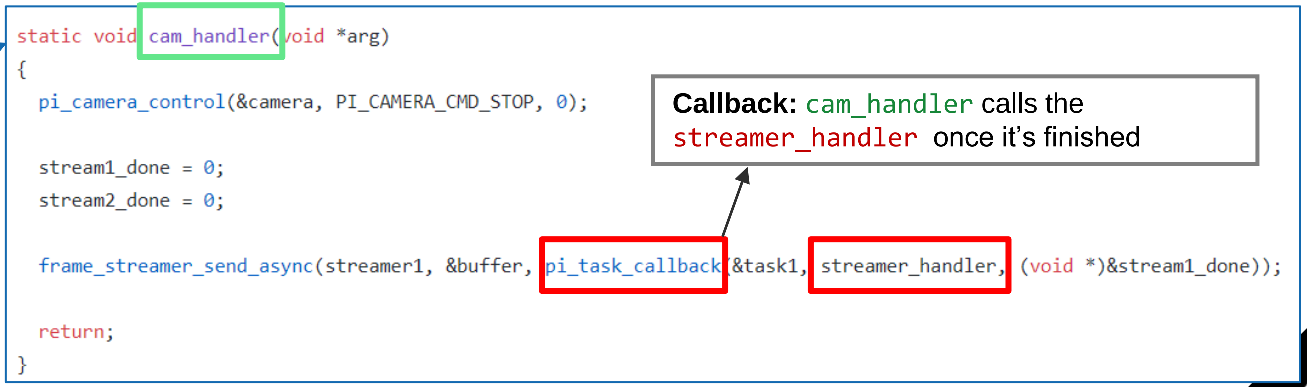
First image acquisition **starts the Main Loop**





```
static void streamer_handler(void *arg)
{
  *(int *)arg = 1;
  if (stream1_done) // && stream2_done)
  {
    pi_camera_capture_async(&camera, imgBuff0, CAM_WIDTH*CAM_HEIGHT, pi_task_callback(&task1, cam_handler, NULL));
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
  }
}
```

Callback: `streamer_handler` calls the `cam_handler` once it's finished



```
static void cam_handler(void *arg)
{
  pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);

  stream1_done = 0;
  stream2_done = 0;

  frame_streamer_send_async(streamer1, &buffer, pi_task_callback(&task1, streamer_handler, (void *)&stream1_done));

  return;
}
```

Callback: `cam_handler` calls the `streamer_handler` once it's finished

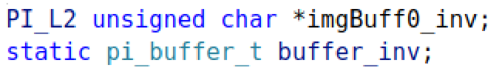
Hands on the code!!

Image manipulation before TX

We can manipulate the images before sending them via Wi-Fi:

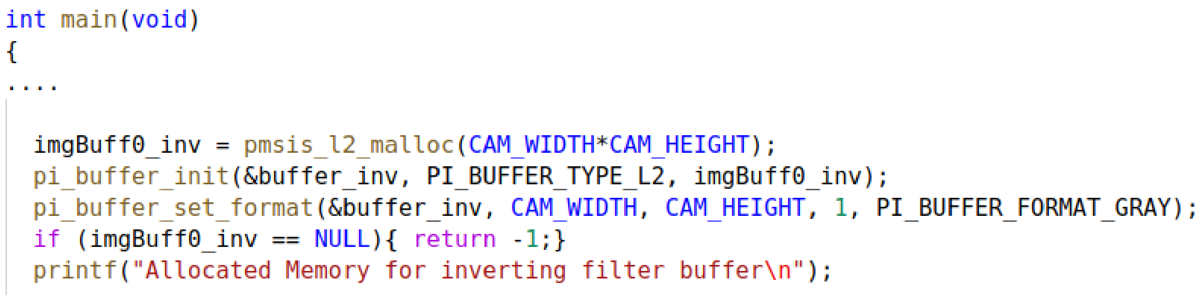
- We will be applying the same `inverting()` kernel that we used in the Hands-on session 2! →

`inverting()` inverts black & white in the image



```
PI_L2 unsigned char *imgBuff0_inv;
static pi_buffer_t buffer_inv;
```

→ Define a buffer as a global variable



```
int main(void)
{
....
  imgBuff0_inv = pmsis_l2_malloc(CAM_WIDTH*CAM_HEIGHT);
  pi_buffer_init(&buffer_inv, PI_BUFFER_TYPE_L2, imgBuff0_inv);
  pi_buffer_set_format(&buffer_inv, CAM_WIDTH, CAM_HEIGHT, 1, PI_BUFFER_FORMAT_GRAY);
  if (imgBuff0_inv == NULL){ return -1;}
  printf("Allocated Memory for inverting filter buffer\n");
```

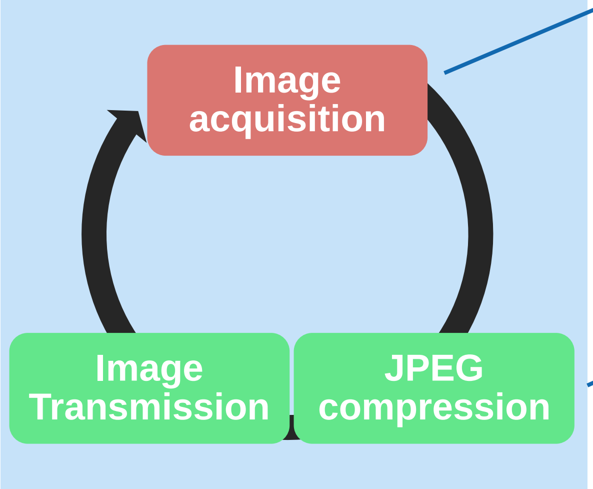
→ We allocate the memory for another image in the L2 memory

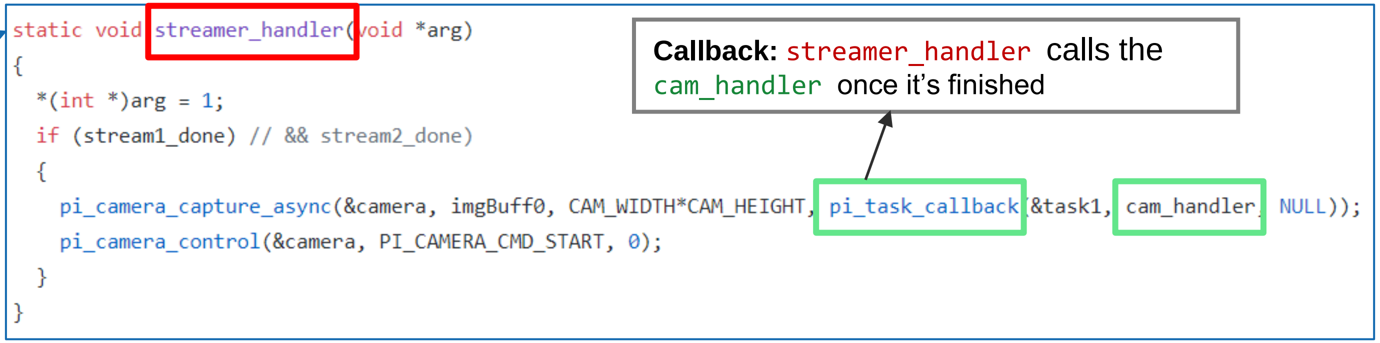
Image manipulation before TX

We keep the very same loop for transmission that we saw before, **but we manipulate the image with the `inverting()` function right before sending it**

`inverting()` inverts black & white in the image

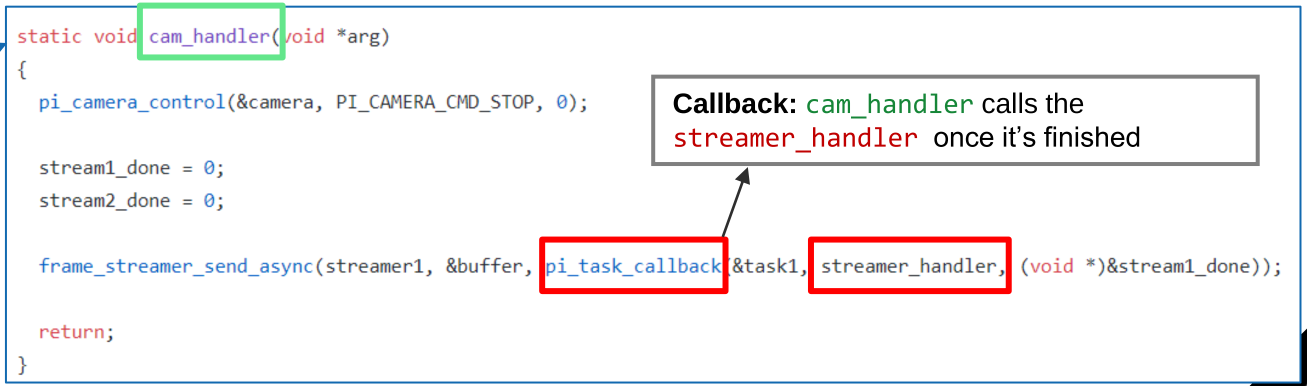
Main loop





```
static void streamer_handler(void *arg)
{
  *(int *)arg = 1;
  if (stream1_done) // && stream2_done)
  {
    pi_camera_capture_async(&camera, imgBuff0, CAM_WIDTH*CAM_HEIGHT, pi_task_callback(&task1, cam_handler, NULL));
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
  }
}
```

Callback: `streamer_handler` calls the `cam_handler` once it's finished



```
static void cam_handler(void *arg)
{
  pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);

  stream1_done = 0;
  stream2_done = 0;

  frame_streamer_send_async(streamer1, &buffer, pi_task_callback(&task1, streamer_handler, (void *)&stream1_done));

  return;
}
```

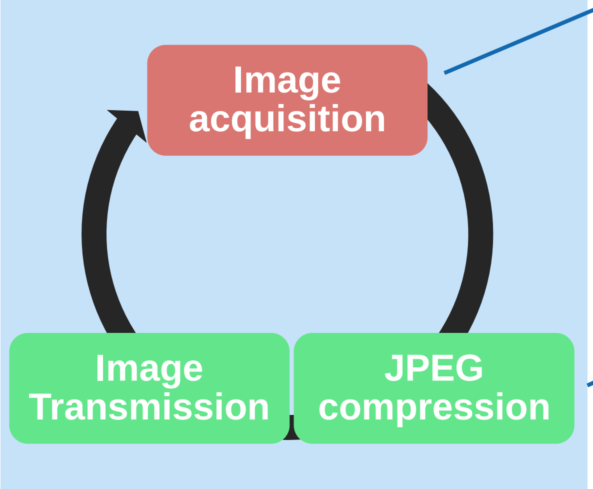
Callback: `cam_handler` calls the `streamer_handler` once it's finished

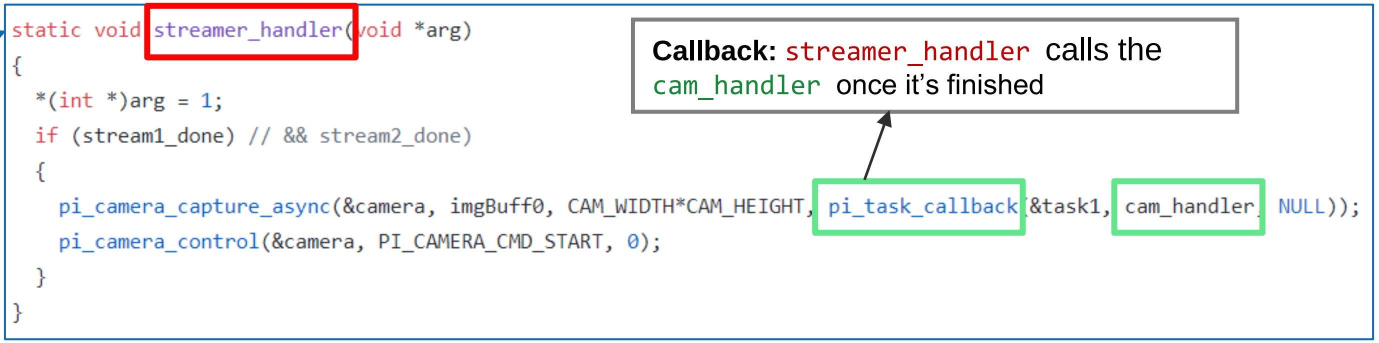
Image manipulation before TX

We keep the very same loop for transmission that we saw before, **but we manipulate the image with the `inverting()` function right before sending it**

→ `inverting()` inverts black & white in the image

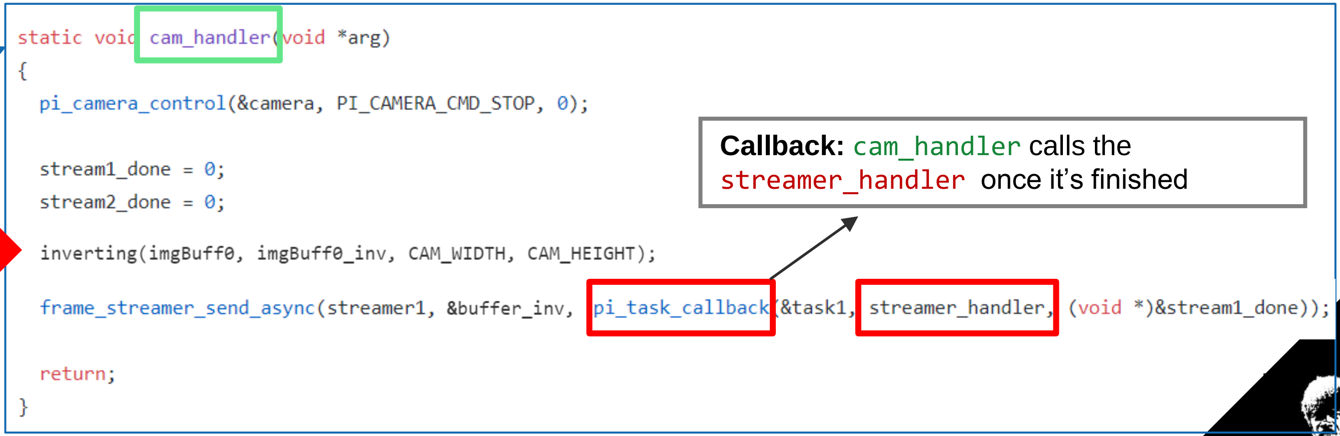
Main loop





```
static void streamer_handler(void *arg)
{
  *(int *)arg = 1;
  if (stream1_done) // && stream2_done)
  {
    pi_camera_capture_async(&camera, imgBuff0, CAM_WIDTH*CAM_HEIGHT, pi_task_callback(&task1, cam_handler, NULL));
    pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
  }
}
```

Callback: `streamer_handler` calls the `cam_handler` once it’s finished

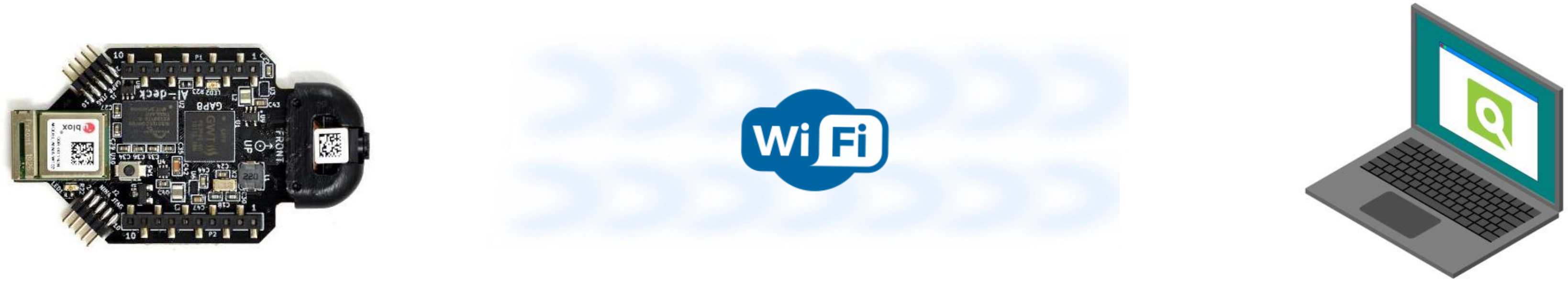


```
static void cam_handler(void *arg)
{
  pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);

  stream1_done = 0;
  stream2_done = 0;

  inverting(imgBuff0, imgBuff0_inv, CAM_WIDTH, CAM_HEIGHT);

  frame_streamer_send_async(streamer1, &buffer_inv, pi_task_callback(&task1, streamer_handler, (void *)&stream1_done));

  return;
}
```

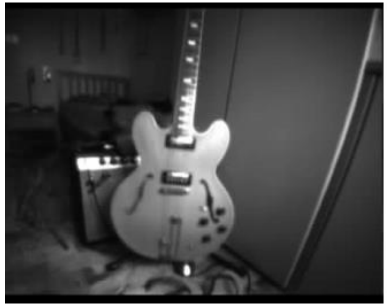
Callback: `cam_handler` calls the `streamer_handler` once it’s finished

Image manipulation before TX

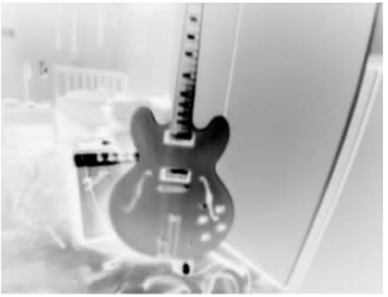
This is the behavior that we will experience



`~~inverting()~~` **(Deactivated)**



`inverting()` **(Activated)**



Hands on the code!!

Thank you for your attention