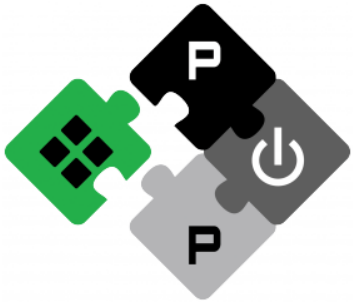


PULP PLATFORM
Open Source Hardware, the way it should be!

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)

Radio:
Nordic BTLE

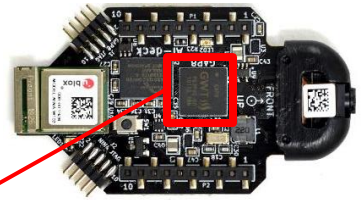
nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

Radio dongle

Crazyflie + AI-Deck

UART Link

Data rate: 1 Mbit/s



Radio:
NINA Wi-Fi

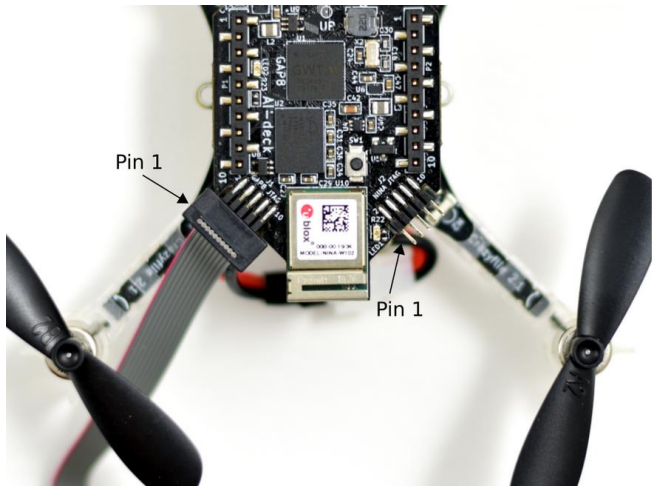
NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

Wi-Fi card

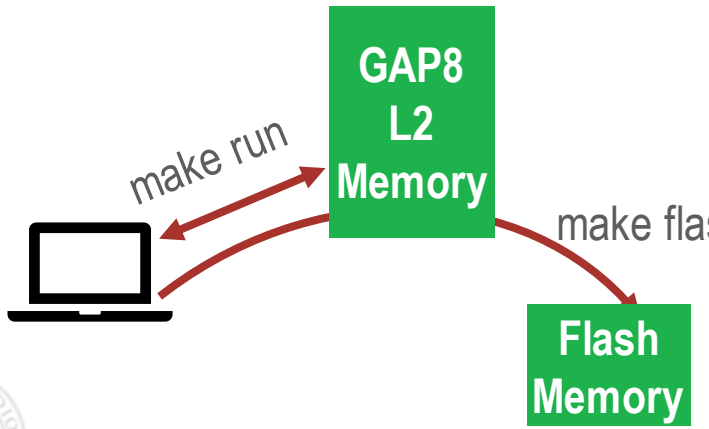
Hands-on 1: GAP8 programming

AI-Deck (GAP8)

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)
{
    /* 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);
}
```

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.

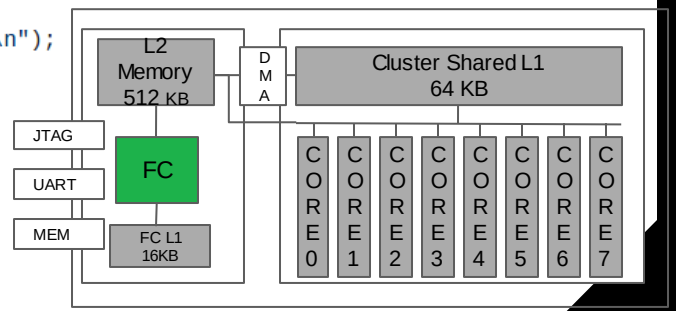
Return values

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.



Hands-on: Hello World!

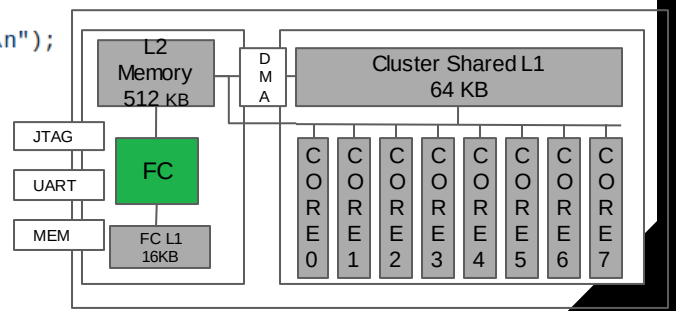
```c
/* PMSIS includes */
#include "pmsis.h"

/* 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};
```

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)

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



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};
```

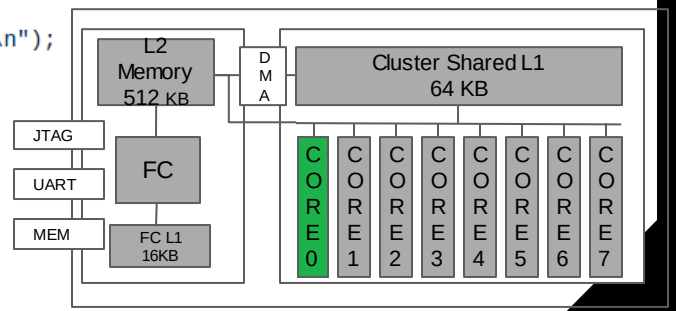
Init cluster config to default values
Set id manually
Point cluster device to your config
Open cluster (power up), blocking

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.

```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"

/* 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

```c
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);
}
```

We are only on core 0 of the cluster yet

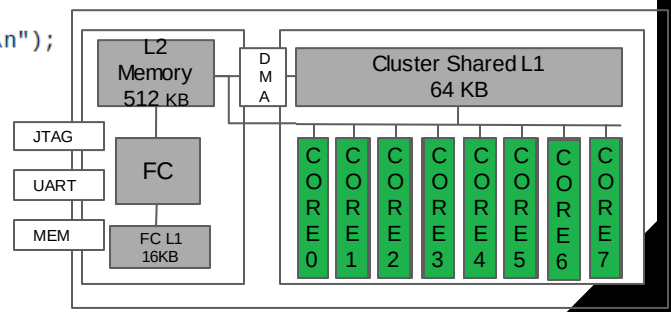
Fork to number of cluster cores available

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

```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);
}
```

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

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



Hands-on: Hello World!

Makefile

```
1  # User Test
2  #------------------------------------
3  APP              = test
4  # App sources
5  APP_SRCS         = helloworld.c
6  # App includes
7  APP_INC          =
8  # Compiler flags
9  APP_CFLAGS       =
10 # Linker flags
11 APP_LDFLAGS      =
12
13 # Custom linker
14 APP_LINK_SCRIPT  =
15
16 include $(RULES_DIR)/pmsis_rules.mk
```

Add sources here

Add directories to include (header files) here