CENTRALE NANTES

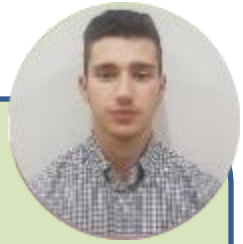
Review 5 M2 ERTS PROJECT

Validation of EDF
&
Implementation of ED-H on Xenomai

DEBUS ALEXY
RAYELLA NIRANJAN

25/01/2022

Our team



Debus Alexy

C dev.

Rayella Niranjan

C dev.

Table of contents

- 1 Context
- 2 Key question
- 3 Approach
- 4 Objectives
- 5 Planning
- 6 Progress
- 7 Project evolutions
- 8 Presentation of sprint 4

1. Context

Meeting with the professor (M.Queudet) on october the 15th :

The interest here is to put into applications a dynamic-priority energy aware scheduling strategie ED-H through the use of a Real-time kernel.

We will use Xenomai, I mean a dual kernel configurations : a Linux kernel (using Ubuntu distributions) supplementing by a RT co-kernel (Cobalt)

A Master student recently integrated EDF into Xenomai…

2. Key question

We need to have ED-H working without affecting integrity of other scheduling policies (fixed priority, EDF).

How could we manage to integrate/validate ED-H on Xenomai ?

3. Approach

1) Learn about Xenomai
2) Validate/Assert EDF implementation on Xenomai
3) Understand EDF implementation
4) Learn about ED-H
5) Implement ED-H In Xenomai
6) Validate/Assert ED-H implementation on Xenomai

The project will be split into sprints (1 sprint = 2 weeks)

4. Primary objectives

Primary objectives are established at the beginning of the project

1) Installation of modified Xenomai and validation for EDF integration
Metric : report for modifications added to have a functional installation + tests to validate xenomai-EDF (report + source code)
Estimated time : 10h

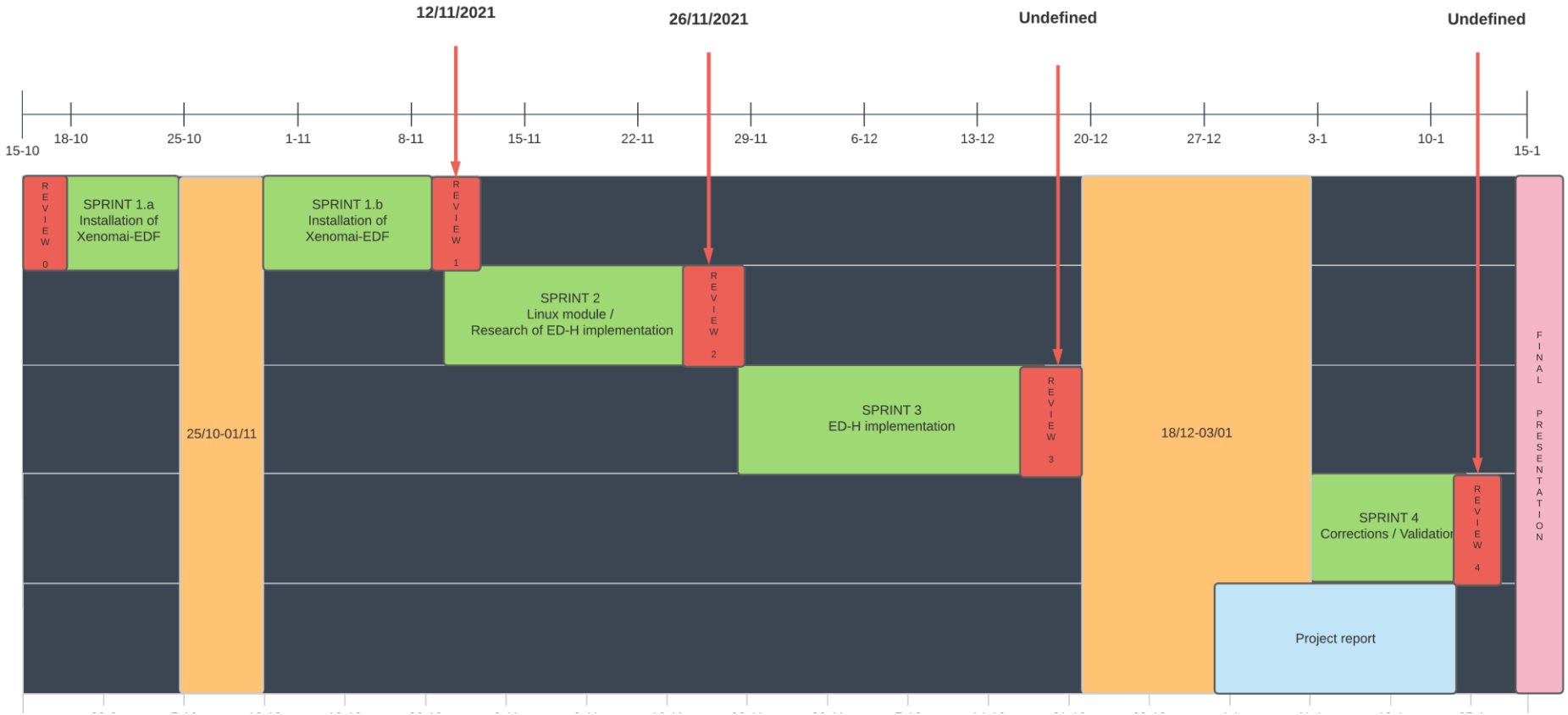
2) Development of a linux module to grab battery information and transfer to Cobalt
Metric : guideline for the creation/usee of the linux module + source code for the linux module
Estimated time : 25h

3) ED-H implementation and validation on Xenomai
Metric : guideline for the creation/use of the linux module + source code for the linux module + source code for ED-H implementation + source code for fake battery module
Estimated time : 50h

Remaining time : 15h (Redaction of the final report, PWP presentation, handle git repo.)

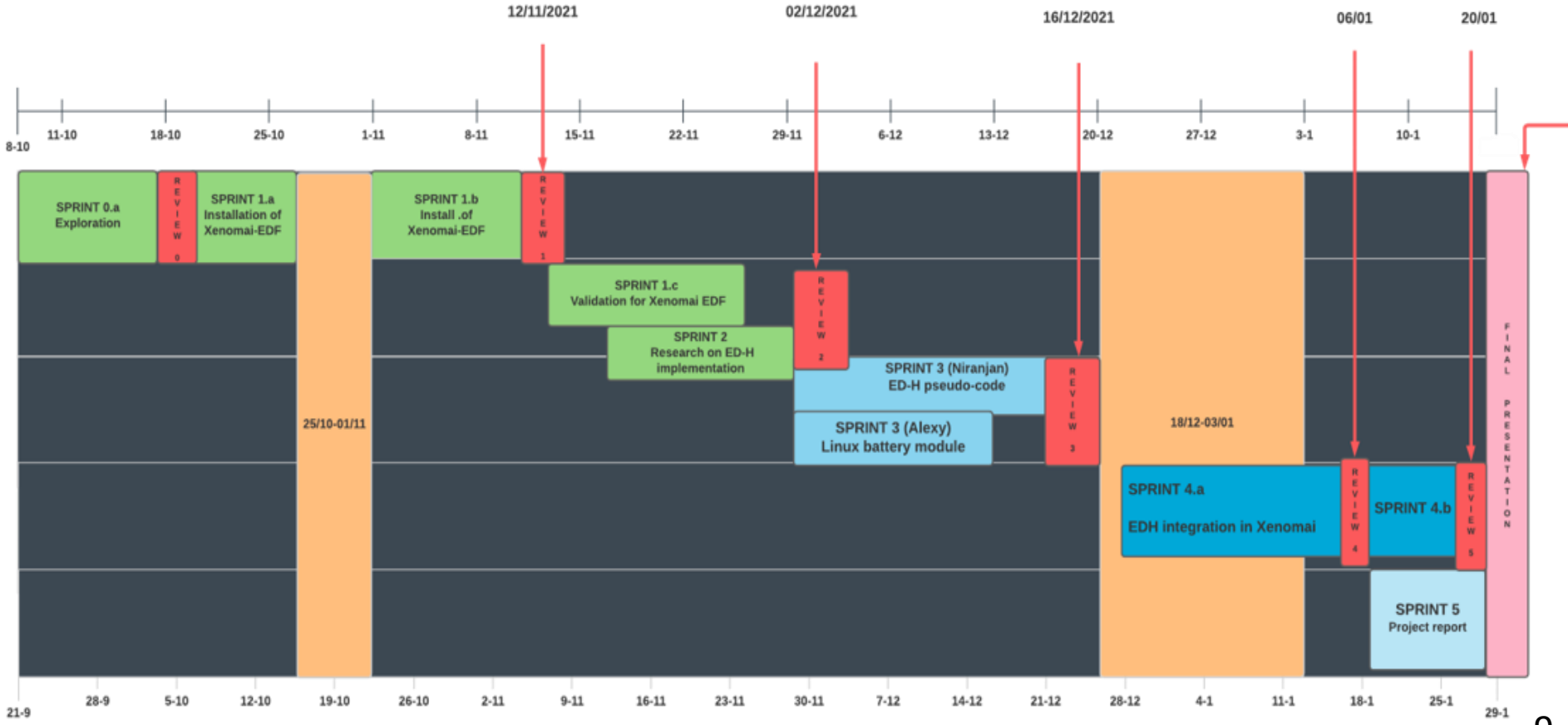
5. Planning

Provisional planning:



5. Planning

Effective planning:



6. Progress

0) Development of linux module to grab battery information
Results: Start implementation of a simple linux module (using kernel header file linux/power_supply)
2 weeks (08/10 - 17/10)
- In charge : Debus Alexy

1) Installation of Xenomai EDF
Results: Successfully installed Xenomai and validate EDF for no-premption test cases
3 weeks (18/10 - 11/11)
- *In charge : Debus Alexy & Rayella Niranjan*

2) Validation of Xenomai EDF
Results: Validate EDF for premption test cases
3 weeks (13/11 - 26/11)
- In charge : Debus Alexy

3) Linux battery module
Results: Working linux module to receive battery information
2 weeks (29/11 - 13/12)
- In charge : Debus Alexy

6. Progress

4) ED-H source code integration in Xenomai + battery module deployment

Results: source code for ED-H on Xenomai
(+ communication with battery module and/or fake battery module)

4 weeks ++ (17/12 - 25/01)
- In charge : Debus Alexy

7. Project evolution

Actual : (06/01 - 25/01) ED-H understanding

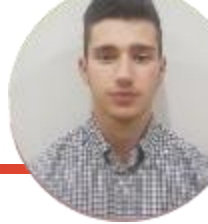
1. ED-H logic part.

2. End of project report

Future (25/01 – ...) :

End report redaction / Add all files/documentations in Github repo.

7. Project evolution



Actual : (06/01 - 25/01)

1. Modification for Alchemy API
2. Correction of batt.c source file for XDDP communication
3. Compilation (Kernel linux / Xenomai source code)
4. Report redaction
5. Add patch for EDF / ED-H

Future (25/01 – ...) :

Test XDDP comm. + end project documentation

8. Presentation of sprint 4.b

Modification for battery source file

```
int batt_init (void){

    //Alternative: Access to /dev/xeno_rtipc
    struct sockaddr_ipc saddr;
    int ret;
    size_t poolsz;

    /*
    * Get a datagram socket to bind to the RT endpoint. Each
    * endpoint is represented by a port number within the XDDP
    * protocol namespace.
    */

    ufd = __rtdm_dev_socket(AF_RTIPC, SOCK_DGRAM, IPCPROTO_XDDP);
    if (ufd < 0) {
            printk("XENO_WARNING __rtdm_dev_socket failed\n");
            return -1;
    }
    /*
     * Set a local 16k pool for the RT endpoint. Memory needed to
     * convey datagrams will be pulled from this pool, instead of
     * Xenomai's system pool.
     */
    poolsz = 16384; /* bytes */
    ret = rtdm_setsockopt(ufd, SOL_XDDP, XDDP_POOLSZ,
                    &poolsz, sizeof(poolsz));
    if (ret)
            printk("XENO_WARNING setsockopt failed\n");
    /*
     * Bind the socket to the port, to setup a proxy to channel
     * traffic to/from the Linux domain.
     *
     * saddr.sipc_port specifies the port number to use.
     */
    memset(&saddr, 0, sizeof(saddr));
    saddr.sipc_family = AF_RTIPC;
    saddr.sipc_port = XDDP_PORT;
    ret = rtdm_bind(ufd, (struct sockaddr *)&saddr, sizeof(saddr));

        if(ret < 0)
        {
          printk(XENO_INFO "bind error\n");
          return 1;
        }else{
          printk(XENO_INFO "bind OK on port %d\n",XDDP_PORT);
        }

        return 0;
}
```

```
Msg_battery battery_read_msg(void)
{
    char buf[128];
    struct timespec ts;
    fd_set readfds;
    Msg_battery battery_message= {};
    int ret = 0,ret_select = 0;

    #if 0
        fcntl(ufd, F_SETFL, O_NONBLOCK);

        //TODO Warning : blocking call here... Need to use select
        ts.tv_sec = 0;
        ts.tv_nsec = 0; /* 0 ms */

        ret_select = select(ufd + 1, &readfds, NULL, NULL, &ts);

        if (ret_select ==1)
          ret = rtdm_recvfrom(ufd, buf, MAX_BATT_READ_MSG_LENGTH, 0, NULL, 0);
    #endif

    ret = rtdm_read(ufd, buf, MAX_BATT_READ_MSG_LENGTH);
    if(ret > MAX_BATT_READ_MSG_LENGTH){
      battery_message.message_integrity = false;
    }
    if (ret >= 0 )
    {

        //Process message in struct
        sscanf((const char*)buf, "[%d,%d,%d,%d,%d]",
              (int*)&battery_message.capacity,
              (int*)&battery_message.chargenow,
              (int*)&battery_message.chargefull,
                (int*)&battery_message.battery_size,
          (int*)&battery_message.energy_production
    );

        if (battery_message.capacity >= 0 &&
          battery_message.chargenow >= 0 && battery_message.chargenow <= 100 &&
          battery_message.chargenow >= 0 && battery_message.chargenow <= 100 &&
    battery_message.battery_size >= 0 && battery_message.chargenow <= BATTERY_SIZE_MAX_VALUE &&
          battery_message.energy_production >= 0 && battery_message.energy_production <= EP_MAX_VALUE
           )
        {
          battery_message.message_integrity = true;
        }

    }else{
      //Handle error
      battery_message.message_integrity = false;
    }
    return battery_message;
}
```

8. Presentation of sprint 4.b

Modification for Alchemy API

```
CURRENT_DECL(int, rt_task_create(RT_TASK *task,
                                 const char *name,
                                 int stksize,
                                 int prio,
                                 int mode));

CURRENT_DECL(int, rt_task_create_dyna(RT_TASK *task,
                                      const char *name,
                                      int stksize,
                                      xnticks_t next_deadline,
                                      double WCET,
                                      double WCEC,
                                      dyna_policy policy,
                                      int mode));

CURRENT_DECL(int, rt_task_spawn(RT_TASK *task, const char *name,
                                int stksize, int prio, int mode,
                                void (*entry)(void *arg),
                                void *arg));

CURRENT_DECL(int, rt_task_spawn_dyna(RT_TASK *task, const char *name,
                                     int stksize, xnticks_t next_deadline,double WCET,double WCEC,dyna_policy policy, int mode,
                                     void (*entry)(void *arg),
                                     void *arg));
```

```
typedef enum{
  EDF = 0,
  EDH_ASAP, //As Soon As Possible
  EDH_ALAP  //As Late As Possible
}dyna_policy;
```

RTIME

8. Presentation of sprint 4.b

Function **xnsched_pick_next** in **kernel/cobalt/sched.c**

```
if (use_EDH){
  thread = &sched->rootcb;

  //ED-H Rule n°3:
  if(total_energy == 0 || slack_energy == 0){
    //Proc. IDLE
  }

  //ED-H Rule n°4:
  if(total_energy == my_msg_battery.capacity || slack_time == 0){
    thread = xnsched_rt_pick(sched);

    if (unlikely(thread == NULL))
      thread = &sched->rootcb;
  }

  //ED-H Rule n°5
  if(total_energy >0 && total_energy < my_msg_battery.capacity
     && slack_time > 0 && slack_energy > 0){
    //Either Busy or standby : actually will be busy
  }

}else{
  thread = xnsched_rt_pick(sched);
  if (unlikely(thread == NULL))
    thread = &sched->rootcb;
}

set_thread_running(sched, thread);

return thread;
```

- **Rule 1:** The priorities assigned by EDF are used to select the next ready job in $L_r(t_c)$.
- **Rule 2:** The processor is on standby during $[t_c, t_c + 1)$ if $L_r(t_c) = \emptyset$.
- **Rule 3:** The processor is on standby during $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and one of the following conditions is true:
  1. $E(t_c) \approx 0$
  2. $PSE_{\mathcal{T}}(t_c) \approx 0$
- **Rule 4:** The processor is busy during $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and one of the following conditions is true:
  1. $E(t_c) \approx C$
  2. $ST_{\mathcal{T}}(t_c) = 0$
- **Rule 5:** The processor can be either busy or on standby if $L_r(t_c) \neq \emptyset$, $0 < E(t_c) < C$, $ST_{\mathcal{T}}(t_c) > 0$ and $PSE_{\mathcal{T}}(t_c) > 0$.

9. Future

Function **xnsched_pick_next** in **kernel/cobalt/sched.c**

```
struct list_head *q = &sched->rt.runnable;
struct xnthread *b_thread;
union xnsched_policy_param param;

if (list_empty(q))
        goto no_battery;

list_for_each_entry(b_thread, q, rlink) {
    if (unlikely(b_thread->sched_class == &xnsched_class_dyna)){
        //Search through EDF thread
        b_thread->sched_class->sched_getparam(b_thread, &param);


        if (param.rt.policy == EDH_ASAP || param.rt.policy == EDH_ALAP){
          use_EDH = true;
          goto battery;
        }
    }
}

/*******************************************************/
battery: //Consider only one battery
  // Access battery data only if one of following task use EDH scheduling class
  if(use_EDH){
    my_msg_battery = battery_read_msg();

    if (my_msg_battery.message_integrity == true){
      printk(XENO_INFO
        "chargenow :%d\ncapacity:%d\n",my_msg_battery.chargenow,my_msg_battery.capacity);

      list_for_each_entry(b_thread, q, rlink) {
        if (unlikely(b_thread->sched_class == &xnsched_class_dyna)){

          /*TODO Compute for each thread:

          1) slack_time
          2) slack_energy

          using : */
          param.rt.WCET;
          param.rt.WCEC;

          my_msg_battery.chargenow;
          my_msg_battery.battery_size;
          my_msg_battery.energy_production;
        }
      }

    }
  }

no_battery:
```

Compute slack_time & slack_energy for each task

CENTRALE NANTES

Thanks for your attention.

Do you have any questions ?