



UNIVERSITÉ DE TECHNOLOGIE DE
COMPIÈGNE

MI11

Rapport des TPs linux embarqué

Clément BLANQUET et Rafik CHENNOUF

Juin 2017

Sommaire

1	Rapport TP 1 - Linux embarqué	4
	Exercice 1 : Prise en main de la carte DevKit8600	4
	1.1 Question 1.1	4
	1.2 Question 1.2	4
	1.3 Question 1.3	4
	1.4 Question 1.4	4
	1.5 Question 1.5	5
	Exercice 2	5
	2.1 Question 2.1	5
	2.2 Question 2.2	5
	2.3 Question 2.3	6
	Exercice 3	7
	3.1 Question 3.1	7
	3.2 Question 3.2	7
	3.3 Question 3.3	7
	3.4 Question 3.4	7
	3.5 Question 3.5	9
	Exercice 4 : Ajout de paquets	10
	4.1 Question 4.1	10
	4.2 Question 4.2	12
	4.3 Question 4.3	12
	4.4 Question 4.4	13
	Exercice 5 : Compilation manuelle du noyau	13
	5.1 Question 5.1	13
	5.2 Question 5.2	13
	5.3 Question 5.3	13
	5.4 Question 5.4	14
	5.5 Question 5.5	16
	5.6 Question 5.6	16
	5.7 Question 5.7	17
2	TP1 Xenomai	22
	Exercice 1 : tâches	22
	1.1 Question 1.1	22

1.2	Question 1.2	22
1.3	Question 1.3	23
1.4	Question 1.4	24
1.5	Question 1.5	24
Exercice 2 :	Synchronisation	25
2.1	Question 2.1	25
2.2	Question 2.2	27
2.3	Question 2.3	27
2.4	Question 2.4	27
2.5	Question 2.5	27
2.6	Question 2.6	29
2.7	Question 2.7	32
Exercice 3 :	Latence	33
3.1	Question 3.1	33
3.2	Question 3.2	34
3.3	Question 3.3	36
3	TP2 Xenomai	37
Exercice :	Pathfinder	37
1.1	Question 1	37
1.2	Question 2	37
1.3	Question 3	38
1.4	Question 4	38
1.5	Question 5	39
1.6	Question 6	41
1.7	Question 7	41
1.8	Question 8	42
1.9	Question 9	43
1.10	Question 10	43
A	Messages de sortie du terminal entre l'allumage de la cible et le prompt de login	51

Table des figures

1.1	Dossier image	6
1.2	Mise en évidence du port série dans /proc/devices	8
1.3	Mise en évidence du port série dans /dev	9
1.4	Dossiers /bin et /sbin	9
1.5	Dossier /usr/bin	10
1.6	Dossier /usr/sbin	10
1.7	Dossier <i>ipk</i>	11
1.8	Dossier <i>ipk/devkit800</i>	18
1.9	Dossier <i>ipk/armv7a-vfp-neon</i>	18
1.10	Dossier <i>/usr/lib</i>	19
1.11	Configurations par défaut de <i>devkit8600</i>	19
1.12	Activation du driver pour les LEDs	20
1.13	Logs de démarrage	21
1.14	Logs du kernel	21
2.1	Statistiques Xenomai avec un programme non temps réel	22
2.2	Statistiques Xenomai avec une tâche temps réel	24
2.3	Statistiques Xenomai avec une tâche temps réel et la fonction <code>rt_task_sleep</code>	24
2.4	Statistiques Xenomai avec une tâche temps réel et les fonctions <code>rt_task_sleep</code> et <code>rt_printf</code>	25
2.5	Fichier de statistiques de Xenomai	32
2.6	Fichier du scheduler de Xenomai	32
2.7	Résultat du programme <i>Latence</i> sans stress	36
2.8	Résultat du programme <i>Latence</i> avec stress	36
3.1	Résultat avec <code>ORDO_BUS</code>	39
3.2	Enchaînement avec meilleur cas pour METEO (40ms)	40
3.3	Enchaînement avec pire cas pour METEO (60ms)	40
3.4	Reset avec le cas extrême pour METEO (60ms)	41
3.5	Chronogramme illustrant l'inversion de priorité	42
3.6	Résultat avec un mutex	43

Rapport TP 1 - Linux embarqué

Exercice 1 : Prise en main de la carte DevKit8600

1.1 Question 1.1

La DevKit8600 est basée sur les processeurs AM3359 de Texas Instrument. Elle possède un ARM Cortex-A8 cadencé à 720 MHz avec un boot ROM On-Chip de 176KB. Elle possède un certain nombre d'interfaces comme un port LAN, une entrée/sortie audio, des USB, une interface JTAG... etc.

1.2 Question 1.2

On peut stocker le noyau via TFTP (Trivial File Transfer Protocol) et le système de fichiers via NFS (Network File System).

1.3 Question 1.3

Nous avons utilisé une liaison série pour nous connecter avec l'application cutecom.

Voici les paramètres que l'on a utilisés :

1. Bits per second : 115200
2. Data bits : 8
3. Parity : None
4. Stop bits : 1
5. Flow Control : None

1.4 Question 1.4

On constate qu'il y a une erreur au démarrage.

```
1 TFTP error: 'File not found' (1)
3 ERROR: can't get kernel image!
```

La cible ne peut donc pas démarrer. Le fichier manquant est l'image du kernel, appelé **uImage**. Il faudrait la placer dans le dossier `/tftpboot` sur notre PC, puisque la cible nous indique au démarrage qu'elle va chercher l'image à cet endroit.

1.5 Question 1.5

Même si l'image était présente, la cible ne pourrait pas démarrer car il manque un rootfs (système de fichier racine).

Exercice 2

2.1 Question 2.1

Le dossier `/home/mi11/poky/build/conf` contient les fichiers de configurations de poky qui permettent donc de configurer l'image selon nos besoins :

- `bblayers.conf`
- `local.conf`
- `sanity_info`
- `templateconf.cfg`

Le dossier `/home/mi11/devkit8600/meta-devkit8600` contient les fichiers spécifiques à notre cible qui vont permettre de construire une image qui lui est adaptée.

2.2 Question 2.2

Nous avons ajouté une ligne dans le fichier `bblayers.conf` :

```
1 BBLAYERS ?= " \
  /home/mi11/poky-dizzy -12.0.3/meta \
3 /home/mi11/poky-dizzy -12.0.3/meta-yocto \
  /home/mi11/poky-dizzy -12.0.3/meta-yocto-bsp \
5 /home/mi11/devkit8600/meta-devkit8600 \ // CELLE LA
  "
```

De plus, à la ligne 36 du fichier `local.conf`, nous avons inscrit :

```
MACHINE ??= "devkit8600"
```

Le nom `"devkit8600"` est en fait le nom du fichier de configuration du même nom situé ici : `/home/mi11/devkit8600/meta-devkit8600/conf/machine`, ce qui fait donc le lien entre la génération de l'image et les paramètres de la cible.

```

mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/images $ ll devkit8600/
total 79736
drwxr-xr-x 2 mi11 mi11 4096 avril 14 15:13 ./
drwxr-xr-x 3 mi11 mi11 4096 avril 14 15:07 ../
-rw-r--r-- 1 mi11 mi11 5442 avril 14 15:12 core-image-base-devkit8600-20170414130644.rootfs.manifest
-rw-r--r-- 1 mi11 mi11 18255233 avril 14 15:13 core-image-base-devkit8600-20170414130644.rootfs.tar.bz2
-rw-r--r-- 1 mi11 mi11 30670848 avril 14 15:13 core-image-base-devkit8600-20170414130644.rootfs.ubi
-rw-r--r-- 1 mi11 mi11 29458432 avril 14 15:13 core-image-base-devkit8600-20170414130644.rootfs.ubifs
lrwxrwxrwx 1 mi11 mi11 57 avril 14 15:13 core-image-base-devkit8600.manifest -> core-image-base-devkit8600-20170414130644.rootfs.manifest
lrwxrwxrwx 1 mi11 mi11 56 avril 14 15:13 core-image-base-devkit8600.tar.bz2 -> core-image-base-devkit8600-20170414130644.rootfs.tar.bz2
lrwxrwxrwx 1 mi11 mi11 52 avril 14 15:13 core-image-base-devkit8600.ubi -> core-image-base-devkit8600-20170414130644.rootfs.ubi
lrwxrwxrwx 1 mi11 mi11 54 avril 14 15:13 core-image-base-devkit8600.ubifs -> core-image-base-devkit8600-20170414130644.rootfs.ubifs
-rw-r--r-- 1 mi11 mi11 21617 avril 14 15:07 modules--3.1.0-r0-devkit8600-20170410160636.tgz
lrwxrwxrwx 1 mi11 mi11 47 avril 14 15:07 modules-devkit8600.tgz -> modules--3.1.0-r0-devkit8600-20170410160636.tgz
-rw-r--r-- 1 mi11 mi11 294 avril 14 15:11 README - DO NOT DELETE FILES IN THIS DIRECTORY.txt
-rw-r--r-- 1 mi11 mi11 192 avril 14 15:13 ubinize.cfg
lrwxrwxrwx 1 mi11 mi11 46 avril 14 15:07 uImage -> uImage--3.1.0-r0-devkit8600-20170410160636.bin
-rw-r--r-- 1 mi11 mi11 3215152 avril 14 15:07 uImage--3.1.0-r0-devkit8600-20170410160636.bin
lrwxrwxrwx 1 mi11 mi11 46 avril 14 15:07 uImage-devkit8600.bin -> uImage--3.1.0-r0-devkit8600-20170410160636.bin
mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/images $

```

FIGURE 1.1 – Dossier image

2.3 Question 2.3

Voici ce qu'on obtient dans le dossier `/home/mi11/poky/build/tmp/deploy/images` après la compilation :

On obtient notre image kernel `uImage` qui est en fait un lien symbolique vers le fichier `uImage-3.1.0-r0-devkit8600-20170410160636.bin`.

On voit également le système de fichier nommé `rootfs` qui est compressé dans une archive bzip2.

Si on compare la taille des fichiers qui viennent d'être générés avec ceux de notre VM sous Linux Mint :

- CIBLE :
 - Taille image : 3,2 MB
 - Taille rootfs : 1,8 MB
- HOTE :
 - Taille image : 6,6 MB (on le voit dans `/boot`)
 - Taille système de fichiers : environ 7 GB

On constate une bonne différence entre les deux images car l'image de la cible n'est faite que pour cette cible là, elle gère moins de choses et propose moins de fonctionnalités que celle de notre VM. Pour le système de fichiers, on constate une énorme différence car beaucoup plus de programmes sont installés sur notre VM par rapport à la cible sur laquelle rien ou presque n'est installé. La cible dispose en fait de la configuration minimale.

Exercice 3

3.1 Question 3.1

Comme nous l'avions indiqué dans le premier exercice, nous avons copié uImage dans le dossier `/tftpboot`, comme c'est indiqué dans le fichier de configuration `/etc/xinetd.d/tftp`. Le rootfs, lui, est copié dans le dossier `/tftpboot/rootfs`

3.2 Question 3.2

Messages de sortie du terminal entre l'allumage de la cible et le prompt de login : voir annexe A.1.

Analyse :

Du début jusqu'à "Starting kernel ...", la cible charge l'image via tftp et la lance. Une fois cela fait, Linux se lance et donne tout un tas d'informations sur le système (le nom de la machine, le type de CPU, la mémoire...). Viennent ensuite plusieurs opérations de calibrages et de tests, puis d'initialisations (comme le Bluetooth ou les interfaces réseaux). Enfin, le système se lance.

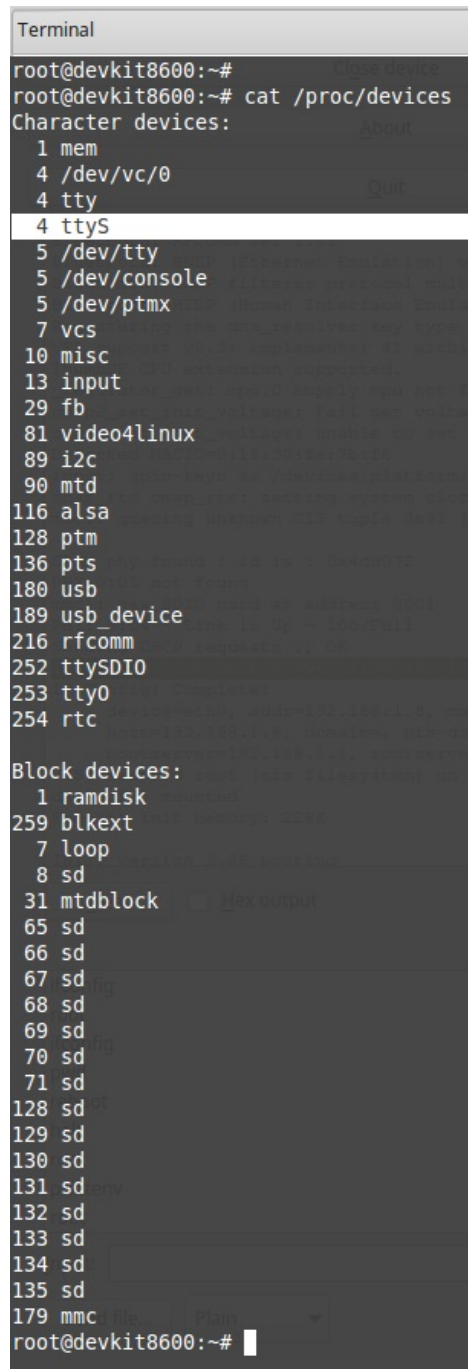
3.3 Question 3.3

L'IP de la cible est 192.168.1.6. On peut voir cette adresse sur les messages de sortie du boot : *"IP-Config : Got DHCP answer from 192.168.1.1, my address is 192.168.1.6 (sortie du boot)"*

3.4 Question 3.4

`/proc/devices` contient les périphériques (dans la première section) ainsi que les stockages (dans la deuxième section).

Le numéro en début de ligne dans le fichier `/proc/devices` correspond au numéro mineur, qui indique si les périphériques sont gérés par le même driver.



```
Terminal
root@devkit8600:~#
root@devkit8600:~# cat /proc/devices
Character devices:
1 mem
4 /dev/vc/0
4 tty
4 ttys
5 /dev/tty
5 /dev/console
5 /dev/ptmx
7 vcs
10 misc
13 input
29 fb
81 video4linux
89 i2c
90 mtd
116 alsa
128 ptm
136 pts
180 usb
189 usb_device
216 rfcomm
252 ttyS0
253 tty0
254 rtc
Block devices:
1 ramdisk
259 blkext
7 loop
8 sd
31 mtdblock
65 sd
66 sd
67 sd
68 sd
69 sd
70 sd
71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
179 mmcblk0
root@devkit8600:~#
```

FIGURE 1.2 – Mise en évidence du port série dans `/proc/devices`

```

root@devkit8600:~# ls /dev/ttyS
ttyS0 ttyS1 ttyS2 ttyS3
root@devkit8600:~# ls /dev/ttyS

```

FIGURE 1.3 – Mise en évidence du port série dans /dev

3.5 Question 3.5

Voici une rapide description des dossiers :

```

root@devkit8600:~# ls /bin/
ash          df            ln            netstat      stty
bash         dmesg         login         pidof        su
busybox      dnsdomainname login.shadow  pidof.sysvinit su.shadow
busybox.nosuid dumpkmap      ls           ping         sync
busybox.suid echo          lsmod        ping6        tar
cat          egrep         lsmod.kmod   ps           touch
chattr       false         mkdir        pwd          true
chgrp        fgrep         mknod        rm           umount
chmod        fgrep         mktemp       rmdir        uname
chown        gunzip        more         run-parts   usleep
cp           gzip          mount        sed          vi
cpio         hostname     mountpoint   sh           watch
date         kill          mountpoint.sysvinit sleep        zcat
dd           kmod         mv           stat

root@devkit8600:~# ls /sbin/
bootlogd     ifconfig      iwpriv        modinfo.kmod  route         sysctl
depmod        ifdown        iwspy         modprobe      runlevel      syslogd
depmod.kmod   ifup          killall5      modprobe.kmod runlevel.sysvinit telinit
fdisk         init          klogd         nologin       setconsole    udhcpc
fsck          init.sysvinit ldconfig      pivot_root    shutdown      vigr
fstab-decode  insmod        loadkmap      poweroff      shutdown.sysvinit vigr.shadow
fstrim        insmod.kmod  logread       poweroff.sysvinit start-stop-daemon vipw
getty         ip            losetup       reboot         sulogin       vipw.shadow
halt          iwconfig     lsmod         reboot.sysvinit swapon        switch_root
halt.sysvinit iwgetid      mkswap        rmmod          swapon
hwclock       iwlist       modinfo       rmmmod.kmod   switch_root

```

FIGURE 1.4 – Dossiers /bin et /sbin

- /bin : contient les binaires utilisables dès le boot, avant que la partition /usr soit montée. On y voit les commandes de base de type ls ou cat.
- /sbin : même chose que /bin, mais commandes nécessitant des droits particuliers (superuser d'où le s)

```

root@devkit8600:~# ls /usr/bin/
[[
ar
awk
basename
bashbug
bunzip2
bzip2
chage
chfn
chfn.shadow
chsh
chsh.shadow
chvt
ciptool
clear
cmp
cut
dbclient
dbus-clean-up-sockets
dbus-daemon
dbus-launch
dbus-monitor
dbus-run-session
dbus-send
dbus-uuidgen
dc
deallocvt
dfutool
diff
dirname
dlist_test
du
dumpleases
env
evtest
expiry
expr
faillog
find
flock
free
fuser
gatttool
get device
get driver
get module
gpsswd
groups
groups.shadow
hcitool
head
hexdump
id
killall
l2ping
l2test
last
last.sysvinit
lastb
lastlog
less
logger
logname
lsusb
lsusb.py
md5sum
msg
msg.sysvinit
microcom
mkfifo
mpicalc
nc
nettle-hash
nettle-lfib-stream
newgidmap
newgrp
newgrp.shadow
newuidmap
nfctool
nohup
nslookup
od
openssl
openvt
opkg
opkg-cl
opkg-key
passwd
passwd.shadow
patch
pkcs11-conv
printf
psplash
psplash-default
psplash-write
python
python-config
python2
python2-config
python2.7
python2.7-config
rctest
readlink
realpath
renice
reset
rfcomm
scp
sdptool
seq
sexp-conv
sg
sha3sum
sort
ssh
stress
strings
systool
tail
tee
telnet
test
tftp
time
top
tr
traceroute
tty
udevadm
uniq
unzip
update-alternatives
uptime
usb-devices
usbhid-dump
users
utmpdump
utmpdump.sysvinit
vlock
wall
wall.sysvinit
wc
wget
which
who
whoami
wpa_passphrase
xargs
yes

```

FIGURE 1.5 – Dossier /usr/bin

- /usr/bin : contient les binaires généraux du système (comme les applications que l'utilisateur installe)

```

root@devkit8600:~# ls /usr/sbin/
addgroup
adduser
avahi-daemon
bccmd
bluetoothd
chpasswd
chpasswd.shadow
chroot
delgroup
deluser
dropbear
dropbearconvert
dropbearkey
dropbearmulti
fbset
genl-ctrl-list
groupadd
groupdel
groupmems
groupmod
grpck
grpconv
grpunconv
hciattach
hciemu
loadfont
logoutd
newusers
nl-class-add
nl-class-delete
nl-class-list
nl-classid-lookup
nl-cls-add
nl-cls-delete
nl-cls-list
nl-link-list
nl-pktloc-lookup
nl-qdisc-add
nl-qdisc-delete
nl-qdisc-list
ofonod
pwck
pwconv
pwunconv
rdate
rfkill
rpcbind
rpcinfo
run-postinsts
udhcpd
update-rc.d
update-usbids.sh
useradd
userdel
usermod
wpa_cli
wpa_supplicant

```

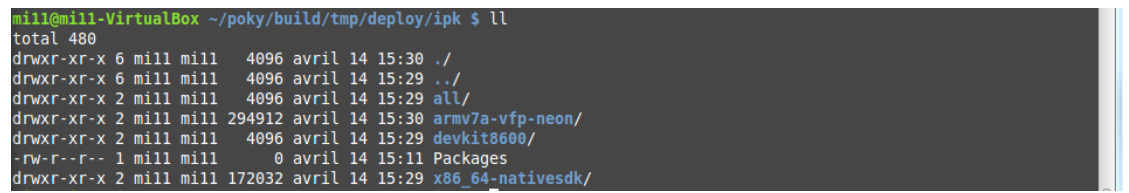
FIGURE 1.6 – Dossier /usr/sbin

- /usr/sbin : même chose que /usr/bin, mais nécessitant des droits particuliers (superuser d'où le s)

Exercice 4 : Ajout de paquets

4.1 Question 4.1

Le dossier `/home/mi11/poky/build/tmp/deploy/ipk` est organisé par architectures (x86/x64, devkit8600, armv7a, etc.).



```
mi11@mi11-VirtualBox ~/poky/build/tmp/deploy/ipk $ ll
total 480
drwxr-xr-x 6 mi11 mi11  4096 avril 14 15:30 ./
drwxr-xr-x 6 mi11 mi11  4096 avril 14 15:29 ../
drwxr-xr-x 2 mi11 mi11  4096 avril 14 15:29 all/
drwxr-xr-x 2 mi11 mi11 294912 avril 14 15:30 armv7a-vfp-neon/
drwxr-xr-x 2 mi11 mi11  4096 avril 14 15:29 devkit8600/
-rw-r--r-- 1 mi11 mi11    0 avril 14 15:11 Packages
drwxr-xr-x 2 mi11 mi11 172032 avril 14 15:29 x86_64-nativesdk/
```

FIGURE 1.7 – Dossier *ipk*

Les sous-dossiers contiennent des paquets d’extension *.ipk*. Le noyau se trouve dans le sous-dossier *devkit8600* comme nous pouvons le voir sur la copie d’écran ci-dessus. Quant au paquet *libxml2*, il se trouve dans le sous-dossier *armv7a-vfp-neon*.

4.2 Question 4.2

Les différents paquets relatifs à *libxml2* sont visibles sur la copie d'écran suivante.

Nous avons ensuite copié le fichier

```
libxml2_2.9.1-r0_armv7a-vfp-neon.ipk
```

dans le système de fichiers de la cible et nous avons installé le paquet avec la commande suivante :

```
opkg install libxml2_2.9.1-r0_armv7a-vfp-neon.ipk
```

4.3 Question 4.3

La première méthode pour copier le fichier dans le système de fichiers de la cible consiste à utiliser directement la commande *cp* comme suit :

```
sudo cp libxml2_2.9.1-r0_armv7a-vfp-neon.ipk /tftpboot/rootfs/home/root/
```

La deuxième méthode consiste à copier le fichier à distance via la commande *scp* qui se base sur *ssh* comme suit :

```
scp libxml2_2.9.1-r0_armv7a-vfp-neon.ipk root@192.168.1.6:/home/root/
```

4.4 Question 4.4

Les fichiers de *libxml2* installés par le gestionnaire de paquets se trouvent dans le dossier */usr/lib* de notre cible.

Ce dossier contient des fichiers *shared object* (.so) qui sont des librairies/bibliothèques partagées dynamiques. Les .so sont des bibliothèques qui se chargent en mémoire au moment de l'exécution d'un programme qui les utilisent. Si plusieurs programmes les utilisant sont lancés en même temps, une seule instance de la bibliothèque dynamique réside en mémoire.

Exercice 5 : Compilation manuelle du noyau

5.1 Question 5.1

La page 72 de la documentation nous dit qu'il est possible d'allumer et d'éteindre les LEDs via les commandes suivantes :

```
root@DevKit8600:~# echo 1 > /sys/class/leds/user_led/brightness
root@DevKit8600:~# echo 0 > /sys/class/leds/user_led/brightness
```

Comme le dossier

user_led

n'existe pas ce n'est pas opérationnel.

5.2 Question 5.2

Nous avons ensuite compiler manuellement le noyau via la chaîne de compilation croisée en exécutant le script suivant :

```
source /opt/poky/1.7.3/environment-setup-armv7a-vfp-neon-poky-linux-gnueabi
unset LDFLAGS
```

Le fichier ci-dessus sert à mettre en place l'environnement de compilation de manière à avoir les bons préfixes pour une compilation croisée. Le préfixe la chaîne de compilation croisée est *arm-poky-linux-gnueabi-*.

5.3 Question 5.3

Pour obtenir la liste des configurations par défaut du noyau pour une architecture ARM, il faut utiliser la commande *make ARCH=arm help*. Les configurations

par défaut possibles pour la carte *devkit8600* sont visibles sur la copie d'écran ci-dessous :

Nous avons retenu uniquement la première configuration et nous l'avons mise en place via la commande suivante :

```
make devkit8600_defconfig
```

5.4 Question 5.4

Ensuite, nous avons lancé la personnalisation du noyau via la commande suivante : *make ARCH=arm menuconfig*. Le but étant d'ajouter un driver pour les LEDs connectées par GPIO. Pour cela, nous avons activé l'option *LED Support for GPIO connected LEDs* comme on peut le voir sur la copie d'écran suivante.

Ce driver peut être activé via deux modes, *modularizes features* ou *built-in*. Le premier mode est utilisé si l'on souhaite ajouter un driver en tant que module qui sera chargé en mémoire uniquement en cas de besoin et déchargé lorsque le kernel n'en a pas plus besoin. Ceci est utile lorsque l'on souhaite avoir un kernel pas très lourd. Quant au deuxième mode, le driver sera directement intégré au kernel et il sera disponible tout le temps. Le kernel sera donc plus gros, plus lent et utilisera plus de mémoire. Dans notre cas, nous avons utilisé le mode *built-in*.

Nous avons ensuite compiler notre noyau avec la commande suivante : *make ARCH=arm uImage -j2*.

5.5 Question 5.5

Le résultat de la compilation se trouve dans *arch/arm/boot/uImage*.

Nous avons ensuite copier le fichier uImage dans /tftpboot pour qu'il soit utilisé par la cible puis nous l'avons démarré.

5.6 Question 5.6

Pour vérifier dans les logs de démarrage que le noyau utilisé est bien celui qui vient d'être compilé, il suffit de lire la date de compilation :

Nous pouvons lire sur la copie d'écran précédente la ligne suivante :

```
Linux version 3.1.0 (mi11@mi11-VirtualBox) (gcc version 4.9.1 (GCC) ) #1 Fri Apr 14
```

La date de compilation du noyau correspond bien à celui qui vient d'être compilé.

5.7 Question 5.7

Pour vérifier dans les logs de démarrage que la fonctionnalité ajoutée est bien présente, on peut utiliser la commande *dmesg* pour afficher les messages du kernel :

Le driver a bien été ajouté et le dossier

`used_led`

a été créé.

```

mill@m11-VirtualBox ~/poky/build/tmp/deploy/ipk/devkit8600 $ ll
total 36764
drwxr-xr-x 2 mill mill 4096 avril 14 15:29 ./
drwxr-xr-x 6 mill mill 4096 avril 14 15:30 ../
-rw-r--r-- 1 mill mill 4046 avril 14 15:08 base-files_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 840 avril 14 15:08 base-files-dbg_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 872 avril 14 15:08 base-files-dev_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 922 avril 14 15:08 base-files-doc_3.0.14-r89_devkit8600.ipk
-rw-r--r-- 1 mill mill 1072 avril 14 15:07 depmodwrapper-cross_1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 704 avril 14 15:07 depmodwrapper-cross-dbg_1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 736 avril 14 15:07 depmodwrapper-cross-dev_1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 2722 avril 14 15:07 kernel-3.1.0_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 678 avril 14 15:07 kernel_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 30265112 avril 14 15:07 kernel-dev_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 3195142 avril 14 15:07 kernel-image-3.1.0_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 702 avril 14 15:07 kernel-modules_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 1922 avril 14 15:07 kernel-module-scsi-wait-scan_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 19494 avril 14 15:07 kernel-module-usbserial_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 3939110 avril 14 15:07 kernel-vmlinux_3.1.0-r0_devkit8600.ipk
-rw-r--r-- 1 mill mill 826 avril 14 15:08 opkg-config-base_1.0-r1_devkit8600.ipk
-rw-r--r-- 1 mill mill 682 avril 14 15:08 opkg-config-base-dbg_1.0-r1_devkit8600.ipk
-rw-r--r-- 1 mill mill 714 avril 14 15:08 opkg-config-base-dev_1.0-r1_devkit8600.ipk
-rw-r--r-- 1 mill mill 776 avril 14 15:07 packagegroup-base_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 692 avril 14 15:07 packagegroup-base-3g_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 726 avril 14 15:07 packagegroup-base-bluetooth_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 790 avril 14 15:07 packagegroup-base-dbg_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 822 avril 14 15:07 packagegroup-base-dev_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 660 avril 14 15:07 packagegroup-base-extended_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 668 avril 14 15:07 packagegroup-base-ipv6_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 688 avril 14 15:07 packagegroup-base-nfc_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 690 avril 14 15:07 packagegroup-base-nfs_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 712 avril 14 15:07 packagegroup-base-usb-gadget_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 742 avril 14 15:07 packagegroup-base-usb-host_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 712 avril 14 15:07 packagegroup-base-vfat_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 772 avril 14 15:07 packagegroup-base-wifi_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 680 avril 14 15:07 packagegroup-base-zeroconf_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 734 avril 14 15:07 packagegroup-core-boot_1.0-r17_devkit8600.ipk
-rw-r--r-- 1 mill mill 804 avril 14 15:07 packagegroup-core-boot-dbg_1.0-r17_devkit8600.ipk
-rw-r--r-- 1 mill mill 832 avril 14 15:07 packagegroup-core-boot-dev_1.0-r17_devkit8600.ipk
-rw-r--r-- 1 mill mill 684 avril 14 15:07 packagegroup-distro-base_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 722 avril 14 15:07 packagegroup-machine-base_1.0-r83_devkit8600.ipk
-rw-r--r-- 1 mill mill 26783 avril 14 15:29 Packages
-rw-r--r-- 1 mill mill 4259 avril 14 15:29 Packages.gz
-rw-r--r-- 1 mill mill 2558 avril 14 15:29 Packages.stamps
-rw-r--r-- 1 mill mill 868 avril 14 15:07 poky-feed-config-opkg_1.0-r2_devkit8600.ipk
-rw-r--r-- 1 mill mill 684 avril 14 15:07 poky-feed-config-opkg-dbg_1.0-r2_devkit8600.ipk
-rw-r--r-- 1 mill mill 714 avril 14 15:07 poky-feed-config-opkg-dev_1.0-r2_devkit8600.ipk
-rw-r--r-- 1 mill mill 1450 avril 14 15:09 shadow-securetty_4.2.1-r3_devkit8600.ipk
-rw-r--r-- 1 mill mill 690 avril 14 15:09 shadow-securetty-dbg_4.2.1-r3_devkit8600.ipk

```

FIGURE 1.8 – Dossier *ipk/devkit800*

```

mill@m11-VirtualBox ~/poky/build/tmp/deploy/ipk/armv7a-vfp-neon $ ls | grep libxml2
libxml2_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-dbg_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-dev_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-doc_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-ptest_2.9.1-r0_armv7a-vfp-neon.ipk
libxml2-staticdev_2.9.1-r0_armv7a-vfp-neon.ipk
mill@m11-VirtualBox ~/poky/build/tmp/deploy/ipk/armv7a-vfp-neon $

```

FIGURE 1.9 – Dossier *ipk/armv7a-vfp-neon*

```

root@devkit8600:~# ls /usr/lib/
dbus/                               libgio-2.0.so.0.4000.0             libnl-genl-3.so.200
gio/                               libglib-2.0.so.0                   libnl-genl-3.so.200.20.0
libX11.so.6                        libglib-2.0.so.0.4000.0           libnl-nf-3.so.200
libX11.so.6.3.0                    libgmodule-2.0.so.0               libnl-nf-3.so.200.20.0
libXau.so.6                        libgmodule-2.0.so.0.4000.0        libnl-route-3.so.200
libXau.so.6.0.0                    libgmp.so.10                       libnl-route-3.so.200.20.0
libXdmp.so.6                       libgmp.so.10.2.0                  libopkg.so.1
libXdmp.so.6.0.0                  libgnutls.so.28                   libopkg.so.1.0.0
libavahi-common.so.3               libgnutls.so.28.38.0              libpyglib-2.0-python.so.0
libavahi-common.so.3.5.3           libgobject-2.0.so.0               libpyglib-2.0-python.so.0.0.0
libavahi-core.so.7                 libgobject-2.0.so.0.4000.0        libpython2.7.so.1.0
libavahi-core.so.7.0.2             libgpg-error.so.0                 libreadline.so.6
libbluetooth.so.3                  libgpg-error.so.0.10.0            libreadline.so.6.3
libbluetooth.so.3.13.0             libgthread-2.0.so.0               libssl.so.1.0.0
libdaemon.so.0                     libgthread-2.0.so.0.4000.0        libtirpc.so.1
libdaemon.so.0.5.0                 libhistory.so.6                    libtirpc.so.1.0.10
libdbus-1.so.3                      libhistory.so.6.3                  libxcb.so.1
libdbus-1.so.3.8.4                  libhogweed.so.2                     libxcb.so.1.1.0
libdbus-glib-1.so.2                 libhogweed.so.2.5                  libxml2.so.2
libdbus-glib-1.so.2.2.2             libkmod.so.2                        libxml2.so.2.9.1
libexpat.so.1                       libkmod.so.2.2.8                    locale/
libexpat.so.1.6.0                  libnettle.so.4                       near/
libffi.so.6                         libnettle.so.4.7                     opkg/
libffi.so.6.0.2                     libnl-3.so.200                       python2.7/
libgcrypt.so.20                     libnl-3.so.200.20.0                 ssl/
libgcrypt.so.20.0.1                 libnl-cli-3.so.200
libgio-2.0.so.0                     libnl-cli-3.so.200.20.0

```

FIGURE 1.10 – Dossier */usr/lib*

```

mill@mill-VirtualBox ~/devkit8600/linux-3.1.0-psp04.06.00.03.sdk $ make ARCH=arm help | grep devkit
Makefile:657: "WARNING: Appending $KFLAGS (--sysroot=/opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi) f
rom environment to kernel $CFLAGS"
      (default: /home/mill/devkit8600/linux-3.1.0-psp04.06.00.03.sdk/usr)
devkit8600_defconfig - Build for devkit8600
devkit8600_tisdk_defconfig - Build for devkit8600_tisdk
mill@mill-VirtualBox ~/devkit8600/linux-3.1.0-psp04.06.00.03.sdk $

```

FIGURE 1.11 – Configurations par défaut de *devkit8600*

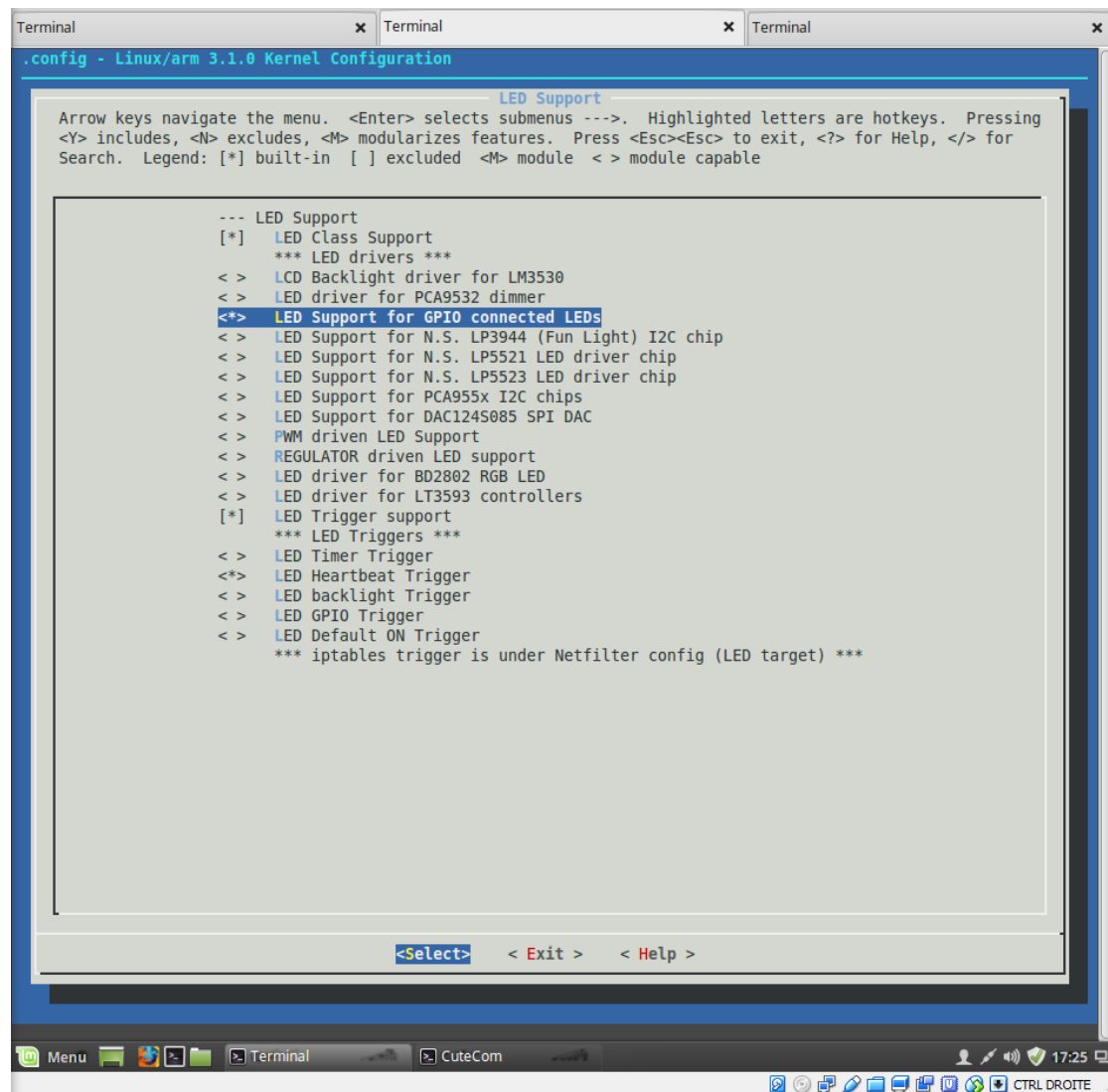


FIGURE 1.12 – Activation du driver pour les LEDs

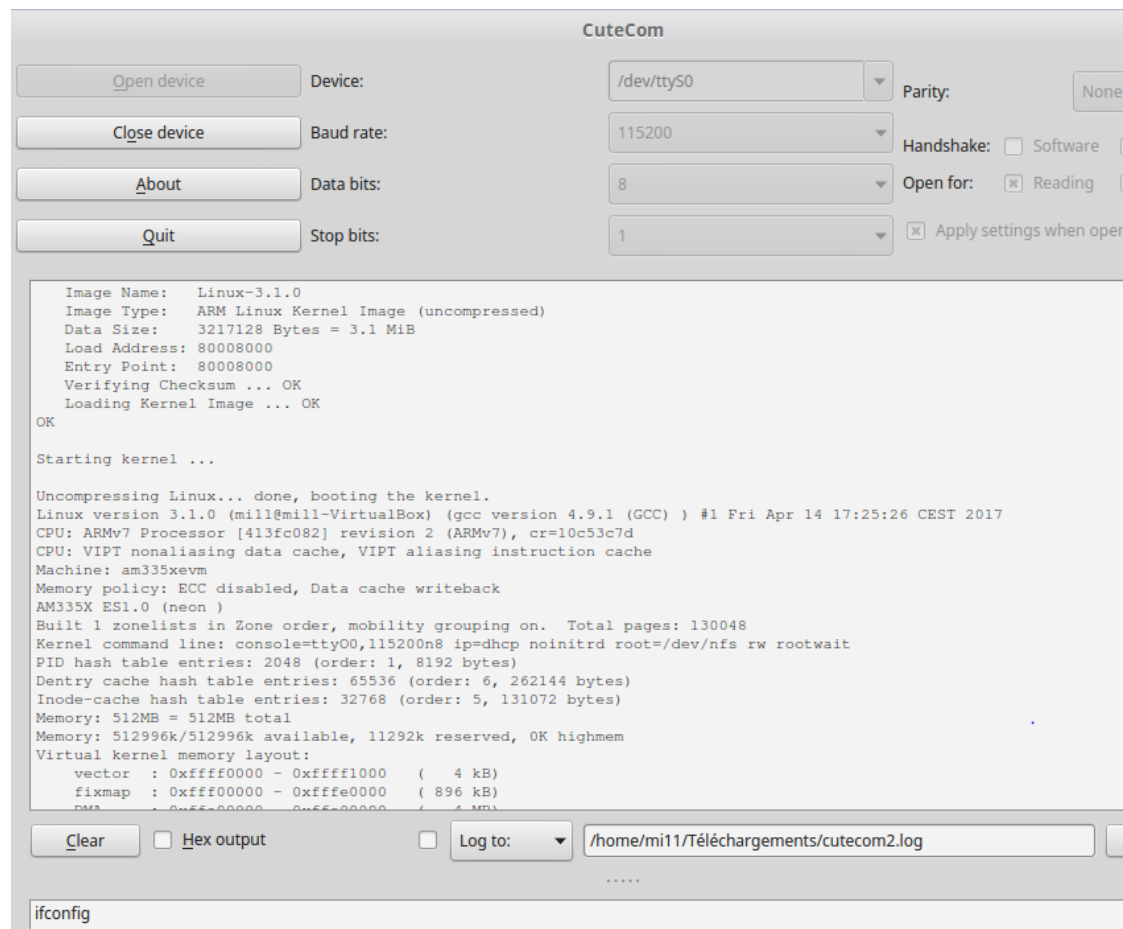


FIGURE 1.13 – Logs de démarrage

```

root@devkit8600:/sys/class/leds/user_led# dmesg | grep led
Memory policy: ECC disabled, Data cache writeback
Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
console [ttyO0] enabled
Registered led device: sys_led
Registered led device: user_led
sgtl5000 1-000a: Failed to add route HPL0UT->Headphone Jack

```

FIGURE 1.14 – Logs du kernel

TP1 Xenomai

Exercice 1 : tâches

1.1 Question 1.1

Un code "classique" ne s'exécute pas de façon temps réel. En effet, il n'apparaît pas dans le fichier `/proc/xenomai/stats` ce qui signifie qu'il n'est pas temps réel.

```
root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID  MSW   CSW   PF   STAT   %CPU  NAME
0  0      0      0      0  00500080  100.0  ROOT
0  0      0  9592      0  00000000   0.0  IRQ68: [timer]
```

FIGURE 2.1 – Statistiques Xenomai avec un programme non temps réel

1.2 Question 1.2

Voici le code qui permet de créer une tâche temps réel :

```
1 #include <stdio.h>
2 #include <native/task.h>
3 #include <analogy/os_facilities.h>
4 #include <unistd.h>
5 #include <sys/mman.h>
6
7 #define TASK_PRIO 99
8 #define TASK_MODE 0
9 #define TASK_STKSZ 0
10
11 RT_TASK task_printf;
12
13 void task_hello()
14 {
15     while(1)
16     {
17         sleep(rt_timer_ns2ticks(1000000000));
18         printf("Hello World !\n");
19     }
20 }
21
```

```
int main()  
23 {  
    mlockall(MCL_CURRENT|MCL_FUTURE);  
25    rt_print_auto_init(1);  
    int err;  
27  
    err = rt_task_create(&task_printf, "hello world", TASK_STKSZ,  
        TASK_PRIO, TASK_MODE);  
29    if (!err)  
        rt_task_start(&task_printf, &task_hello, NULL);  
31  
    getchar();  
33  
    return 0;  
35 }
```

Le chemin des fichiers à inclure est :

```
1 /opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi/usr/  
    include/xenomai
```

Le chemin des bibliothèques est :

```
1 /opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi/usr/lib
```

Cela nous donne la ligne de commande (après avoir fait un *source*) :

```
1 $CC -o main main.c -I/opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-  
    linux-gnueabi/usr/include/xenomai -L/opt/poky/1.7.3/sysroots/  
    armv7a-vfp-neon-poky-linux-gnueabi/usr/lib -lxenomai -lnative
```

1.3 Question 1.3

Cette application n'est toujours pas temps réel. En effet, malgré la création d'une tâche temps réel, les fonctions *sleep* et *printf* qui s'y trouvent ne sont pas temps réel.

Le fichier de statistiques Xenomai donne :


```

root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID  MSW   CSW   PF   STAT  %CPU  NAME
 0    0    0     335   0    00500080 100.0  ROOT
 0    0    0    11084  0    00000000  0.0  IRQ68: [timer]

```

FIGURE 2.2 – Statistiques Xenomai avec une tâche temps réel

1.4 Question 1.4

Voici le code après avoir remplacé la fonction sleep par son équivalent temps réel :

```

1 void task_hello()
2 {
3     while(1)
4     {
5         rt_task_sleep(rt_timer_ns2ticks(1000000000));
6         printf("Hello World !\n");
7     }
8 }

```

Le fichier de statistiques Xenomai donne :

```

root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID  MSW   CSW   PF   STAT  %CPU  NAME
 0    0    0     202   0    00500080 100.0  ROOT
 0   965    4      9   0    00300184  0.0  hello world
 0    0    0    7802  0    00000000  0.0  IRQ68: [timer]

```

FIGURE 2.3 – Statistiques Xenomai avec une tâche temps réel et la fonction rt_task_sleep

1.5 Question 1.5

Voici le code après avoir remplacé les fonctions sleep et printf par leurs équivalents temps réel :

```

1 void task_hello()
2 {
3     while(1)
4     {
5         rt_task_sleep(rt_timer_ns2ticks(1000000000));
6         rt_printf("Hello World !\n");
7     }
8 }

```

Le fichier de statistiques Xenomai donne :

```
root@devkit8600-xenomai:~# cat /proc/xenomai/stat
CPU  PID  MSW   CSW   PF   STAT   %CPU  NAME
0    0     0     46    0    00500080 100.0  ROOT
0    957   0     46    0    00300184 0.0    hello world
0    0     0    2860   0    00000000 0.0    IRQ68: [timer]
```

FIGURE 2.4 – Statistiques Xenomai avec une tâche temps réel et les fonctions `rt_task_sleep` et `rt_printf`

Interprétation des statistiques Xenomai : On constate que :

- Lorsqu'on crée une tâche temps réel qui n'utilise aucune fonction temps réel (dans notre exemple, les fonctions *sleep* et *printf*, elle n'apparaît pas dans le fichier de statistiques. Elle n'est en fait absolument pas temps réel.
- Lorsque, dans cette même tâche, on rend la fonction *sleep* temps réel (en la remplaçant par *rt_task_sleep*), elle apparaît dans le fichier de statistiques. On constate que son nombre de changements de contexte (9) est très inférieur au nombre de changements de contexte du ROOT (282), ce qui signifie que ... A COMPLETER
- Lorsque les fonctions *sleep* et *printf* ont été remplacées par leurs équivalents temps réel (*rt_task_sleep* et *rt_printf*), le nombre changements de contextes de la tâche est égal à celui de ROOT (46), ce qui montre alors que la tâche est "entièrement" temps réel.

Exercice 2 : Synchronisation

2.1 Question 2.1

Voici le code du programme lançant deux tâches Xenomai qui afficheront chacune une partie du message "Hello World!" :

```
#include <stdio.h>
2 #include <native/task.h>
#include <analogy/os_facilities.h>
4 #include <unistd.h>
#include <sys/mman.h>
6
#define TASK_PRIO 99
8 #define TASK_MODE 0
#define TASK_STKSZ 0
```

```
10 RT_TASK task_printf;
12 RT_TASK task_printf2;

14 void task_hello()
15 {
16     rt_printf("Hello\n");
17 }

18 void task_world()
19 {
20     rt_printf("World !\n");
22     rt_task_sleep(rt_timer_ns2ticks(1000000000));
23 }

24

26 int main()
27 {
28     mlockall(MCL_CURRENT|MCL_FUTURE);
29     rt_print_auto_init(1);
30     int err1, err2;

32     err1 = rt_task_create(&task_printf, "hello", TASK_STKSZ, TASK_PRIO,
33                          TASK_MODE);
34     err2 = rt_task_create(&task_printf2, "world", TASK_STKSZ, TASK_PRIO,
35                          TASK_MODE);
36     if (!err1 && !err2)
37     {
38         rt_task_start(&task_printf, &task_hello, NULL);
39         rt_task_join(&task_printf);
40         rt_task_start(&task_printf2, &task_world, NULL);
41         rt_task_join(&task_printf2);
42     }

44     getchar();

46     return 0;
47 }
```

Le résultat est le suivant :

```
1 root@devkit8600-xenomai:~# ./synchro
Hello
3 World !
c
5 root@devkit8600-xenomai:~#
```

2.2 Question 2.2

Pour le moment, les priorités des tâches n'ont aucune influence. En effet, elles sont lancées dans l'ordre dans lequel elles se trouvent dans notre code : après le `rt_task_start`, rien ne "bloque" l'exécution de la tâche ; les tâches se lancent donc l'une après l'autre comme on l'a défini. Pour afficher les messages de le désordre, il faut donc inverser les deux tâches le code. On peut aussi utiliser des sémaphores, comme vu dans les questions suivantes.

2.3 Question 2.3

Il faut initialiser le sémaphore à 0 de manière à bloquer les tâches.

2.4 Question 2.4

Le paramètre *mode* lors de la création du sémaphore nous sert à définir le mode d'ordonnancement à utiliser. Par exemple, si on choisit le mode *S_FIFO* alors les tâches seront ordonnancés en suivant la méthode FIFO (First In First Out), c'est à dire que les tâches attendront dans leur ordre d'arrivée que le sémaphore se libère. Un second exemple de *mode* utilisable est le mode *S_PRIO* qui permet d'ordonner les tâches par ordre de priorité, c'est à dire que les tâches avec la plus haute priorité auront accès au sémaphore avant les tâches de plus faible priorité.

2.5 Question 2.5

```
1 #include <stdio.h>
  #include <native/task.h>
3 #include <analogy/os_facilities.h>
  #include <unistd.h>
5 #include <sys/mman.h>
  #include <native/sem.h>
7
  #define TASK_PRIO_HELLO 98
9 #define TASK_PRIO_WORLD 99
  // #define TASK_PRIO 99
11 #define TASK_MODE 0
  #define TASK_STKSZ 0
13
14 RT_TASK task_printf;
15 RT_TASK task_printf2;
  RT_SEM sem;
17
```

```
19 void task_hello()      // affichage de 'Hello'
20 {
21     rt_sem_p (&sem, 0); // décrémentation du sémaphore
22     rt_printf("Hello\n");
23 }
24
25 void task_world()      // affichage de 'World'
26 {
27     rt_sem_p (&sem, 0); // décrémentation du sémaphore
28     rt_printf("World !\n");
29 }
30
31 int main()
32 {
33     mlockall(MCL_CURRENT|MCL_FUTURE);
34     rt_print_auto_init(1);
35     int err1, err2;
36
37     // création du sémaphore en mode FIFO
38     rt_sem_create (&sem, "sem", 0, S_FIFO);
39
40     // création des deux tâches
41     err1 = rt_task_create(&task_printf, "hello", TASK_STKSZ,
42                          TASK_PRIO_HELLO, TASK_MODE);
43     err2 = rt_task_create(&task_printf2, "world", TASK_STKSZ,
44                          TASK_PRIO_WORLD, TASK_MODE);
45
46     if (!err1 && !err2)
47     {
48         // démarrage de la tâche qui affiche 'Hello'
49         rt_task_start(&task_printf, &task_hello, NULL);
50
51         // attente de sa terminaison
52         rt_task_join(&task_printf);
53
54         // démarrage de la tâche qui affiche 'World'
55         rt_task_start(&task_printf2, &task_world, NULL);
56
57         // attente de sa terminaison
58         rt_task_join(&task_printf2);
59
60         getchar();
61
62         rt_sem_v (&sem); // incrémentation du sémaphore
63         rt_sem_v (&sem); // incrémentation du sémaphore
64     }
65 }
```

```
67 }  
    return 0;
```

Le programme précédent avec le mode *S_FIFO* pour le sémaphore nous donne le résultat suivant :

```
1 root@devkit8600-xenomai:~# ./synchro  
3 Hello  
   World !
```

Cependant, si nous utilisons le mode *S_PRIO* pour le sémaphore nous obtenons :

```
root@devkit8600-xenomai:~# ./synchro  
2  
   World !  
4 Hello
```

Ceci est cohérent puisque nous avons défini les priorités comme suit :

```
1 #define TASK_PRIO_HELLO 98  
2 #define TASK_PRIO_WORLD 99
```

La tâche qui s'occupe d'afficher le 'World!' est plus prioritaire que la tâche qui affiche le 'Hello'.

2.6 Question 2.6

```
#include <stdio.h>  
2 #include <native/task.h>  
  #include <analogy/os_facilities.h>  
4 #include <unistd.h>  
  #include <sys/mman.h>  
6 #include <native/sem.h>  
  
8 #define TASK_PRIO_HELLO 98  
  #define TASK_PRIO_WORLD 97  
10 #define TASK_PRIO_METRO 99  
   // #define TASK_PRIO 99  
12 #define TASK_MODE 0  
   #define TASK_STKSZ 0
```

```
14 RT_TASK task_printf;
16 RT_TASK task_printf2;
17 RT_TASK task_metronome;
18 RT_SEM sem;

20 void task_hello()
21 {
22     while(1)
23     {
24         rt_sem_p (&sem, 0); // décrémentation du sémaphore
25         rt_printf("Hello\n");
26     }
27 }

28 void task_world()
29 {
30     while(1)
31     {
32         rt_sem_p (&sem, 0); // décrémentation du sémaphore
33         rt_printf("World !\n");
34         rt_sem_v (&sem); // incrémentation du sémaphore
35     }
36 }

38 void task_metro() // tâche métronome
39 {
40     while(1)
41     {
42         rt_sem_v (&sem); // incrémentation du sémaphore
43         rt_sem_v (&sem); // incrémentation du sémaphore

44         rt_sem_p (&sem, 0); // décrémentation du sémaphore
45         rt_task_sleep(rt_timer_ns2ticks(1000000000)); // attente 1 sec
46     }
47 }

50

52 int main()
53 {
54     mlockall(MCL_CURRENT|MCL_FUTURE);
55     rt_print_auto_init(1);
56     int err1, err2, err3;

58     // création du sémaphore en mode PRIO
59     rt_sem_create (&sem, "sem", 0, S_PRIO);

60     err1 = rt_task_create(&task_printf, "hello", TASK_STKSZ,
        TASK_PRIO_HELLO, TASK_MODE);
```

```
62 err2 = rt_task_create(&task_printf2, "world", TASK_STKSZ,  
TASK_PRIO_WORLD, TASK_MODE);  
  
64 // création de la tâche métronome  
err3= rt_task_create(&task_metronome, "metro", TASK_STKSZ,  
TASK_PRIO_METRO, TASK_MODE);  
  
66 if (!err1 && !err2 && !err3)  
68 {  
    rt_task_start(&task_printf, &task_hello, NULL);  
70    rt_task_join(&task_printf);  
    rt_task_start(&task_printf2, &task_world, NULL);  
72    rt_task_join(&task_printf2);  
  
74    // démarrage de la tâche métronome  
    rt_task_start(&task_metronome, &task_metro, NULL);  
76    // attente de sa terminaison  
    rt_task_join(&task_metronome);  
  
78    getchar();  
  
80 }  
  
82 return 0;  
84 }
```

Le programme ci-dessus nous donne le résultat suivant à l'infini :

```
root@devkit8600-xenomai:~# ./synchro  
2 Hello  
4 World !  
Hello  
6 World !  
Hello  
8 World !
```

La tâche métronome possède la plus haute priorité comme on peut le voir ci-dessous :

```
#define TASK_PRIO_HELLO 98  
2 #define TASK_PRIO_WORLD 97  
#define TASK_PRIO_METRO 99
```


Initialement, les tâches 'Hello' et 'World' sont bloquées. A chaque activation de la tâche métronome, le sémaphore est incrémenté de 2. A la première incrémentation, c'est la tâche 'Hello' qui prend la main puisque elle a une priorité supérieure à la tâche 'World'. La tâche 'Hello' décrémente le sémaphore et affiche 'Hello'. Ensuite, la tâche métronome reprend son exécution puisque la tâche 'World' est bloquée et elle incrémente une seconde fois le sémaphore. La tâche 'World' est donc débloquée, elle décrémente le sémaphore, affiche 'World!' puis incrémente le sémaphore pour débloquer la tâche métronome qui va reprendre son exécution en décrémentant le sémaphore et en faisant une attente d'une seconde avant de relancer le même processus. *** CLEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEMENT : TU PEUX CONFIRMER CE QUE JE DIS J'AI UN DOUTE ***

2.7 Question 2.7

Le fichier de statistiques de Xenomai lors du lancement de notre programme est le suivant :

```
root@devkit8600-xenomai:~# cat /proc/xenomai/stat
*CPU  PID    MSW      CSW      PF      STAT      %CPU  NAME
0  0      0        513      0      00500080  99.8  ROOT
0  1264   0        43       0      00300182  0.0   hello
0  1265   0        85       0      00300182  0.0   world
0  1266   0        84       0      00300184  0.0   metro
0  0      0        53562    0      00000000  0.0   IRQ68: [timer]
```

FIGURE 2.5 – Fichier de statistiques de Xenomai

On retrouve bien nos 3 tâches 'hello', 'world' et 'metro'. On voit également que ces tâches ne subissent que très peu de changements de contexte (CSW) car ce sont des tâches temps-réel.

Pour le fichier du scheduler de Xenomai nous avons :

```
root@devkit8600-xenomai:~# cat /proc/xenomai/sched
CPU  PID    CLASS  PRI      TIMEOUT  TIMEBASE  STAT  NAME
0  0      idle   -1       -        master    R     ROOT
0  1264   rt     98       -        master    W     hello
0  1265   rt     97       -        master    W     world
0  1266   rt     99       625ms 11us master    D     metro
```

FIGURE 2.6 – Fichier du scheduler de Xenomai

Nous retrouvons bien les priorités de nos tâches que nous avons défini dans notre code. On voit également que nos trois tâches sont temps-réel via la colonne *CLASS* (rt). On voit également que la tâche 'ROOT' qui correspond à Linux est en mode 'idle', c'est à dire qu'elle a priorité la plus faible (-1). Dans la colonne *STAT*, on remarque que la tâche 'metro' est marquée de la lettre 'D' qui signifie 'Delayed', c'est à dire que la tâche est retardée sans aucune autre condition d'attente (wait d'une seconde). Les tâches 'hello' et 'metro' sont marquées de la lettre 'W' qui signifie que ces tâches sont en attente d'une ressource (sémaphore) et la tâche 'ROOT' est marquée de la lettre 'R' qui signifie 'Runnable', c'est à dire que la tâche est exécutable.

Exercice 3 : Latence

3.1 Question 3.1

```
1 #include <stdio.h>
2 #include <native/task.h>
3 #include <analogy/os_facilities.h>
4 #include <unistd.h>
5 #include <sys/mman.h>
6 #include <native/sem.h>
7 #include <nucleus/timer.h>
8
9 #define TASK_PRIO 99
10 #define TASK_MODE 0
11 #define TASK_STKSZ 0
12 #define TAILLE 10000
13
14 RT_TASK task_latence;
15
16 void task_wait()
17 {
18     int i;
19
20     // boucle de 1 à 10000
21     for(i = 0; i < TAILLE; i++)
22     {
23         // attente de 1ms
24         rt_task_sleep(rt_timer_ns2ticks(1000000));
25     }
26 }
27
28 int main()
29 {
30     mlockall(MCL_CURRENT|MCL_FUTURE);
```

```
31  rt_print_auto_init(1);
    int err1;
33
    err1 = rt_task_create(&task_latence, "wait", TASK_STKSZ, TASK_PRIO,
        TASK_MODE);
35
    if(!err1)
37    {
        rt_task_start(&task_latence, &task_wait, NULL);
39        rt_task_join(&task_latence);

41        getchar();
    }
43
45    return 0;
}
```

3.2 Question 3.2

```
#include <stdio.h>
2 #include <native/task.h>
#include <analogy/os_facilities.h>
4 #include <unistd.h>
#include <sys/mman.h>
6 #include <native/sem.h>
#include <nucleus/timer.h>
8

10 #define TASK_PRIO 99
#define TASK_MODE 0
12 #define TASK_STKSZ 0
#define TAILLE 10000
14
RT_TASK task_latence;
16

18 void task_wait()
{
    int i;
20    RTIME moy = 0, min, max;
    RTIME begin, end;

22    min = 999999999999999;
24    max = 0;

26
```

```
28  for(i = 0; i < TAILLE; i++)
    {
30      begin = rt_timer_read(); // lecture du temps
      rt_task_sleep(rt_timer_ns2ticks(1000000));
      end = rt_timer_read(); // lecture du temps
32
      if(min > end - begin) // calcul du min
34          min = end - begin;

36      if(max < end - begin) // calcul du max
      {
38          max = end - begin;
      }

40      moy += end - begin; // calcul de la moyenne
42  }

44  moy = moy / TAILLE;

46  rt_printf("Moyenne :%llu \nMaximum : %llu \nMinimum : %llu\n", moy,
      max, min);

48  }

50
51  int main()
52  {
53      mlockall(MCL_CURRENT|MCL_FUTURE);
54      rt_print_auto_init(1);
55      int err1;

56      err1 = rt_task_create(&task_latence, "wait", TASK_STKSZ, TASK_PRIO,
          TASK_MODE);

58
59      if(!err1)
60      {
61          rt_task_start(&task_latence, &task_wait, NULL);
62          rt_task_join(&task_latence);

64          getchar();
65      }

66

68      return 0;
    }
```

Le programme précédent nous donne le résultat suivant :

```
root@devkit8600-xenomai:~# ./latence
Moyenne :1002756
Maximum : 1044920
Minimum : 1002440
```

FIGURE 2.7 – Résultat du programme *Latence* sans stress

3.3 Question 3.3

Ici, on charge le CPU via la commande *stress* avant de lancer notre programme :

```
^Croot@devkit8600-xenomai:~# ./latence
Moyenne :1005789
Maximum : 1030760
Minimum : 1003560
```

FIGURE 2.8 – Résultat du programme *Latence* avec stress

On remarque que les résultats avec charge CPU et sans charge CPU sont pratiquement les mêmes ce qui veut dire que la charge n'a aucune influence sur les tâches exécutées. Ceci est logique puisque le programme est temps-réel donc une charge CPU ne ralentira que les programmes non temps-réel qui sont toujours reliés à Linux (ROOT) comme on a pu le constater dans le tp précédent. Ici, les tâches temps-réel ont la plus grande priorité sur le CPU.

TP2 Xenomai

Exercice : Pathfinder

1.1 Question 1

La fonction *create_and_start_rt_task...*

La fonction *rt_task...*

La structure *task_descriptor* permet d'obtenir des informations de la tâche en cours, c'est à dire :

- la tâche elle même (de type `RT_TASK`)
- le pointeur vers la fonction associée
- la période de la tâche
- la durée d'exécution la tâche
- la priorité de la tâche
- si la tâche utilise une ressource ou non

1.2 Question 2

La fonction *rt_task_name* sert à obtenir le nom de la tâche en cours grâce à la structure `RT_TASK_INFO` et son champ *name*. Cette structure a d'autres champs :

- *bprio* : priorité de base (ne change pas au cours du temps)
- *cprio* : priorité actuelle (peut changer au cours du temps)
- *status* : statut de la tâche
- *relpoint* : temps restant avant la prochaine exécution
- *exectime* : temps d'exécution de la tâche depuis son lancement
- *modeswitches* : nombre de changements de mode primaire / secondaire
- *ctxswitches* : nombre de changements de contextes
- *pagefaults* : nombre de défauts de page

1.3 Question 3

Voici notre fonction *busy_wait* :

```
1 void busy_wait(RTIME time)
  {
3   static RT_TASK_INFO info;    // info sur la tâche
   rt_task_inquire(NULL,&info);  // initialisation des infos dont
   exectime
5   RTIME begin = info.exectime;  // recuperation du temps d'
   execution initial (> 0)

7
   while(info.exectime - begin < time)
9       rt_task_inquire(NULL,&info); // initialisation des infos
   dont exectime
  }
```

On commence par acquérir les informations relatives à la tâche grâce à une structure `RT_TASK_INFO` et à la fonction `rt_task_inquire`. La champ *exectime* de cette structure nous permet d'obtenir le temps d'exécution de la tâche depuis son lancement (donc il y a certainement eu plusieurs occurrences). On appelle ce temps *begin*. Ce qu'on veut, c'est simuler le temps d'une exécution de la tâche.

Pour cela, on va, dans une boucle, vérifier la différence entre le temps d'exécution actuel de la tâche (avec le champ *exectime* de `RT_TASK_INFO` mis à jour) et le temps d'exécution de la tâche initial (qu'on a appelé *begin*). Dès que cette différence atteint la durée voulue (durée d'exécution de la tâche donnée en paramètre), on peut sortir de la boucle et de la fonction. De cette façon, on a réalisé une attente active pendant la durée recherchée.

1.4 Question 4

On se sert de la fonction `time_since_start` dans la fonction `rt_task` pour avoir un point de repère temporel :

```
rt_printf("doing %s      time : %d\n",rt_task_name(), time_since_start
());
```

Voici le résultat de l'exécution du programme :

```
root@devkit8600-xenomai:~# ./pathfinder
started task ORDO_BUS, period 125ms, duration 25ms, use resource 0
doing ORDO_BUS    time : 125
doing ORDO_BUS ok  time : 150
doing ORDO_BUS    time : 250
doing ORDO_BUS ok  time : 275
doing ORDO_BUS    time : 375
doing ORDO_BUS ok  time : 400
doing ORDO_BUS    time : 500
doing ORDO_BUS ok  time : 525
doing ORDO_BUS    time : 625
doing ORDO_BUS ok  time : 650
doing ORDO_BUS    time : 750
doing ORDO_BUS ok  time : 775
doing ORDO_BUS    time : 875
doing ORDO_BUS ok  time : 900
doing ORDO_BUS    time : 1000
doing ORDO_BUS ok  time : 1025
doing ORDO_BUS    time : 1125
doing ORDO_BUS ok  time : 1150
doing ORDO_BUS    time : 1250
doing ORDO_BUS ok  time : 1275
```

FIGURE 3.1 – Résultat avec ORDO_BUS

Le timing est bon : 25ms d'exécution et 125ms de période.

1.5 Question 5

Pour une bonne coordination des tâches, le sémaphore doit être utilisé comme suit :

- Faire un `sem_p` (-1) dans `acquire_resource` et un `sem_v` (+1) dans `release_resource` sur le sémaphore
- Initialisation du sémaphore à 0 (bloque tout)
- Initialisation de toutes les tâches (et donc toutes les tâches sont bloquées)
- Faire un `sem_v` (+1) sur le sémaphore juste après ces initialisations (ce qui débloquera la tâche la plus prioritaire)

De cette façon, les tâches s'enchaîneront de la bonne manière selon leurs priorités.


```

doing METEO      time : 5226
doing ORDO_BUS   time : 5250
doing ORDO_BUS ok time : 5275
doing RADIO      time : 5275
doing RADIO ok   time : 5300
doing CAMERA     time : 5300
doing CAMERA ok  time : 5325
doing METEO ok   time : 5341
doing DISTRIB_DONNEES time : 5341
doing DISTRIB_DONNEES ok time : 5366
doing PILOTAGE   time : 5366
doing ORDO_BUS   time : 5375
doing ORDO_BUS ok time : 5400
doing PILOTAGE ok time : 5400
doing DISTRIB_DONNEES time : 5400
doing DISTRIB_DONNEES ok time : 5425
doing ORDO_BUS   time : 5500
doing ORDO_BUS ok time : 5525

```

FIGURE 3.2 – Enchaînement avec meilleur cas pour METEO (40ms)

```

doing METEO      time : 5226
doing ORDO_BUS   time : 5250
doing ORDO_BUS ok time : 5275
doing RADIO      time : 5275
doing RADIO ok   time : 5300
doing CAMERA     time : 5300
doing CAMERA ok  time : 5325
doing METEO ok   time : 5361
doing DISTRIB_DONNEES time : 5361
doing ORDO_BUS   time : 5375
doing ORDO_BUS ok time : 5400
doing DISTRIB_DONNEES ok time : 5411
doing PILOTAGE   time : 5411
doing PILOTAGE ok time : 5436
doing DISTRIB_DONNEES time : 5436
doing DISTRIB_DONNEES ok time : 5461
doing ORDO_BUS   time : 5500
doing ORDO_BUS ok time : 5525

```

FIGURE 3.3 – Enchaînement avec pire cas pour METEO (60ms)

On se rend compte que, lors du meilleur cas pour la tâche METEO (soit 40ms), ORDO_BUS n'a pas encore terminé sa période et n'est donc pas prêt pour exécution avant le début de la tâche PILOTAGE ou la fin de DISTRIB_DONNEES. Dans le pire cas pour la tâche METEO (soit 60ms), ORDO_BUS a atteint sa période et est donc exécuté avant la tâche PILOTAGE et s'intercale entre le début et le fin de l'exécution de DISTRIB_DONNEES.

1.6 Question 6

Pour être sûr qu'entre deux exécutions de `ORDO_BUS`, `DISTRIB_DONNEES` s'est exécutée entièrement, nous pouvons utiliser une seconde sémaphore en mode `S_PRIO` (comme la première) qui vaut 1 avant la création des tâches pour que la tâche `ORDO_BUS` puisse s'exécuter au début. Nous créons ensuite deux nouvelles fonctions `rt_task_ordo_bus` et `rt_task_distrib_donnees` pour remplacer la fonction de base `rt_task`. Dans la fonction `rt_task_distrib_donnees`, lors de l'exécution de la tâche `DISTRIB_DONNEES`, il faut incrémenter la sémaphore pour prévenir la tâche `ORDO_BUS` que la tâche `DISTRIB_DONNEES` est en cours d'exécution. Si la tâche `DISTRIB_DONNEES` est en cours d'exécution et qu'on est en train d'exécuter la tâche `ORDO_BUS` alors il faut lancer un reset et terminer le programme. Pour cela, dans la fonction `rt_task_ordo_bus`, il faut vérifier si une unité du sémaphore est disponible, si c'est le cas alors on exécute la tâche normalement sinon on lance un reset et on termine le programme. Il faut faire attention de ne pas bloquer la tâche `ORDO_BUS` sur le sémaphore car la suite du code doit pouvoir s'exécuter.

1.7 Question 7

Nous avons testé notre programme pour le cas extrême du temps d'exécution de `METEO` (60ms) sans terminer notre programme après le reset afin de mieux voir ce qu'il se passe. Voici le résultat obtenu :

```
doing METEO    time : 5226
doing ORDO_BUS time : 5250
doing ORDO_BUS ok time : 5275
doing RADIO    time : 5275
doing RADIO ok  time : 5300
doing CAMERA    time : 5300
doing CAMERA ok  time : 5325
doing METEO ok   time : 5361
doing DISTRIB_DONNEES time : 5361
RESET

doing ORDO_BUS time : 5375
doing ORDO_BUS ok time : 5400
doing DISTRIB_DONNEES ok time : 5411
doing PILOTAGE  time : 5411
```

FIGURE 3.4 – Reset avec le cas extrême pour `METEO` (60ms)

On voit que le reset se place entre le début de la tâche DISTRIB_DONNEES et le début de la tâche ORDO_BUS ce qui valide le fonctionnement de la solution.

Cependant, on remarque que lorsque la tâche METEO démarre elle s'empare du sémaphore puis elle est préemptée par la tâche ORDO_BUS de plus haute priorité. A la fin de la tâche ORDO_BUS, la tâche DISTRIB_DONNEES démarre mais elle est bloquée par le sémaphore car c'est la tâche METEO qui le possède. Ensuite, après les exécutions des tâches RADIO et CAMERA, la tâche METEO reprend la main et se termine en libérant le sémaphore ce qui permet à la tâche DISTRIB_DONNEES de reprendre son exécution avant le Reset. Ceci est problématique car une tâche de plus haute priorité (DISTRIB_DONNEES) ne peut pas avoir accès au processeur car une tâche de plus faible priorité (METEO) l'utilise. C'est ce qu'on appelle une inversion de priorité. Le chronogramme suivant illustre ce qu'il se passe :

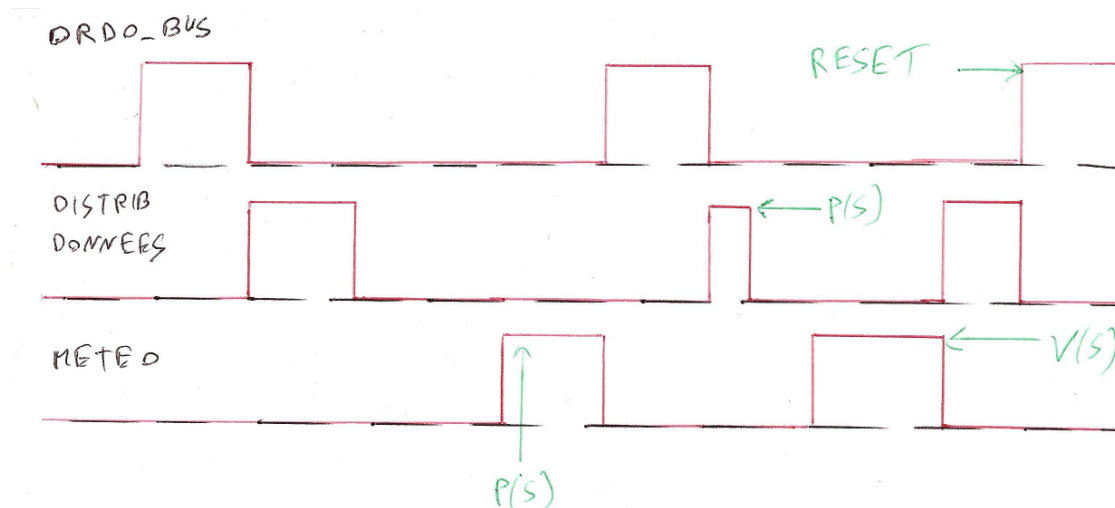


FIGURE 3.5 – Chronogramme illustrant l'inversion de priorité

1.8 Question 8

Afin de résoudre ce problème, nous pouvons utiliser un mutex à la place de notre premier sémaphore. En effet, un mutex hérite de la priorité la plus élevée parmi celles de toutes les tâches qui le demandent et une tâche s'exécute avec la priorité la plus élevée parmi celles de tous les mutex qu'il tient.

1.9 Question 9

Après lancement du programme, nous obtenons le résultat suivant :

```
Name : ORDO_BUS Base prio : 7 Current prio : 7
doing ORDO_BUS ok time : 10275
Name : METEO Base prio : 1 Current prio : 6
doing METEO ok time : 10275
doing DISTRIB_DONNEES time : 10275
Name : DISTRIB_DONNEES Base prio : 6 Current prio : 6
doing DISTRIB_DONNEES ok time : 10300
```

FIGURE 3.6 – Résultat avec un mutex

On voit bien que la tâche METEO a hérité de la priorité de la tâche DISTRIB_DONNEES (6) ce qui lui permet de reprendre son exécution normalement et de se terminer. Ensuite la tâche DISTRIB_DONNEES prend la main et s'exécute. Il n'y a plus du tout d'inversion de priorité.

1.10 Question 10

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #include <sys/mman.h>
5
6 #include <native/task.h>
7 #include <native/timer.h>
8 #include <native/sem.h>
9 #include <native/mutex.h>
10 #include <rtdk.h>
11
12 #define TASK_MODE T_JOINABLE
13 #define TASK_STKSZ 0
14
15 RTIME init_time;
16 RT_SEM sem;
17 RT_SEM semReset;
18 RT_MUTEX mutex;
19
20
21 typedef struct task_descriptor{
22     RT_TASK task;
23     void (*task_function)(void*);
```

```

25  RTIME period;
    RTIME duration;
    int priority;
27  bool use_resource;
    } task_descriptor;
29
    //////////////////////////////////////
31 char* rt_task_name(void) {
    static RT_TASK_INFO info;
33  rt_task_inquire(NULL,&info);

    return info.name;
    }
37
    //////////////////////////////////////
39 int time_since_start(void) {
    return (rt_timer_read()-init_time)/1000000;
41 }

    //////////////////////////////////////
43 void acquire_resource(void) {
45
    // Version non fonctionnelle avec sémaphore
47  // rt_sem_p (&sem, TM_INFINITE );

    // Version fonctionnelle avec mutex
    rt_mutex_acquire (&mutex, TM_INFINITE);
51 }

    //////////////////////////////////////
53 void release_resource(void) {
55
    // Version non fonctionnelle avec sémaphore
57  // rt_sem_v (&sem);

    // Version fonctionnelle avec mutex
    rt_mutex_release(&mutex);
61 }

    //////////////////////////////////////
63 void busy_wait(RTIME time)
65 {
    static RT_TASK_INFO info;      // Info sur la tâche

67
    // Initialisation des infos dont exectime
69  rt_task_inquire(NULL,&info);

71
    // Recuperation du temps d'execution initial (> 0)
    RTIME begin = info.exectime;

```

```

73
75     while(info.exectime - begin < time)
76     {
77         // Mise à jour des infos dont exectime
78         rt_task_inquire(NULL,&info);
79     }
80 }
81
82 void rt_task(void *cookie) {
83     struct task_descriptor* params=(struct task_descriptor*)cookie;
84
85     rt_printf("started task %s, period %ims, duration %ims, use
86             resource %i\n",rt_task_name(),(int)(params->period/1000000),(int)(
87             params->duration/1000000),params->use_resource);
88
89     while(1) {
90         rt_task_wait_period(NULL);
91         if(params->use_resource) acquire_resource();
92         rt_printf("doing %s      time : %d\n",rt_task_name(),
93             time_since_start());
94         busy_wait(params->duration);
95
96         static RT_TASK_INFO info;      // Info sur la tâche
97         rt_task_inquire(NULL,&info);    // Initialisation des infos
98
99         // Test du mutex avec priorité de base et priorité courrante
100        rt_printf("Name : %s Base prio : %d Current prio : %d\n", info.
101            name, info.bprio, info.cprio);
102
103        rt_printf("doing %s ok      time : %d\n",rt_task_name(),
104            time_since_start());
105        if(params->use_resource) release_resource();
106    }
107 }
108
109 // Fonction de la tâche ORDO_BUS implémentant le RESET
110 void rt_task_ordo_bus(void *cookie) {
111     struct task_descriptor* params=(struct task_descriptor*)cookie;
112
113     rt_printf("started task %s, period %ims, duration %ims, use
114             resource %i\n",rt_task_name(),(int)(params->period/1000000),(int)(
115             params->duration/1000000),params->use_resource);
116
117     while(1) {
118         rt_task_wait_period(NULL);
119         if(params->use_resource) acquire_resource();

```

```

115 // Si la décrémentation n'est pas possible, on ne bloque pas la tâche
117 if(rt_sem_p (&semReset, TM_NONBLOCK ) == -EWOULDBLOCK )
119 {
121     rt_printf("RESET\n\n\n\n\n\n\n\n\n\n\n"); // Affichage du RESET
    exit(-1); // Exit du programme
123 }

    rt_printf("doing %s      time : %d\n",rt_task_name() ,
time_since_start());
    busy_wait(params->duration);

125     static RT_TASK_INFO info;
127     rt_task_inquire(NULL,&info);

129 // Test du mutex avec priorité de base et priorité courante
    rt_printf("Name : %s Base prio : %d Current prio : %d\n", info.
name, info.bprio, info.cprio);

131     rt_printf("doing %s ok      time : %d\n",rt_task_name() ,
time_since_start());
133     if(params->use_resource) release_resource();
135 }

137 // Fonction de la tâche DISTRIB_DONNEES implémentant le RESET
139 void rt_task_distrib_donnees(void *cookie) {

141     struct task_descriptor* params=(struct task_descriptor*)cookie;

143     rt_printf("started task %s, period %ims, duration %ims, use
resource %i\n",rt_task_name() ,(int)(params->period/1000000) ,(int)(
params->duration/1000000),params->use_resource);

145     while(1) {
        rt_task_wait_period(NULL);
147         if(params->use_resource) acquire_resource();
        rt_printf("doing %s      time : %d\n",rt_task_name() ,
time_since_start());
149         busy_wait(params->duration);

151         static RT_TASK_INFO info;
        rt_task_inquire(NULL,&info);

153         // Test du mutex avec priorité de base et priorité courante
        rt_printf("Name : %s Base prio : %d Current prio : %d\n", info.
name, info.bprio, info.cprio);

```

```
157     rt_printf("doing %s ok      time : %d\n",rt_task_name() ,
time_since_start());

159     // Incrémentation du sémaphore pour le RESET
rt_sem_v (&semReset);

161     if(params->use_resource) release_resource();

163

165 }

167 }

169 int create_and_start_rt_task(struct task_descriptor* desc, char* name)
{
171     int status=rt_task_create(&desc->task, name, TASK_STKSZ, desc->
priority, TASK_MODE);
    if(status!=0) {
173         printf("error creating task %s\n", name);
        return status;
175     }

177     status=rt_task_set_periodic(&desc->task, TM_NOW, desc->period);
    if(status!=0) {
179         printf("error setting period on task %s\n", name);
        return status;
181     }

183     status=rt_task_start(&desc->task, desc->task_function, desc);
    if(status!=0) {
185         printf("error starting task %s\n", name);
    }
    return status;
187 }

189

191 int main(void) {
193     /* Avoids memory swapping for this program */
    mlockall(MCL_CURRENT|MCL_FUTURE);

195

197     rt_print_auto_init(1);

199     init_time=rt_timer_read();

201     // Version non fonctionnelle avec sémaphore
    //rt_sem_create (&sem, "sem", 0, S_PRIO);
```



```
203 // Version fonctionnelle avec mutex
205 rt_mutex_create (&mutex, "mutex");

207 // Sémaphore pour le RESET
rt_sem_create (&semReset, "semReset", 1, S_PRIO);

209 // Initialisation, création et lancement de la tâche ORDO_BUS
211 task_descriptor* ORDO_BUS_Descriptor = (task_descriptor*)malloc(
    sizeof(task_descriptor));
ORDO_BUS_Descriptor->task_function = rt_task_ordo_bus;
213 ORDO_BUS_Descriptor->priority = 7;
ORDO_BUS_Descriptor->duration = 25000000;
215 ORDO_BUS_Descriptor->period = 125000000;
ORDO_BUS_Descriptor->use_resource = 0;

217 create_and_start_rt_task(ORDO_BUS_Descriptor, "ORDO_BUS");

219

221 // Initialisation, création et lancement de la tâche
    DISTRIB_DONNEES
task_descriptor* DISTRIB_DONNEES_Descriptor = (task_descriptor*)
    malloc(sizeof(task_descriptor));
223 DISTRIB_DONNEES_Descriptor->task_function = rt_task_distrib_donnees
    ;
DISTRIB_DONNEES_Descriptor->priority = 6;
225 DISTRIB_DONNEES_Descriptor->duration = 25000000;
DISTRIB_DONNEES_Descriptor->period = 125000000;
227 DISTRIB_DONNEES_Descriptor->use_resource = 1;

229 create_and_start_rt_task(DISTRIB_DONNEES_Descriptor, "
    DISTRIB_DONNEES");

231

233 // Initialisation, création et lancement de la tâche PILOTAGE
task_descriptor* PILOTAGE_Descriptor = (task_descriptor*)malloc(
    sizeof(task_descriptor));
PILOTAGE_Descriptor->task_function = rt_task;
235 PILOTAGE_Descriptor->priority = 5;
PILOTAGE_Descriptor->duration = 25000000;
237 PILOTAGE_Descriptor->period = 250000000;
PILOTAGE_Descriptor->use_resource = 1;

239 create_and_start_rt_task(PILOTAGE_Descriptor, "PILOTAGE");

241

243 // Initialisation, création et lancement de la tâche RADIO
task_descriptor* RADIO_Descriptor = (task_descriptor*)malloc(sizeof
    (task_descriptor));
```

```
245 RADIO_Descriptor->task_function = rt_task;
    RADIO_Descriptor->priority = 4;
247 RADIO_Descriptor->duration = 25000000;
    RADIO_Descriptor->period = 250000000;
249 RADIO_Descriptor->use_resource = 0;

251 create_and_start_rt_task(RADIO_Descriptor, "RADIO");

253
    // Initialisation, création et lancement de la tâche CAMERA
255 task_descriptor* CAMERA_Descriptor = (task_descriptor*)malloc(
    sizeof(task_descriptor));
    CAMERA_Descriptor->task_function = rt_task;
257 CAMERA_Descriptor->priority = 3;
    CAMERA_Descriptor->duration = 25000000;
259 CAMERA_Descriptor->period = 250000000;
    CAMERA_Descriptor->use_resource = 0;

261 create_and_start_rt_task(CAMERA_Descriptor, "CAMERA");

263
    // Initialisation, création et lancement de la tâche MESURES
265 task_descriptor* MESURES_Descriptor = (task_descriptor*)malloc(
    sizeof(task_descriptor));
267 MESURES_Descriptor->task_function = rt_task;
    MESURES_Descriptor->priority = 2;
269 MESURES_Descriptor->duration = 50000000;
    MESURES_Descriptor->period = 5000000000;
271 MESURES_Descriptor->use_resource = 1;

273 create_and_start_rt_task(MESURES_Descriptor, "MESURES");

275
    // Initialisation, création et lancement de la tâche METEO
277 task_descriptor* METEO_Descriptor = (task_descriptor*)malloc(sizeof
    (task_descriptor));
    METEO_Descriptor->task_function = rt_task;
279 METEO_Descriptor->priority = 1;
    METEO_Descriptor->duration = 60000000; // 40000000
281 METEO_Descriptor->period = 5000000000;
    METEO_Descriptor->use_resource = 1;

283 create_and_start_rt_task(METEO_Descriptor, "METEO");

285
    // Version non fonctionnelle avec sémaphore
287 // rt_sem_v (&sem); // Incrémentation du sémaphore principal

289 getchar();
```

```
291 |  return  EXIT_SUCCESS;  
    | }
```

Messages de sortie du terminal entre l'allumage de la cible et le prompt de login

```

2 U-Boot SPL 2011.09-svn (May 22 2012 - 11:19:00)
  Texas Instruments Revision detection unimplemented
4 Booting from NAND...

6 U-Boot 2011.09-svn (May 22 2012 - 11:19:00)

8 I2C:   ready
10 DRAM: 512 MiB
  WARNING: Caches not enabled
12 Did not find a recognized configuration , assuming General purpose EVM
    in Profile 0 with Daughter board
  NAND: HW ECC Hamming Code selected
14 512 MiB
  MMC:   OMAP SD/MMC: 0
16 Net:   cpsw
  Hit any key to stop autoboot:  3 \0x08\0x08\0x08 2 \0x08\0x08\0x08 1
    \0x08\0x08\0x08 0
18 Card did not respond to voltage select!
  Booting from network ...
20 miiphy read id fail
  link up on port 0, speed 100, full duplex
22 BOOTP broadcast 1
  DHCP client bound to address 192.168.1.6
24 Using cpsw device
  TFTP from server 192.168.1.1; our IP address is 192.168.1.6
26 Filename 'uImage'.
  Load address: 0x82000000
28 Loading: *\0x08##
    #####
  \0x09 ##
    #####
30 \0x09 ##
    #####
  \0x09 ##
    #####

```

```

32 \0x09 ##
    #####
34 \0x09 ##
    #####
36 \0x09 ##
    #####
38 done
Bytes transferred = 3215152 (310f30 hex)
40 ## Booting kernel from Legacy Image at 82000000 ...
    Image Name:      Linux-3.1.0
    Image Type:      ARM Linux Kernel Image (uncompressed)
    Data Size:       3215088 Bytes = 3.1 MiB
    Load Address:   80008000
    Entry Point:     80008000
    Verifying Checksum ... OK
    Loading Kernel Image ... OK
48 OK

50 Starting kernel ...

52 Uncompressing Linux... done, booting the kernel.
Linux version 3.1.0 (mi11@mi11-VirtualBox) (gcc version 4.9.1 (GCC) )
    #1 Mon Apr 10 18:15:11 CEST 2017
54 CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c7d
CPU: VIPT nonaliasing data cache, VIPT aliasing instruction cache
56 Machine: am335xevm
Memory policy: ECC disabled, Data cache writeback
58 AM335X ES1.0 (neon )
Built 1 zonelists in Zone order, mobility grouping on. Total pages:
130048
60 Kernel command line: console=ttyO0,115200n8 ip=dhcp noinitrd root=/
dev/nfs rw rootwait
PID hash table entries: 2048 (order: 1, 8192 bytes)
62 Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
64 Memory: 512MB = 512MB total
Memory: 512996k/512996k available, 11292k reserved, 0K highmem
66 Virtual kernel memory layout:
    vector   : 0xffff0000 - 0xffff1000   (  4 kB)
    fixmap   : 0xffff0000 - 0xfffe0000   ( 896 kB)
    DMA      : 0xfffa0000 - 0xffe00000   (  4 MB)
    vmalloc  : 0xe0800000 - 0xf8000000   ( 376 MB)
    lowmem   : 0xc0000000 - 0xe0000000   ( 512 MB)
    modules  : 0xbf000000 - 0xc0000000   ( 16 MB)

```

```

74         .text : 0xc0008000 - 0xc05c6000    (5880 kB)
        .init : 0xc05c6000 - 0xc05ff000    ( 228 kB)
        .data : 0xc0600000 - 0xc065e618    ( 378 kB)
76         .bss : 0xc065e63c - 0xc0699694    ( 237 kB)
NR_IRQS:396
78 IRQ: Found an INTC at 0xfa200000 (revision 5.0) with 128 interrupts
    Total of 128 interrupts on 1 active controller
80 OMAP clockevent source: GPTIMER1 at 25000000 Hz
    OMAP clocksource: GPTIMER2 at 25000000 Hz
82 sched_clock: 32 bits at 25MHz, resolution 40ns, wraps every 171798ms
    Console: colour dummy device 80x30
84 Calibrating delay loop... 718.02 BogoMIPS (lpj=3590144)
    pid_max: default: 32768 minimum: 301
86 Security Framework initialized
    Mount-cache hash table entries: 512
88 CPU: Testing write buffer coherency: ok
    devtmpfs: initialized
90 print_constraints: dummy:
    NET: Registered protocol family 16
92 GPMC revision 6.0
    OMAP GPIO hardware version 0.1
94 omap_l3_smx omap_l3_smx.0: couldn't find resource
    omap_mux_init: Add partition: #1: core, flags: 0
96 omap_i2c.1: alias fck already exists
    The board is general purpose EVM in profile 0
98 omap_hsmmc.0: alias fck already exists
    omap_hsmmc.2: alias fck already exists
100 Configure Bluetooth Enable pin...
    error setting wl12xx data
102 omap2_mcspi.1: alias fck already exists
    omap2_mcspi.2: alias fck already exists
104 bio: create slab <bio-0> at 0
    SCSI subsystem initialized
106 usbcore: registered new interface driver usbfs
    usbcore: registered new interface driver hub
108 usbcore: registered new device driver usb
    registerd cppi-dma Intr @ IRQ 17
110 Cppi41 Init Done Qmgr-base(e083a000) dma-base(e0838000)
    Cppi41 Init Done
112 omap_i2c omap_i2c.1: bus 1 rev4.0 at 100 kHz
    Advanced Linux Sound Architecture Driver Version 1.0.24.
114 Bluetooth: Core ver 2.16
    NET: Registered protocol family 31
116 Bluetooth: HCI device and connection manager initialized
    Bluetooth: HCI socket layer initialized
118 Bluetooth: L2CAP socket layer initialized
    Bluetooth: SCO socket layer initialized
120 Switching to clocksource gp timer
    Switched to NOHz mode on CPU #0

```

```

122 musb-hdrc: version 6.0, ?dma?, otg (peripheral+host)
musb-hdrc musb-hdrc.0: dma type: dma-cppi41
124 musb-hdrc musb-hdrc.0: USB OTG mode controller at e080a000 using DMA,
    IRQ 18
musb-hdrc musb-hdrc.1: dma type: dma-cppi41
126 musb-hdrc musb-hdrc.1: USB OTG mode controller at e080c800 using DMA,
    IRQ 19
NET: Registered protocol family 2
128 IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
TCP established hash table entries: 16384 (order: 5, 131072 bytes)
130 TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
132 TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
134 UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
136 RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
138 RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
140 NetWinder Floating Point Emulator V0.97 (double precision)
VFS: Disk quotas dquot_6.5.2
142 Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
JFFS2 version 2.2. (NAND) (SUMMARY) \0xc2\0xa9 2001-2006 Red Hat,
    Inc.
144 msgmni has been set to 1001
io scheduler noop registered
146 io scheduler deadline registered
io scheduler cfq registered (default)
148 Could not set LED4 to fully on
da8xx_lcd dc da8xx_lcd.0: GLCD: Found AT043TN24 panel
150 Console: switching to colour frame buffer device 60x34
Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
152 omap_uart.0: ttyO0 at MMIO 0x44e09000 (irq = 72) is a OMAP UART0
console [ttyO0] enabled
154 omap_uart.1: ttyO1 at MMIO 0x48022000 (irq = 73) is a OMAP UART1
omap_uart.2: ttyO2 at MMIO 0x48024000 (irq = 74) is a OMAP UART2
156 omap_uart.3: ttyO3 at MMIO 0x481a6000 (irq = 44) is a OMAP UART3
omap_uart.4: ttyO4 at MMIO 0x481a8000 (irq = 45) is a OMAP UART4
158 omap_uart.5: ttyO5 at MMIO 0x481aa000 (irq = 46) is a OMAP UART5
brd: module loaded
160 loop: module loaded
i2c-core: driver [tsl2550] using legacy suspend method
162 i2c-core: driver [tsl2550] using legacy resume method
mtdoops: mtd device (mtddev=name/number) must be supplied
164 omap2-nand driver initializing
ONFI flash detected
166 ONFI param page 0 valid

```

```

NAND device: Manufacturer ID: 0xad, Chip ID: 0xdc (Hynix H27U4G8F2DTR
-BC)
168 Creating 8 MID partitions on "omap2-nand.0":
0x000000000000-0x000000020000 : "SPL"
170 0x000000020000-0x000000040000 : "SPL.backup1"
0x000000040000-0x000000060000 : "SPL.backup2"
172 0x000000060000-0x000000080000 : "SPL.backup3"
0x000000080000-0x000000260000 : "U-Boot"
174 0x000000260000-0x000000280000 : "U-Boot Env"
0x000000280000-0x000000780000 : "Kernel"
176 0x000000780000-0x000020000000 : "File System"
OneNAND driver initializing
178 davinci_mdio davinci_mdio.0: davinci mdio revision 1.6
davinci_mdio davinci_mdio.0: detected phy mask ffffffff
180 davinci_mdio.0: probed
davinci_mdio davinci_mdio.0: phy[4]: device 0:04, driver unknown
182 CAN device driver interface
CAN bus driver for Bosch D_CAN controller 1.0
184 d_can d_can: d_can device registered (irq=55, irq_obj=56)
usbcore: registered new interface driver cdc_ether
186 usbcore: registered new interface driver cdc_subset
Initializing USB Mass Storage driver...
188 usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
190 gadget: using random self ethernet address
gadget: using random host ethernet address
192 usb0: MAC 06:50:55:8c:0c:93
usb0: HOST MAC 62:9a:93:a2:1e:53
194 gadget: Ethernet Gadget, version: Memorial Day 2008
gadget: g_ether ready
196 musb-hdrc musb-hdrc.0: MUSB HDRC host driver
musb-hdrc musb-hdrc.0: new USB bus registered, assigned bus number 1
198 usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
200 usb usb1: Product: MUSB HDRC host driver
usb usb1: Manufacturer: Linux 3.1.0 musb-hcd
202 usb usb1: SerialNumber: musb-hdrc.0
hub 1-0:1.0: USB hub found
204 hub 1-0:1.0: 1 port detected
musb-hdrc musb-hdrc.1: MUSB HDRC host driver
206 musb-hdrc musb-hdrc.1: new USB bus registered, assigned bus number 2
usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
208 usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
usb usb2: Product: MUSB HDRC host driver
210 usb usb2: Manufacturer: Linux 3.1.0 musb-hcd
usb usb2: SerialNumber: musb-hdrc.1
212 hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
214 mousedev: PS/2 mouse device common for all mice

```



```

input: ti-tsc-adcc as /devices/platform/tsc/input/input0
omap_rtc omap_rtc: rtc core: registered omap_rtc as rtc0
i2c /dev entries driver
Linux video capture interface: v2.00
usbcore: registered new interface driver uvcvideo
USB Video Class driver (1.1.1)
OMAP Watchdog Timer Rev 0x01: initial timeout 60 sec
Bluetooth: HCI UART driver ver 2.2
Bluetooth: HCI H4 protocol initialized
Bluetooth: HCI BCSP protocol initialized
Bluetooth: HCILL protocol initialized
Bluetooth: HCIATH3K protocol initialized
cpuidle: using governor ladder
cpuidle: using governor menu
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
usbcore: registered new interface driver snd-usb-audio
_regulator_get: 1-000a supply VDDA not found, using dummy regulator
_regulator_get: 1-000a supply VDDIO not found, using dummy regulator
_regulator_get: 1-000a supply VDDD not found, using dummy regulator
sgtl5000 1-000a: sgtl5000 revision 17
print_constraints: 1-000a: 850 <=> 1600 mV at 1200 mV normal
_regulator_get: 1-000a supply VDDA not found, using dummy regulator
_regulator_get: 1-000a supply VDDIO not found, using dummy regulator
sgtl5000 1-000a: Using internal LDO instead of VDDD
mmcl: card claims to support voltages below the defined range. These
will be ignored.
sgtl5000 1-000a: Failed to add route HPLOUT->Headphone Jack
sgtl5000 1-000a: dapm: unknown pin MONO_LOUT
sgtl5000 1-000a: dapm: unknown pin HPLCOM
sgtl5000 1-000a: dapm: unknown pin HPRCOM
asoc: sgtl5000 <-> davinci-mcasp.0 mapping ok
ALSA device list:
#0: AM335X EVM
oprofile: hardware counters not available
oprofile: using timer interrupt.
nf_contrack version 0.5.0 (8015 buckets, 32060 max)
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP cubic registered
NET: Registered protocol family 17
can: controller area network core (rev 20090105 abi 8)
NET: Registered protocol family 29
can: raw protocol (rev 20090105)
can: broadcast manager protocol (rev 20090105 t)
Bluetooth: RFCOMM TTY layer initialized
Bluetooth: RFCOMM socket layer initialized
Bluetooth: RFCOMM ver 1.11
Bluetooth: BNEP (Ethernet Emulation) ver 1.3
Bluetooth: BNEP filters: protocol multicast

```

```

Bluetooth: HIDP (Human Interface Emulation) ver 1.2
264 Registering the dns_resolver key type
VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev
3
266 ThumbEE CPU extension supported.
_regulator_get: mpu.0 supply mpu not found, using dummy regulator
268 omap2_set_init_voltage: Fail set voltage-dpll_mpu_ck(f=720000000 v
=1260000)on vddmpu
omap2_set_init_voltage: unable to set vdd_mpu
270 Detected MACID=0:18:30:fe:7b:f6
input: gpio-keys as /devices/platform/gpio-keys/input/input1
272 omap_rtc omap_rtc: setting system clock to 2000-01-01 00:00:00 UTC
(946684800)
mmc1: queuing unknown CIS tuple 0x91 (3 bytes)
274
CPSW phy found : id is : 0x4dd072
276 PHY 0:01 not found
mmc1: new SDIO card at address 0001
278 PHY: 0:04 - Link is Up - 100/Full
Sending DHCP requests ., OK
280 IP-Config: Got DHCP answer from 192.168.1.1, my address is
192.168.1.6
IP-Config: Complete:
282 device=eth0, addr=192.168.1.6, mask=255.255.255.0, gw
=255.255.255.255,
host=192.168.1.6, domain=, nis-domain=(none),
284 bootserver=192.168.1.1, rootserver=192.168.1.1, rootpath=/
tftpboot/rootfs
VFS: Mounted root (nfs filesystem) on device 0:15.
286 devtmpfs: mounted
Freeing init memory: 228K
288
INIT: version 2.88 booting
290
Starting udev
292 udevd[718]: starting version 182
bootlogd: cannot allocate pseudo tty: No such file or directory
294 Populating dev cache
Fri Apr 14 13:12:58 UTC 2017
296
INIT: Entering runlevel: 5
298
Configuring network interfaces... ifup skipped for nfsroot interface
eth0
300 run-parts: /etc/network/if-pre-up.d/nfsroot exited with code 1
Starting system message bus: dbus.
302 Starting Dropbear SSH server: Generating key, this may take a while
...
Public key portion is:

```

```

304 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDHQW67zEKPRleKx6VZxLER0R/2HThSN
/
SWD3fMk23eXy9j3PudyVMJfjGBF258qnZNicoMsK0Mx5JrpV124XKCzvKTAYsMjLc67WdqX73zSzt1Cpl
+Dq16Nld2ZuhfGDidLPvrSqOfRaUYRH6048XV/
E7penoQ8oP2tJV0kiGnXRQMoqSyLyZFWW/0xNzatB/Wr6o+
I7Iboc2KWDyOu8caJxP4fxrV/4zEjyTvSQCBCzwuv2RSFPJV7lleMD2XkKN+
QOGCgG1Eq8TgrNDU0Jps+RelENrtkj+lgVJF2odabmPMtmsB+8
lhtgdgk8wth0dbjQzedRFt root@devkit8600
Fingerprint: md5 a7:0c:a2:b1:07:9a:77:98:01:7e:31:13:10:02:42:2c
306 dropbear.
Starting rpcbind daemon...rpcbind: cannot create socket for udp6
308
rpcbind: cannot create socket for tcp6
310
done.
312 Starting syslogd/klogd: done
* Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
314 ...done.
Starting Telephony daemon
316 Starting Linux NFC daemon
/etc/rc5.d/S64neard: line 26: /usr/lib/neard/nfc/neard: No such file
or directory
318
320 Poky (Yocto Project Reference Distro) 1.7.3 devkit8600 /dev/ttyO0
322
324 devkit8600 login: root
root@devkit8600:~#

```

Listing A.1 – Messages de sortie du terminal entre l’allumage de la cible et le prompt de login