Compte-rendu TP "Prise en main de Linux embarqué - Partie 1"

Mewen Michel et Sander Ricou - MI11 UTC

Exercice 1

Question 1.1 *Écrivez les caractéristiques de la carte.*

On trouve ces informations dans le manuel utilisateur p.9 et 10. En plus du processeur et de la mémoire vive ci-dessous, la carte possède les caractéristiques suivantes :

- Nom : `DevKit8600 evaluation board`
- Mémoire morte : 512MB NAND Flash + 176KB On-chip boot ROM
- Périphériques : port LAN, interface entrée/sortie audio, USB OTG, USB HOST, interfaces CAN, RS485, SPI, IIC, ADC, GPMC, JTAG, TF slot, serial port, TFT LCD, écran tactile et clavier.

Quelle est la référence du processeur de la carte, quelle est sa fréquence et son architecture ?

- Référence : `AM3359 Cortex A8`
- Fréquence : 720 MHz
- Architecture : `ARM 32 Bit RISC` (*Reduced Instruction Set Computer*)

Quelle est la quantité de mémoire vive sur la carte ?

Il y a 512MB de SDRAM en DDR3.

Question 1.2 *Quelles sont les différentes possibilités pour stocker le noyau et le système de fichiers nécessaires pour démarrer la carte ?*

- Boot sur un périphérique bloc standard (cf. cours slide 49) :
    - Flash
    - Carte SD
- Boot dans un ramdisk :
    - Système de fichier chargé en RAM à partir d'une archive cpio
- Cas particulier du boot sur un disque réseau :
    - Requête BOOTP (slide 65 du cours) pour connaître son adresse IP et la localisation du noyau.
    - Appel TFTP (slide 62) pour récupérer le noyau avant de le charger
    - Montage du système de fichier NFS à partir du réseau

Question 1.3 *Quelle interface de communication avez-vous utilisé pour vous connecter à la cible ? Donnez les paramètres de connexion.*

Nous nous sommes connectés à travers le port série grâce à Cutecom :

- Device: `/dev/ttyS0`

- Baudrate: 115200
- Data bits: 8
- Stop bits: 1
- Open for: reading et writing
- Line end: LF

Question 1.4 *Décrivez la séquence de démarrage.*

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

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

I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in
Profile 0 with Daughter board
NAND:  HW ECC Hamming Code selected
512 MiB
MMC:   OMAP SD/MMC: 0
Net:   cpsw
Hit any key to stop autoboot:  3 \0x08\0x08\0x08 2 \0x08\0x08\0x08 1
\0x08\0x08\0x08 0
Card did not respond to voltage select!
Booting from network ...
miiphy read id fail
link up on port 0, speed 100, full duplex
BOOTP broadcast 1
DHCP client bound to address 192.168.1.6
Using cpsw device
TFTP from server 192.168.1.1; our IP address is 192.168.1.6
Filename 'uImage'.
Load address: 0x82000000
Loading: *\0x08
TFTP error: 'File not found' (1)
Not retrying...
Wrong Image Format for bootm command
ERROR: can't get kernel image!
```

Que se passe-t-il et pourquoi (analysez la configuration sur le PC)?

Au lancement, le bootloader initialise la carte (I2C, RAM, réseau) puis essaie de booter depuis le réseau :

```
Booting from network
...
link up on port 0
BOOTP broadcast 1
```

Notre VM contenant un serveur DHCP et étant relié *via* le switch, elle donne une IP à la cible à son démarrage (lorsqu'elle fait une requête BOOTP):

```
DHCP client bound to address 192.168.1.6
TFTP from server 192.168.1.1; our IP address is 192.168.1.6
Filename 'uImage'.
```

Quel fichier est manquant et où faudrait-il le mettre ? A quoi sert ce fichier ?

Il manque le fichier du noyau uImage (`TFTP error: 'File not found'`) qui se doit se trouver dans `/tftpboot`:

`/etc/dhcp/dhcpd.conf` :

```
allow bootp;
ddns-update-style none;

subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.1.2 192.168.1.200;
        filename "uImage";
        option root-path "/tftpboot/rootfs";
}
```

Question 1.5 *Si le fichier manquant était présent, que se passerait-t-il ensuite ?*

Si le fichier était présent, la cible chargerait son noyau grace à TFTP pour le décompresser et booter dessus.

Quel serait le problème suivant ?

Il faudrait que la cible trouve le système de fichier sur le serveur NFS. La configuration de celui-ci est correcte sur notre VM mais le dossier contenant le système de fichiers de la cible (`/tftpboot/rootfs/`) est vide.

Exercice 2

Question 2.1 *Analysez brièvement le contenu des dossiers suivants :*

- `/home/mi11/poky/build/conf` : configuration de poky
    - `bblayers.conf` : configuration des couches
    - `local.conf`
    - `sanity_info`
    - `templateconf.cfg`
- `/home/mi11/devkit8600/meta-devkit8600` : couche spécifique à la cible pour poky
    - `conf`
    - `recipes-core`
    - `recipes-extra`
    - `recipes-kernel` : contient un script pour que Poky récupère une image linux depuis le SVN de l'UTC
    - `recipes-support`
    - `recipes-xenomai`

Nous n'avons pas eu le temps d'étudier plus en détail le contenu de ces dossier.

Question 2.2 *Qu'avez vous changé dans les fichiers de configuration afin de supporter la devkit8600 ?*

- /home/mi11/poky/build/conf/bblayers.conf : ajout du bblayer de notre devkit /home/mi11/devkit8600/meta-devkit8600
- /home/mi11/poky/build/conf/local.conf : changer la machine pour laquelle poky va compiler en devkit8600 (même nom que ~/devkit8600/meta-devkit8600/conf/machine/devkit8600.conf)

On a ensuite pu compiler avec les commandes suivantes :

```
cd /home/mi11/poky
source ../poky-dizzy-12.0.3/oe-init-build-env
bitbake core-image-base
```

Question 2.3 *Analysez le résultat de la compilation se trouvant dans le répertoire /home/mi11/poky/build/tmp/deploy/images.*

ls -lah:

```
total 78M
drwxr-xr-x 2 mi11 mi11 4,0K avril 10 18:29 .
drwxr-xr-x 3 mi11 mi11 4,0K avril 10 18:20 ..
-rw-r--r-- 1 mi11 mi11 5,4K avril 10 18:22 core-image-base-devkit8600-
20170410161127.rootfs.manifest
-rw-r--r-- 1 mi11 mi11  18M avril 10 18:22 core-image-base-devkit8600-
20170410161127.rootfs.tar.bz2
-rw-r--r-- 1 mi11 mi11  30M avril 10 18:23 core-image-base-devkit8600-
20170410161127.rootfs.ubi
-rw-r--r-- 1 mi11 mi11  29M avril 10 18:23 core-image-base-devkit8600-
20170410161127.rootfs.ubifs
lrwxrwxrwx 1 mi11 mi11   57 avril 10 18:23 core-image-base-devkit8600.manifest
-> core-image-base-devkit8600-20170410161127.rootfs.manifest
lrwxrwxrwx 1 mi11 mi11   56 avril 10 18:23 core-image-base-devkit8600.tar.bz2 -
> core-image-base-devkit8600-20170410161127.rootfs.tar.bz2
lrwxrwxrwx 1 mi11 mi11   52 avril 10 18:23 core-image-base-devkit8600.ubi ->
core-image-base-devkit8600-20170410161127.rootfs.ubi
lrwxrwxrwx 1 mi11 mi11   54 avril 10 18:23 core-image-base-devkit8600.ubifs ->
core-image-base-devkit8600-20170410161127.rootfs.ubifs
-rw-r--r-- 1 mi11 mi11  22K avril 10 18:20 modules--3.1.0-r0-devkit8600-
20170410161127.tgz
lrwxrwxrwx 1 mi11 mi11   47 avril 10 18:20 modules-devkit8600.tgz -> modules-
-3.1.0-r0-devkit8600-20170410161127.tgz
-rw-r--r-- 1 mi11 mi11  294 avril 10 18:21 README_-
_DO_NOT_DELETE_FILES_IN_THIS_DIRECTORY.txt
-rw-r--r-- 1 mi11 mi11  192 avril 10 18:23 ubinize.cfg
lrwxrwxrwx 1 mi11 mi11   46 avril 10 18:20 uImage -> uImage--3.1.0-r0-
devkit8600-20170410161127.bin
-rw-r--r-- 1 mi11 mi11 3,1M avril 10 18:20 uImage--3.1.0-r0-devkit8600-
20170410161127.bin
lrwxrwxrwx 1 mi11 mi11   46 avril 10 18:20 uImage-devkit8600.bin -> uImage-
-3.1.0-r0-devkit8600-20170410161127.bin
```

Quels sont les différents fichiers, à quoi servent-ils ?

- `*.ubi` : image mémoire flash pour la cible (non utilisées ici)
- `*.tar.bz2` : archive contenant le système de fichier de la cible
- `uImage` et `*.bin` : noyau Linux exécutable par la cible

Quelle est la taille des fichiers générés ? Comparez avec ceux de votre VM sous Linux Mint. Pourquoi y a t-il une différence ?

Le noyau pèse ici 3,1Mo et le système de fichier 29Mo, ce qui est très faible comparé à un système classique (~1Go). La différence s'explique par le fait que notre noyau ne contient que le strict nécessaire à notre cible et que le système de fichier n'inclu que le strict minimum des programmes.

Ensuite, nous compilons et installons la chaîne de compilation croisée :

```
bitbake meta-toolchain
cd /home/mi11/poky/build/tmp/deploy/sdk
sudo ./poky-glibc-x86_64-meta-toolchain-armv7a-vfp-neon-toolchain-1.7.3.sh
```

Exercice 3

Question 3.1 *Compte tenu de l'exercice 1, que faut-t-il faire pour démarrer sur le noyau et le système de fichiers que vous avez générés ?*

Il faut :

- mettre l'uImage dans `/tftpboot/`
- extraire `core-image-base-devkit8600.tar.bz2` dans `/tftpboot/rootfs`

Question 3.2 *Décrivez le processus de boot complet de la cible.*

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


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

I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
Did not find a recognized configuration, assuming General purpose EVM in
Profile 0 with Daughter board
NAND:  HW ECC Hamming Code selected
512 MiB
MMC:   OMAP SD/MMC: 0
Net:   cpsw
Hit any key to stop autoboot:  3 \0x08\0x08\0x08 2 \0x08\0x08\0x08 1
\0x08\0x08\0x08 0
Card did not respond to voltage select!
Booting from network ...
miiphy read id fail
link up on port 0, speed 100, full duplex
```

```
BOOTP broadcast 1
DHCP client bound to address 192.168.1.6
Using cpsw device
TFTP from server 192.168.1.1; our IP address is 192.168.1.6
Filename 'uImage'.
Load address: 0x82000000
Loading:
*\0x08#################################################################
\0x09 #################################################################
\0x09 #################################################################
\0x09 #################################################################
\0x09 #################################################################
\0x09 #################################################################
\0x09 #################################################################
\0x09 #################################################################
\0x09 #################################################################
\0x09 ########################################
done
Bytes transferred = 3215152 (310f30 hex)
## 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
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Linux version 3.1.0 (mi11@mi11-VirtualBox) (gcc version 4.9.1 (GCC) ) #1 Mon
Apr 10 18:19:47 CEST 2017
CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c53c7d
CPU: VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: am335xevm
Memory policy: ECC disabled, Data cache writeback
AM335X ES1.0 (neon )
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
Kernel command line: console=ttyO0,115200n8 ip=dhcp noinitrd root=/dev/nfs rw
rootwait
PID hash table entries: 2048 (order: 1, 8192 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 512MB = 512MB total
Memory: 512996k/512996k available, 11292k reserved, 0K highmem
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
    DMA     : 0xffa00000 - 0xffe00000   (   4 MB)
    vmalloc : 0xe0800000 - 0xf8000000   ( 376 MB)
```

```
    lowmem  : 0xc0000000 - 0xe0000000   ( 512 MB)
    modules : 0xbf000000 - 0xc0000000   (  16 MB)
      .text : 0xc0008000 - 0xc05c6000   (5880 kB)
      .init : 0xc05c6000 - 0xc05ff000   ( 228 kB)
      .data : 0xc0600000 - 0xc065e618   ( 378 kB)
       .bss : 0xc065e63c - 0xc0699694   ( 237 kB)
NR_IRQS:396
IRQ: Found an INTC at 0xfa200000 (revision 5.0) with 128 interrupts
Total of 128 interrupts on 1 active controller
OMAP clockevent source: GPTIMER1 at 25000000 Hz
OMAP clocksource: GPTIMER2 at 25000000 Hz
sched_clock: 32 bits at 25MHz, resolution 40ns, wraps every 171798ms
Console: colour dummy device 80x30
Calibrating delay loop... 718.02 BogoMIPS (lpj=3590144)
pid_max: default: 32768 minimum: 301
Security Framework initialized
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
devtmpfs: initialized
print_constraints: dummy:
NET: Registered protocol family 16
GPMC revision 6.0
OMAP GPIO hardware version 0.1
omap_l3_smx omap_l3_smx.0: couldn't find resource
omap_mux_init: Add partition: #1: core, flags: 0
 omap_i2c.1: alias fck already exists
The board is general purpose EVM in profile 0
 omap_hsmmc.0: alias fck already exists
 omap_hsmmc.2: alias fck already exists
Configure Bluetooth Enable pin...
error setting wl12xx data
 omap2_mcspi.1: alias fck already exists
 omap2_mcspi.2: alias fck already exists
bio: create slab <bio-0> at 0
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
registerd cppi-dma Intr @ IRQ 17
Cppi41 Init Done Qmgr-base(e083a000) dma-base(e0838000)
Cppi41 Init Done
omap_i2c omap_i2c.1: bus 1 rev4.0 at 100 kHz
Advanced Linux Sound Architecture Driver Version 1.0.24.
Bluetooth: Core ver 2.16
NET: Registered protocol family 31
Bluetooth: HCI device and connection manager initialized
Bluetooth: HCI socket layer initialized
Bluetooth: L2CAP socket layer initialized
Bluetooth: SCO socket layer initialized
Switching to clocksource gp timer
Switched to NOHz mode on CPU #0
musb-hdrc: version 6.0, ?dma?, otg (peripheral+host)
```

```
musb-hdrc musb-hdrc.0: dma type: dma-cppi41
musb-hdrc musb-hdrc.0: USB OTG mode controller at e080a000 using DMA, IRQ 18
musb-hdrc musb-hdrc.1: dma type: dma-cppi41
musb-hdrc musb-hdrc.1: USB OTG mode controller at e080c800 using DMA, IRQ 19
NET: Registered protocol family 2
IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
TCP established hash table entries: 16384 (order: 5, 131072 bytes)
TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
NetWinder Floating Point Emulator V0.97 (double precision)
VFS: Disk quotas dquot_6.5.2
Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
JFFS2 version 2.2. (NAND) (SUMMARY)  \0xc2\0xa9 2001-2006 Red Hat, Inc.
msgmni has been set to 1001
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
Could not set LED4 to fully on
da8xx_lcdc da8xx_lcdc.0: GLCD: Found AT043TN24 panel
Console: switching to colour frame buffer device 60x34
Serial: 8250/16550 driver, 4 ports, IRQ sharing enabled
omap_uart.0: ttyO0 at MMIO 0x44e09000 (irq = 72) is a OMAP UART0
console [ttyO0] enabled
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
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
omap_uart.5: ttyO5 at MMIO 0x481aa000 (irq = 46) is a OMAP UART5
brd: module loaded
loop: module loaded
i2c-core: driver [tsl2550] using legacy suspend method
i2c-core: driver [tsl2550] using legacy resume method
mtdoops: mtd device (mtddev=name/number) must be supplied
omap2-nand driver initializing
ONFI flash detected
ONFI param page 0 valid
NAND device: Manufacturer ID: 0xad, Chip ID: 0xdc (Hynix H27U4G8F2DTR-BC)
Creating 8 MTD partitions on "omap2-nand.0":
0x000000000000-0x000000020000 : "SPL"
0x000000020000-0x000000040000 : "SPL.backup1"
0x000000040000-0x000000060000 : "SPL.backup2"
0x000000060000-0x000000080000 : "SPL.backup3"
0x000000080000-0x000000260000 : "U-Boot"
0x000000260000-0x000000280000 : "U-Boot Env"
```

```
0x000000280000-0x000000780000 : "Kernel"
0x000000780000-0x000020000000 : "File System"
OneNAND driver initializing
davinci_mdio davinci_mdio.0: davinci mdio revision 1.6
davinci_mdio davinci_mdio.0: detected phy mask fffffffef
davinci_mdio.0: probed
davinci_mdio davinci_mdio.0: phy[4]: device 0:04, driver unknown
CAN device driver interface
CAN bus driver for Bosch D_CAN controller 1.0
d_can d_can: d_can device registered (irq=55, irq_obj=56)
usbcore: registered new interface driver cdc_ether
usbcore: registered new interface driver cdc_subset
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
 gadget: using random self ethernet address
 gadget: using random host ethernet address
usb0: MAC ee:91:77:7d:63:a9
usb0: HOST MAC 12:89:72:46:fd:85
 gadget: Ethernet Gadget, version: Memorial Day 2008
 gadget: g_ether ready
musb-hdrc musb-hdrc.0: MUSB HDRC host driver
musb-hdrc musb-hdrc.0: new USB bus registered, assigned bus number 1
usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
usb usb1: Product: MUSB HDRC host driver
usb usb1: Manufacturer: Linux 3.1.0 musb-hcd
usb usb1: SerialNumber: musb-hdrc.0
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
musb-hdrc musb-hdrc.1: MUSB HDRC host driver
musb-hdrc musb-hdrc.1: new USB bus registered, assigned bus number 2
usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
usb usb2: Product: MUSB HDRC host driver
usb usb2: Manufacturer: Linux 3.1.0 musb-hcd
usb usb2: SerialNumber: musb-hdrc.1
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
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
mmc1: 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_conntrack 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
Registering the dns_resolver key type
VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 3
ThumbEE CPU extension supported.
_regulator_get: mpu.0 supply mpu not found, using dummy regulator
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
Detected MACID=0:17:ea:96:2:c1
input: gpio-keys as /devices/platform/gpio-keys/input/input1
omap_rtc omap_rtc: setting system clock to 2000-01-01 00:00:00 UTC (946684800)
mmc1: queuing unknown CIS tuple 0x91 (3 bytes)

CPSW phy found : id is : 0x4dd072
PHY 0:01 not found
```

```
mmc1: new SDIO card at address 0001
PHY: 0:04 - Link is Up - 100/Full
Sending DHCP requests ., OK
IP-Config: Got DHCP answer from 192.168.1.1, my address is 192.168.1.6
IP-Config: Complete:
     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),
     bootserver=192.168.1.1, rootserver=192.168.1.1, rootpath=/tftpboot/rootfs
VFS: Mounted root (nfs filesystem) on device 0:15.
devtmpfs: mounted
Freeing init memory: 228K

INIT: version 2.88 booting

Starting udev
udevd[718]: starting version 182
bootlogd: cannot allocate pseudo tty: No such file or directory
Populating dev cache
Mon Apr 10 16:22:53 UTC 2017

INIT: Entering runlevel: 5

Configuring network interfaces... ifup skipped for nfsroot interface eth0
run-parts: /etc/network/if-pre-up.d/nfsroot exited with code 1
Starting system message bus: dbus.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDWosDEc30tXCkA03IhSIad3JdjY9pDEVcNtGNeJ4WNM62ZAsZ
Y6Mlt7fddK2tyfDOnLMZmWDPnifyX/IJ792eEiSzo0soevDhxVuUM/QQUpC7/5t4VgoZywr0t+XN8wd
gE/Ix23C3BDrJQA0jxNHrYFExPIEvepeR0GIPQUArGDbIauF48SQQQAwZuy+qIMCWlOpKgHrRRrlKtu
+EaD9ehFwyWKmjiUkUg9F9L9RfQnku5en8VU1vkZAbhbf1vW6ceZdiNfzhu/4Ij4ZHIgjiunNBYXeRl
X1mFT0XKPCSaxpaIjgJxWC6DjDNoEVuqIJm9kxaB93ksuZvTJsg0Q13v root@devkit8600
Fingerprint: md5 79:75:f3:ca:52:de:ef:fd:5a:60:4d:d7:6f:e3:dc:f7
dropbear.
Starting rpcbind daemon...rpcbind: cannot create socket for udp6

rpcbind: cannot create socket for tcp6

done.
Starting syslogd/klogd: done
 * Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
   ...done.
Starting Telephony daemon
Starting Linux NFC daemon
/etc/rc5.d/S64neard: line 26: /usr/lib/neard/nfc/neard: No such file or
directory


Poky (Yocto Project Reference Distro) 1.7.3 devkit8600 /dev/ttyO0
```

Quelles sont les applications démarrées et dans quel ordre ?

- téléchargement du noyau à l'adresse mémoire `0x82000000`
- vérification et lancement du noyau
- vérification du système et lancement de la console
- paramétrage du matériel (mémoire, GPIO, Bluetooth, USB, NAND)
- envoi d'une requête DHCP, récupération de la même adresse IP, paramétrage de l'interface réseau et du serveur SSH

Question 3.3 *Quelle est l'ip de la cible ? Comment l'avez vous trouvée ?*

Son IP est `192.168.1.6` car on connaît la réponse à la requête BOOTP au démarrage de la cible (et à la requête DHCP au lancement de Linux) :

```
DHCP client bound to address 192.168.1.6
```

Question 3.4

Que contient le fichier `/proc/devices` ?

```
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 ttySDIO
253 ttyO
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 mmc
```

Quelle est la différence entre les 2 sections que l'on trouve dans ce fichier ?

- `Character devices` : composants de communication

- Block devices : espaces de stockage

Identifiez le périphérique correspondant au port série, mettez le en évidence dans le fichier /proc/devices et affichez la(es) ligne(s) correspondantes dans le dossier /dev.

Le port série correspond à :

```
/dev/console
/dev/ttyO0
```

car
cat /proc/cmdline :

```
console=ttyO0,115200n8 ip=dhcp noinitrd root=/dev/nfs rw rootwait
```

cat /etc/inittab :

```
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $

# The default runlevel.
id:5:initdefault:

# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
# Normally not reached, but fallthrough in case of emergency.
z6:6:respawn:/sbin/sulogin
00:12345:respawn:/sbin/getty -L 115200 tty00
# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
#
# Format:
#  <id>:<runlevels>:<action>:<process>
#

1:2345:respawn:/sbin/getty 38400 tty1
```

A quoi correspond le numéro que l'on trouve en début de ligne dans le fichier /proc/devices ?

Ce numéro correspond au majeur. Selon le cours, le numéro majeur correspond à un type de périphérique (un numéro par driver) et le numéro mineur correspond à un périphérique particulier parmi plusieurs de même type (les numéros mineurs sont donc gérés par un même driver).

On peut retrouver ces majeurs et mineurs dans /dev :
`ls -l /dev`

Question 3.5 *Analysez le contenu de /bin, /sbin, /usr/bin et /usr/sbin. Qu'y a t-il de particulier ? Quel est l'avantage ?*

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

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

```
ls /usr/bin/
[                       dbus-monitor            get_device              md5sum
passwd                  seq                     update-alternatives
[[                      dbus-run-session        get_driver              mesg
passwd.shadow           sexp-conv               uptime
ar                      dbus-send               get_module              mesg.sysvinit
patch                   sg                      usb-devices
awk                     dbus-uuidgen            gpasswd                 microcom
pkcs1-conv              sha3sum                 usbhid-dump
basename                dc                      groups                  mkfifo
printf                  sort                    users
bashbug                 deallocvt               groups.shadow           mpicalc
psplash                 ssh                     utmpdump
bunzip2                 dfutool                 hcitool                 nc
psplash-default         stress                  utmpdump.sysvinit
bzcat                   diff                    head                    nettle-hash
psplash-write           strings                 vlock
chage                   dirname                 hexdump                 nettle-lfib-
stream    python                systool               wall
chfn                    dlist_test              id                      newgidmap
python-config           tail                    wall.sysvinit
chfn.shadow             du                      killall                 newgrp
python2                 tee                     wc
chsh                    dumpleases              l2ping                  newgrp.shadow
python2-config          telnet                  wget
chsh.shadow             env                     l2test                  newuidmap
python2.7               test                    which
chvt                    evtest                  last                    nfctool
python2.7-config        tftp                    who
ciptool                 expiry                  last.sysvinit           nohup
rctest                  time                    whoami
clear                   expr                    lastb                   nslookup
readlink                top                     wpa_passphrase
cmp                     faillog                 lastlog                 od
realpath                tr                      xargs
cut                     find                    less                    openssl
renice                  traceroute              yes
dbclient                flock                   logger                  openvt
reset                   tty
dbus-cleanup-sockets    free                    logname                 opkg
rfcomm                  udevadm
dbus-daemon             fuser                   lsusb                   opkg-cl
scp                     uniq
dbus-launch             gatttool                lsusb.py                opkg-key
sdptool                 unzip
```

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

Selon le dossier, on ne trouve pas les mêmes programmes (système ou non, utilisateur ou non).

Exercice 4

Question 4.1 *Analysez le contenu du dossier /home/mi11/poky/build/tmp/deploy/ipk. Comment est-il organisé ?*

```
ls /home/mi11/poky/build/tmp/deploy/ipk
all
armv7a-vfp-neon
devkit8600
x86_64-nativesdk
```

Ce dossier contient des sous dossier contenant eux-même les programmes à installer dans Poky. Selon le sous dossier, on a un paquet plus ou moins spécifique à la cible.

Que contiennent les sous dossiers ?

- `all` : ipk de dépendances uniquement
- `armv7a-vfp-neon` : packages spécifiques ARMv7a
- `devkit8600` : packages spécifiques à notre cible
- `x86_64-nativesdk` : packages nécessaire à l'hôte pour la crosscompilation

Où se trouve le noyau ?

Il se trouve dans le dossier `devkit8600`

Où se trouve libxml2 ?

La librairie se trouve elle dans `armv7a-vfp-neon` car elle n'est pas spécifique à notre cible (au contraire du noyau).

Question 4.2 *Quels sont les différents paquets relatifs à 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

On installe donc la lib : opkg install libxml2_2.9.1-r0_armv7a-vfp-neon.ipk.

Question 4.3 *Donnez au moins deux méthodes pour copier le fichier dans le système de fichiers de la cible.*

On peut copier un fichier sur la cible par les méthodes suivantes :

- copier sur l'hôte vers le filesystem stocké en local (/tftpboot/rootfs/home/root/Download par exemple)
- scp vers 192.168.1.6 : scp fichier root@192.168.1.6:/home/root/
- monter le système de fichier en local avec sftp

Question 4.4 *Où se trouvent les fichiers de libxml2 installés par le gestionnaire de paquets. Quels sont ces fichiers ?*

Ces fichiers se trouvent dans /usr/lib/.

```
ls /usr/lib | grep xml
libxml2.so.2
libxml2.so.2.9.1
```

Exercice 5

Question 5.1 *À l'aide de la documentation de la carte, comment peut on accéder aux LEDs en mode utilisateur ? Vérifiez cette fonctionnalité sur la cible.*

Selon le manuel chap. 3.8.1.1 p. 72 il faut exécuter echo 1 > /sys/class/leds/user_led/brightness ce qui ne fonctionne pas pour le moment car le noyaux ne supporte pas les LEDs.

```
ls /sys/class/leds/
sys_led   user_led
```

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

Question 5.2 *À quoi sert ce fichier ? Quel est le préfixe de la chaîne de compilation croisée ?*

Ce fichier permet de définir des variables d'environnement pour permettre la compilation croisée. Il définit notamment les commandes de compilation préfixées par arm-poky-linux-gnueabi-.

Question 5.3 *Comment obtenir la liste des configurations pour défaut du noyau pour une architecture ARM ?*

```
make ARCH=arm help
Makefile:657: "WARNING: Appending $KCFLAGS (--
sysroot=/opt/poky/1.7.3/sysroots/armv7a-vfp-neon-poky-linux-gnueabi) from
environment to kernel $CFLAGS"
```

```
Cleaning targets:
  clean          - Remove most generated files but keep the config and
                   enough build support to build external modules
  mrproper       - Remove all generated files + config + various backup files
  distclean      - mrproper + remove editor backup and patch files

Configuration targets:
  config      - Update current config utilising a line-oriented program
  nconfig         - Update current config utilising a ncurses menu based
program
  menuconfig      - Update current config utilising a menu based program
  xconfig      - Update current config utilising a QT based front-end
  gconfig      - Update current config utilising a GTK based front-end
  oldconfig       - Update current config utilising a provided .config as base
  localmodconfig  - Update current config disabling modules not loaded
  localyesconfig  - Update current config converting local mods to core
  silentoldconfig - Same as oldconfig, but quietly, additionally update deps
  defconfig      - New config with default from ARCH supplied defconfig
  savedefconfig   - Save current config as ./defconfig (minimal config)
  allnoconfig      - New config where all options are answered with no
  allyesconfig      - New config where all options are accepted with yes
  allmodconfig      - New config selecting modules when possible
  alldefconfig    - New config with all symbols set to default
  randconfig      - New config with random answer to all options
  listnewconfig   - List new options
  oldnoconfig     - Same as silentoldconfig but set new symbols to n (unset)

Other generic targets:
  all          - Build all targets marked with [*]
* vmlinux      - Build the bare kernel
* modules      - Build all modules
  modules_install - Install all modules to INSTALL_MOD_PATH (default: /)
  firmware_install- Install all firmware to INSTALL_FW_PATH
                    (default: $(INSTALL_MOD_PATH)/lib/firmware)
  dir/            - Build all files in dir and below
  dir/file.[oisS] - Build specified target only
  dir/file.lst    - Build specified mixed source/assembly target only
                    (requires a recent binutils and recent build (System.map))
  dir/file.ko     - Build module including final link
  modules_prepare - Set up for building external modules
  tags/TAGS      - Generate tags file for editors
  cscope      - Generate cscope index
  gtags           - Generate GNU GLOBAL index
  kernelrelease      - Output the release version string
  kernelversion      - Output the version stored in Makefile
  headers_install - Install sanitised kernel headers to INSTALL_HDR_PATH
                    (default: /home/mi11/devkit8600/linux-3.1.0-
psp04.06.00.03.sdk/usr)

Static analysers
  checkstack      - Generate a list of stack hogs
  namespacecheck  - Name space analysis on compiled kernel
```

```
  versioncheck    - Sanity check on version.h usage
  includecheck    - Check for duplicate included header files
  export_report   - List the usages of all exported symbols
  headers_check   - Sanity check on exported headers
  headerdep       - Detect inclusion cycles in headers
  coccicheck      - Check with Coccinelle.

Kernel packaging:
  rpm-pkg             - Build both source and binary RPM kernel packages
  binrpm-pkg          - Build only the binary kernel package
  deb-pkg             - Build the kernel as an deb package
  tar-pkg             - Build the kernel as an uncompressed tarball
  targz-pkg           - Build the kernel as a gzip compressed tarball
  tarbz2-pkg          - Build the kernel as a bzip2 compressed tarball
  tarxz-pkg           - Build the kernel as a xz compressed tarball
  perf-tar-src-pkg    - Build perf-3.1.0.tar source tarball
  perf-targz-src-pkg  - Build perf-3.1.0.tar.gz source tarball
  perf-tarbz2-src-pkg - Build perf-3.1.0.tar.bz2 source tarball
  perf-tarxz-src-pkg  - Build perf-3.1.0.tar.xz source tarball

Documentation targets:
 Linux kernel internal documentation in different formats:
  htmldocs        - HTML
  pdfdocs         - PDF
  psdocs          - Postscript
  xmldocs         - XML DocBook
  mandocs         - man pages
  installmandocs  - install man pages generated by mandocs
  cleandocs       - clean all generated DocBook files

Architecture specific targets (arm):
* zImage        - Compressed kernel image (arch/arm/boot/zImage)
  Image         - Uncompressed kernel image (arch/arm/boot/Image)
* xipImage      - XIP kernel image, if configured (arch/arm/boot/xipImage)
  uImage        - U-Boot wrapped zImage
  bootpImage    - Combined zImage and initial RAM disk
                  (supply initrd image via make variable INITRD=<path>)
  dtbs          - Build device tree blobs for enabled boards
  install       - Install uncompressed kernel
  zinstall      - Install compressed kernel
  uinstall      - Install U-Boot wrapped compressed kernel
                  Install using (your) ~/bin/installkernel or
                  (distribution) /sbin/installkernel or
                  install to $(INSTALL_PATH) and run lilo

  acs5k_defconfig          - Build for acs5k
  acs5k_tiny_defconfig     - Build for acs5k_tiny
  afeb9260_defconfig       - Build for afeb9260
  ag5evm_defconfig         - Build for ag5evm
  am200epdkit_defconfig    - Build for am200epdkit
  am335x_evm_defconfig     - Build for am335x_evm
  ap4evb_defconfig         - Build for ap4evb
```

```
...
  cpu9260_defconfig        - Build for cpu9260
  cpu9g20_defconfig        - Build for cpu9g20
  da8xx_omapl_defconfig    - Build for da8xx_omapl
  davinci_all_defconfig    - Build for davinci_all
  devkit8600_defconfig     - Build for devkit8600
  devkit8600_tisdk_defconfig - Build for devkit8600_tisdk
  dove_defconfig           - Build for dove
  ebsa110_defconfig        - Build for ebsa110
  edb7211_defconfig        - Build for edb7211
  em_x270_defconfig        - Build for em_x270
  ep93xx_defconfig         - Build for ep93xx
...
  versatile_defconfig      - Build for versatile
  vexpress_defconfig       - Build for vexpress
  viper_defconfig          - Build for viper
  xcep_defconfig           - Build for xcep
  zeus_defconfig           - Build for zeus

  make V=0|1 [targets] 0 => quiet build (default), 1 => verbose build
  make V=2   [targets] 2 => give reason for rebuild of target
  make O=dir [targets] Locate all output files in "dir", including .config
  make C=1   [targets] Check all c source with $CHECK (sparse by default)
  make C=2   [targets] Force check of all c source with $CHECK
  make RECORDMCOUNT_WARN=1 [targets] Warn about ignored mcount sections
  make W=n   [targets] Enable extra gcc checks, n=1,2,3 where
        1: warnings which may be relevant and do not occur too often
        2: warnings which occur quite often but may still be relevant
        3: more obscure warnings, can most likely be ignored
        Multiple levels can be combined with W=12 or W=123
  make RECORDMCOUNT_WARN=1 [targets] Warn about ignored mcount sections

Execute "make" or "make all" to build all targets marked with [*]
For further info see the ./README file
```

Quelles sont les configurations par défaut possibles pour la devkit8600 ? Par la suite on ne gardera que la première.

On peut utiliser les configurations devkit8600_defconfig et devkit8600_tisdk_defconfig.

```
make ARCH=arm devkit8600_defconfig
```

Question 5.4 *Quelle est l'option à activer dans le noyau ?*

Il faut activer `Device Drivers -> LED Support -> LED Support for GPIO conected LEDs`

Quelles sont les différentes possibilités pour l'activer ?

On peut faire la modification en dur dans le noyau (y) ou en module (m) à l'exécution.

La configuration actuelle du noyau (modification de la configuration par défaut) se trouve à `~/devkit8600/linux-3.1.0-psp04.06.00.03.sdk/.config`

Question 5.5 *Où se trouve le résultat de la compilation ?*

La sortie se trouve dans ~/devkit8600/linux-3.1.0-psp04.06.00.03.sdk/arch/arm/boot/uImage

Question 5.6 *Comment vérifier dans les logs de démarrage que le noyau utilisé est bien celui qui vient d'être compilé ?*

```
dmesg | grep "Linux version"
Linux version 3.1.0 (mi11@mi11-VirtualBox) (gcc version 4.9.1 (GCC) ) #1 Mon
Apr 10 20:00:59 CEST 2017
```

Question 5.7 *Comment vérifier dans les logs de démarrage que la fonctionnalité ajoutée est bien présente ?*

```
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 HPLOUT->Headphone Jack
```

Désormais, les commandes suivantes fonctionnent :

```
echo 1 > /sys/class/leds/user_led/brightness
echo 0 > /sys/class/leds/user_led/brightness
```