Projet long 2013/2014

Mise en œuvre d'un noyau Linux configuré sur mesure Pour des cartes de développement ARMADEUS APF28

Responsable TER :

Mr. Pascal BERTHOU : berthou@laas.fr

Binôme :

Chabane safouane

Ras Walid

RESUME

Le but de ce projet est la mise en œuvre d'un noyau linux configuré sur mesure pour les cartes de développement Amadeus APF28.
Notre objectif est de configurer et mettre en œuvre la chaine de développement (cross compilation, installation) pour installer une distribution linux sur la carte, les outils seront Buildroot et Open Embedded.Notre projet TER se base sur les étapes suivantes :

- Installation des outils pour la communication avec la carte.
- Configuration de FTP.
- Manipulation application sur la carte.
- Mesure des performances de la carte.
- Configuration et Installation d'un noyau temps réel.
- Installation d'un noyau temps réel sur les cartes (Xenomai ou RTAI (de Préférence)).
- Rédaction d'un Cook-book.
- Définir une méthode d'évaluation par les E/S existantes des Temps de Commutation de l'OS temps-réel.

Abstract

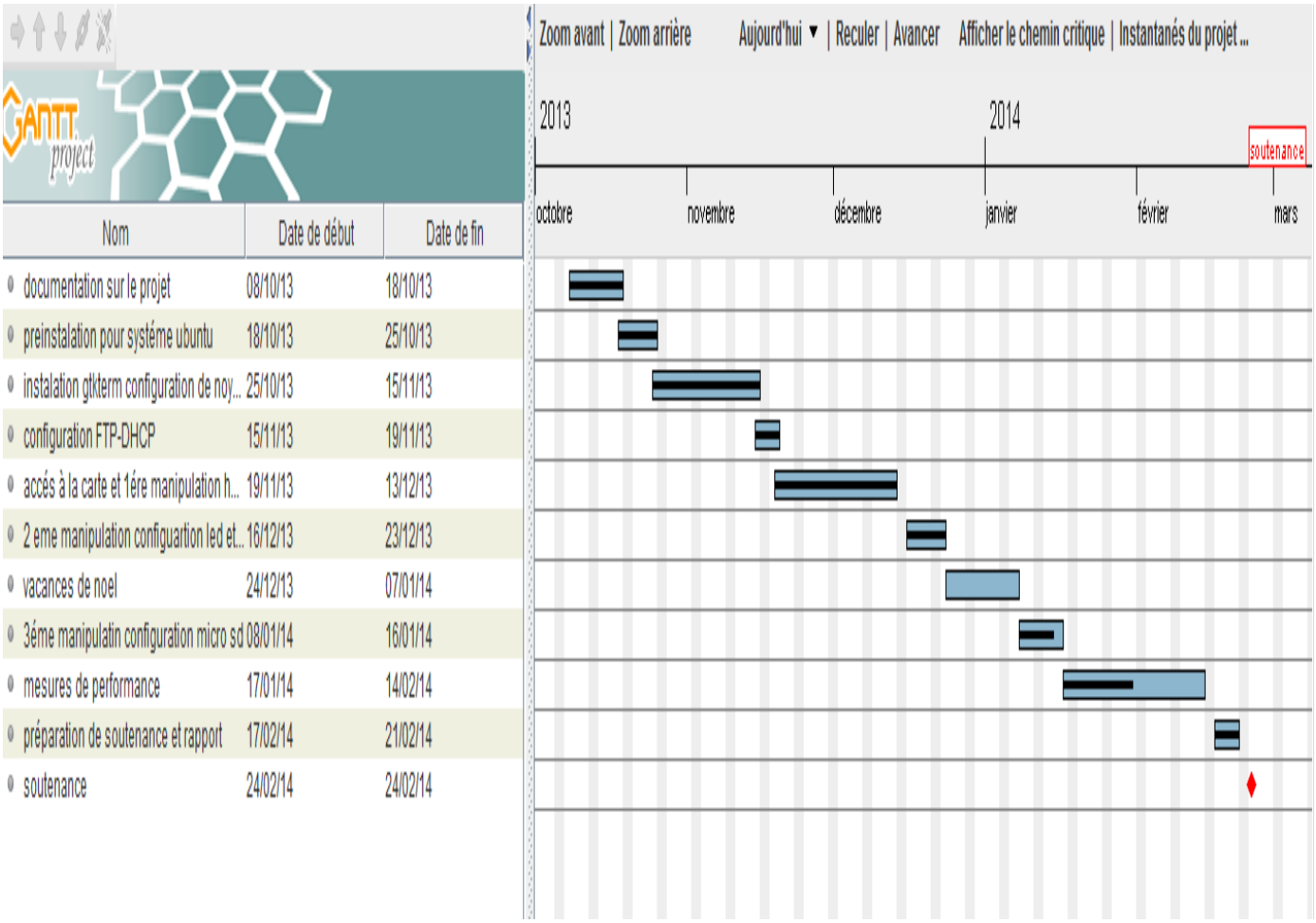
The purpose of this project is the implementation of a Linux kernel configured for custom development boards Armadeus APF28. Our goal is to set up and implement the chain of development (cross compiling, installing) to install a Linux distribution on the map tools will Buildroot and Open Embedded. TER is our eight objectives:

- Configuration and Installation of a real-time kernel.
- Installation tools for communication with the card.
- Configuring FTP.
- Manipulation some application on the card.
- Measuring performance of the card.
- Installation of a real-time kernel on the cards (or Xenomai RTAI (Preference)).
- Writing a cook-book .
- Define an evaluation method I / O existing times Switching OS real-time.

Sommaire

Plan

Le projet nous a pris un peu prés 200 heures répartis de cette façon :



Nom	Date de début	Date de fin
documentation sur le projet	08/10/13	18/10/13
preinstalation pour système ubuntu	18/10/13	25/10/13
instalation gkterm configuration de noy...	25/10/13	15/11/13
configuration FTP-DHCP	15/11/13	19/11/13
accès à la carte et 1ére manipulation h...	19/11/13	13/12/13
2 eme manipulation configuartion led et...	16/12/13	23/12/13
vacances de noel	24/12/13	07/01/14
3éme manipulatin configuration micro sd	08/01/14	16/01/14
mesures de performance	17/01/14	14/02/14
préparation de soutenance et rapport	17/02/14	21/02/14
soutenance	24/02/14	24/02/14

i)Introduction

Le but de notre projet TER est la mise en œuvre d'un noyau linux configuré sur mesure pour les cartes de développement Armadeus APF28 DEV[1].
Il s'agit d'une plateforme de développement pour l'expérimentation et la réalisation d'applications linux embarquées à l'aide des drivers permettant de commander les périphériques au-dessus de la carte, comme le microcontrôleur, les ports USB, Ethernet, Micro SD ,les leds,etc…
La procédure suivie est la suivante :
Installer un linux Ubuntu ordinaire sur la machine pour pouvoir interagir avec la carte, ensuite l'installation de ckermit,utilisé pour pouvoir accéder au root de la carte et aussi pour l'échange de fichiers entre celle-là et la machine, et aussi preempt_r,un patch qui force le linux installé à adopter un comportement temps réel à travers la réduction du temps de latence induit par les fonctionnements système. [2]
D'ailleurs un système temps réel est un comportement système qui prend en compte les contraintes temporelles et doit délivrer des résultats exacts dans des délais imposés. Par exemple, l'os test pour chaque taches qui se présente sa priorité avant de lui attribuer un tems borné pour qu'elle s'exécute sans tolérance pour les dépassements d'échéances.

II) Linux xenomai

a) Présentation xénomai

D'un point de vue logiciel, la carte Armadeus est fournie avec Linux. Le système d'exploitation tournant sur la carte est grâce un outil nommé Buildroot. Cet outil permet de construire une image qui sera ensuite installé dans la mémoire flash de la carte. La génération se faisant dans le cas présent par cross-compilation, à l'aide d'une toolchain spécifique au microprocesseur de destination. Xenomai est un noyau temps réel dur, collaborant avec le noyau Linux, dont il peut utiliser les pilotes. (Gerum 2004) Il est contenu dans le domaine à plus forte priorité, le noyau Linux résidant dans un domaine de plus faible priorité. Il prend en charge toutes les taches temps réels laissant les autres au noyau Linux. Le portage d'une application temps réel depuis un autre système d'exploitation vers Linux pose des problèmes à plusieurs niveaux :

- Les APIs des divers RTOS, bien qu'ayant les mêmes concepts, n'ont pas forcément le même vocabulaire, entraînant la réécriture d'une grande partie du code. [3]

- Certains comportements sous-jacents sont différents vis-à-vis des politiques implémentées dans l'API POSIX de Linux. Ceci impose de devoir repenser entièrement l'application en tenant compte des contraintes de Linux.[4]

- pour éviter ces problèmes, Xenomai, propose une interface générique qui peut être considérée comme une couche d'abstraction.

Xenomai implémente une version générique de l'ensemble des besoins en termes de temps réel. Les skins, se trouvant au-dessus, fournissent une implémentation des spécifications de l'interface de programmation du système d'exploitation concerné et en émulent le comportement.
Grâce à ce mécanisme, le portage d'une application vers Xenomai est relativement transparent, le skin correspondant au système d'exploitation d'origine offrant exactement le même comportement que si l'application était lancée sur celui-ci. Xenomai ne met pas en valeur une API vis-à-vis d'une autre, permettant au développeur habitué au temps réel de réaliser une application avec l'API de son choix et à un développeur Découvrant le temps réel d'utiliser l'API POSIX telle qu'il la connaît avec Linux. Le dernier point important est que Xenomai, en plus d'offrir des APIs pour le développement d'applications en espace utilisateur offre également la possibilité de développer en espace noyau, grâce au skin RTDM.

III) Carte APF28 dev

a) DESCRIPTION

La carte électronique **APF28_dev** est une plateforme de développement idéale pour l'expérimentation et la réalisation d'applications Linux embarquées 'low cost'.
Elle permet d'accéder facilement aux fonctionnalités de l'**APF28**.
L'ensemble des drivers permettant d'utiliser les périphériques présents sur la carte sont disponibles dans le BSP Armadeus.[1]

b) Alimentation

☐ Tension d'entrée 7 à 16 VDC
☐ Possibilité d'alimenter par port USB (OTG ou la console)
☐ Compatible avec une batterie Li-Po 3.7V

c) Spécificités

☐ 1 port Ethernet
☐ 1 USB OTG 2.0
☐ 1 USB Host 2.0 (High Speed)
☐ 1 USB Device (console Linux)
☐ 1 LED utilisateur
☐ 1 bouton poussoir utilisateur
☐ 1 bouton de reset
☐ 1 bouton de On/Off

d) Connecteurs

☐ 1 x Jack 2.5 mm pour l'alimentation externe
☐ 1 x Ethernet (RJ45) avec transformateur d'isolement intégré et LEDs d'état
☐ 1 x USB maître (type A)
☐ 1 x USB OTG (type mini-B)
☐ 1 x USB Device (type mini-B pour la console Linux))
☐ 2 x connecteurs Hirose pour la carte **APF28**
☐ 1 x slot micro-SD
☐ Connecteur (au pas de 2.54mm) pour CAN
☐ Connecteur (au pas de 2.54mm) pour ADC
☐ Connecteur (au pas de 2.54mm) pour JTAG
☐ Connecteur (au pas de 2.54mm) pour GPIOs (I^2C, SD/MMC, SSP, 5xUARTs, ...) compatible 3.3V
☐ Connecteur pour écran & dalle tactile (au pas de 2.54mm) compatible avec le kit**LW700AT_adapt**
☐ Connecteur pour TFT TM035KBH02
☐ Cavalier de sélection du mode de démarrage

e) Environnement

☐ Température d'utilisation : 0°C à +70°C
☐ Température de stockage: -20..85°C
☐ Humidité 5-90%

f) Mécanique

- Dimensions : 110 mm x 80 mm (4,33" x 3,15")

VI) Installation de linux

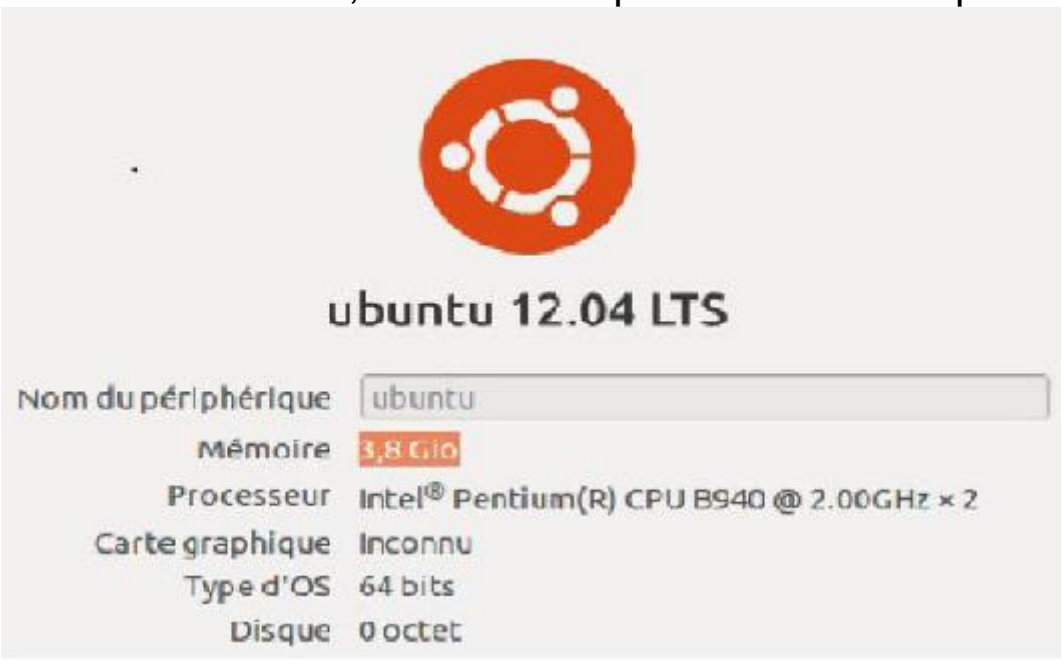
a) À propos d'Ubuntu

Ubuntu est une distribution GNU/Linux qui réunit stabilité et convivialité. Elle s'adresse aussi bien aux particuliers qu'aux professionnels, débutants ou confirmés qui souhaitent disposer d'un système d'exploitation libre et sécurisé. [2]
« Ubuntu » est un ancien mot africain qui signifie « Humanité ». Ubuntu signifie également « Je suis ce que je suis grâce à ce que nous sommes tous ». La distribution Ubuntu apporte l'esprit Ubuntu au monde logiciel.

- Le monde de l'audio et de la vidéo numérique est à vous !
- Exploitez pleinement Internet
- Tous les outils de graphisme à votre disposition
- Une suite bureautique reconnue et compatible

b) Installation ubunto

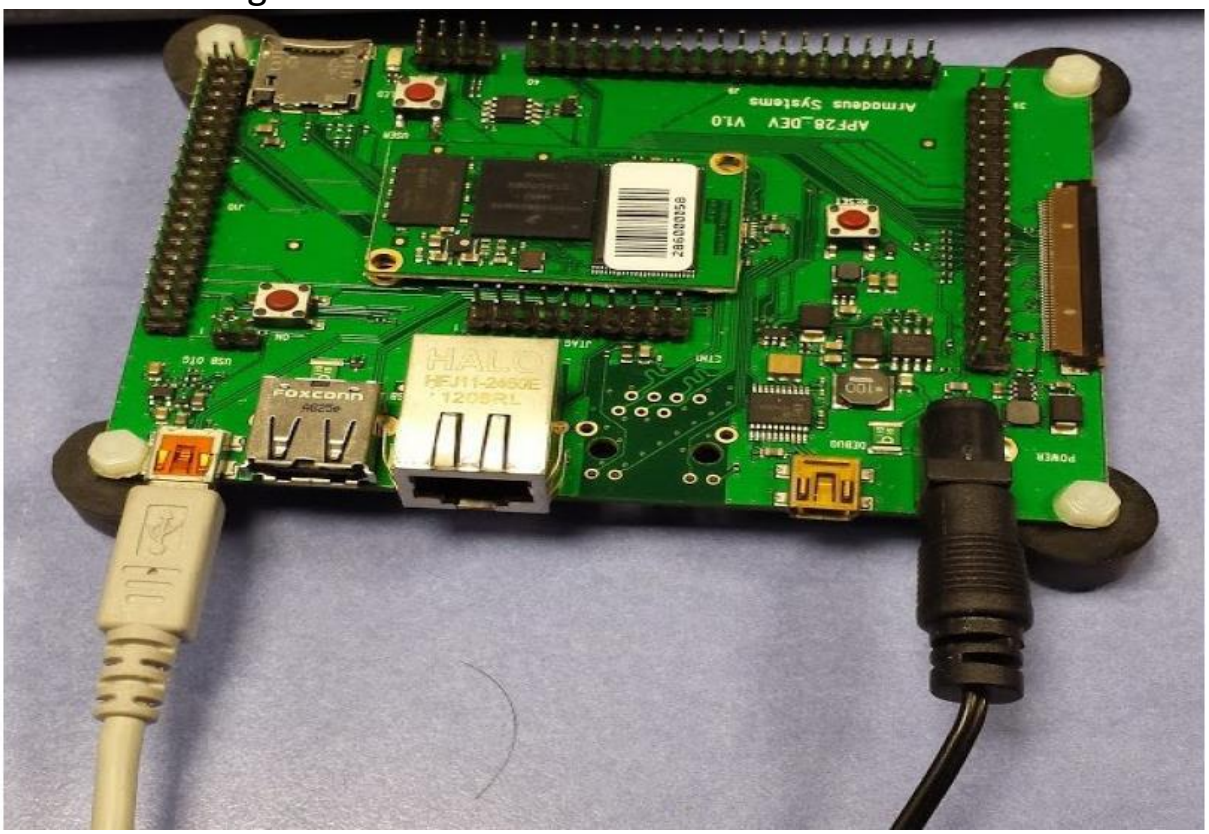
On installe le système d'exploitation Ubuntu version 12.04, malgré le fait qu'il ne soit pas vraiment un OS temps réel, mais c'est un excellent système d'exploitation pour commencer avec. Selon la nécessité, un OS RTAI pourra être utilisé par la suite.



V) Installation des outils de communication avec la carte

a) Connecter la carte correctement :

D'une part avec le Host à travers un port USB, et d'autre part avec l'alimentation grâce au câble fourni.



B) Pré-installations : outils et gtkterm

```
sudo apt-get install -y build-essential gcc g++
autoconf automake libtool bison flex gettext
sudo apt-get install -y patch subversion texinfo     wget git-core
sudo apt-get install -y libncurses5 libncurses5-dev
sudo apt-get install -y zlib1g-dev liblzo2-2 liblzo2- dev
sudo apt-get install -y libacl1 libacl1-dev gawk cvs          curl lzma
sudo apt-get install -y uuid-dev mercurial
sudo apt-get install -y python-serial python-usb
sudo apt-get -y install libglib2.0-dev
sudo apt-get -y install libnetpbm10-dev
sudo apt-get -y install python-xcbgen
sudo apt-get -y install xutils-dev
```

C)Installation du kit de développement fournit par armadeus :

On télécharge le kit

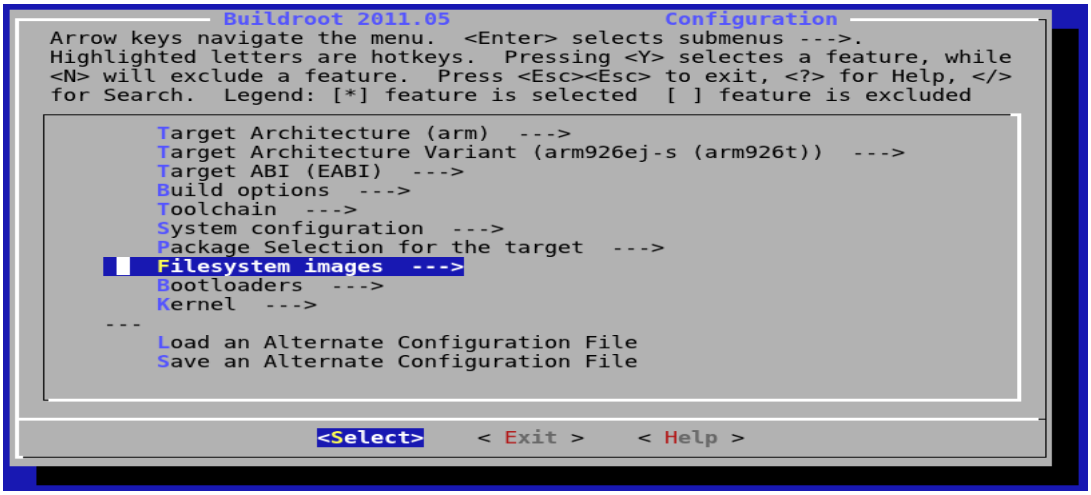
```
$ tar xjvf armadeus-5.3.tar.bz2
```

Un dossier Armadeus/ ou armadeus-5.3/
Copiez le dans votre /home/votre Nom/

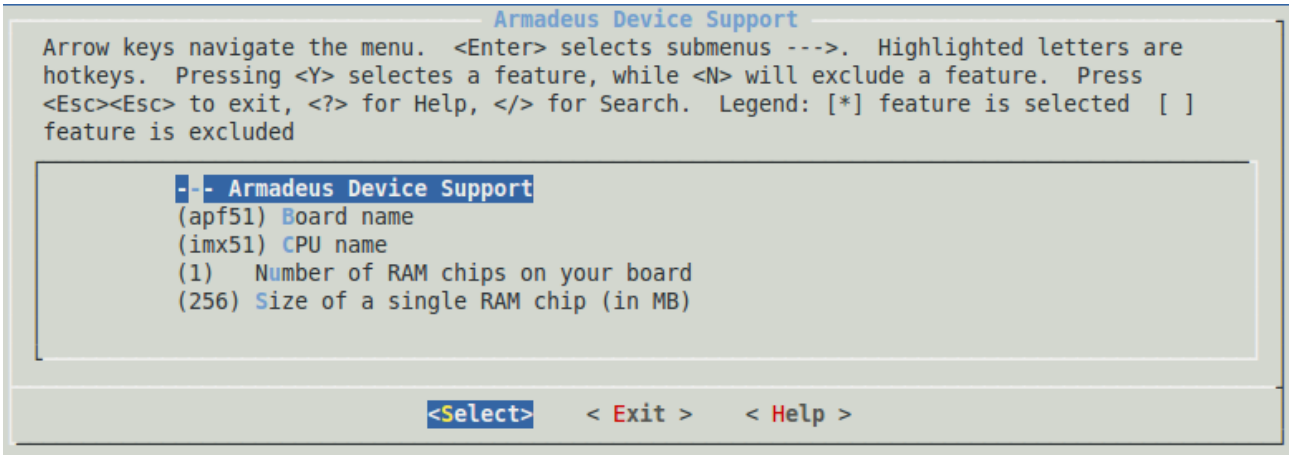
Puis faire :

```
Cd /..../armadeus-5.3/
Make ap28_deconfig
```

Cette commande charge les configurations par défaut pour l'apf28 et lance le menu de configuration de buildroot.[5]



Allez System configuration ---> [*] Armadeus Device Support --->
Et vérifiez la configuration suivante :



Sauvegarder et quittez buildroot.
Si vous voulez retourner au menu de configuration de buildroot faites: make menuconfig.[6]
Lancer la compilation par make et Si c'est la première fois que vous lancez la compilation cela peut durer un peu plus d'une heure.

Pour pouvoir communiquer et contrôler la carte Armadeus APF28 nous avons besoin d'un logiciel de connections, Armadeus nous propose trois logiciel possible :

- Kermit
- MiniCom
- GTKTerm

Dans notre cas nous choisissons d'installer **GTKTerm.**

D) Installation GTKTerm :

GTKterm est un émulateur de terminal série qui vous permet de communiquer avec la carte à travers une liaison série (RS 232, port ttyS0).c'est avec cette outil qu'on peut accéder à la console U-boot.[7]

Pour installer le logiciel (sous Ubuntu /Fedora):

```
$ sudo apt-get install gtkterm
```

E) Configuration

1. Lancez *GTKterm* : un fichier De configuration par défaut sera créé :

```
$ gtkterm
```

Dans le menu configuration on sélectionne ports et on positionne les paramètres suivants :[3][7][8]

```
Port: /dev/ttyS0    Speed:115200   Parity: None
Bits: 8             Stop bit: 1    Flow control: None
```

Cliquez sur « OK » et, depuis le même menu, sauvegardez la configuration sous le nom « armadeus ». La prochaine fois que vous lancerez *GTKterm*, n'oubliez pas de recharger cette configuration.
Vérifiez que vous avez bien les droits en lecture/écriture sur le port `/dev/ttyS0` :

```
$ ls -al /dev/ttyS0
crw-rw---- 1 root dialout 4, 64 2009-05-15 14:56 /dev/ttyS0
```

- Pour que ce soit le cas, il faut que votre UID (*user ID*) appartienne au même groupe que le port série (ici, `dialout`) :

```
$ id
uid=1000(diou) gid=1000(diou) groupes=4(adm),20(dialout),...
```

- Si `dialout` n'apparaît pas à la suite de `groupes`, vous ne pourrez pas ouvrir le fichier spécial `/dev/ttyS0`. Appelez au secours.

Après la configuration on peut bien accéder à la carte comme vous voyez dans En tapant root dans le mot de passe

```
Welcome to the Armadeus development environment.
armadeus login:
```

Notre système a booté il nous demande d'entré un nom d'utilisateur (login) afin de nous connecter. En entre root par défaut.

```
Welcome to the Armadeus development environment.
armadeus login: root
#
```

Donc on accède à la carte on vérifié bien les paramètres on peut afficher n'importe quoi sur la carte :

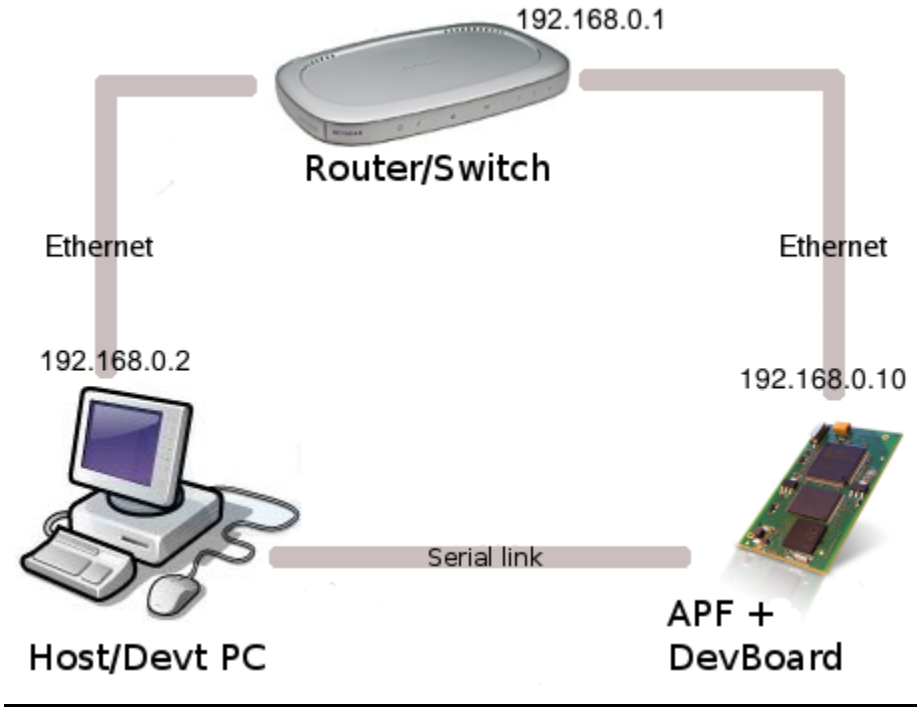
```
# df -h
Filesystem                Size      Used Available Use% Mounted on
rootfs                  249.5M     27.5M    222.0M  11% /
/dev/root               249.5M     27.5M    222.0M  11% /
none                     64.0k         0     64.0k   0% /dev
tmpfs                    61.4M     48.0k     61.4M   0% /tmp
#
```

Ou en affichant n'importe quoi :

```
# echo "Super, ça marche !"

Super, ça marche !
#
```

vi)Réglage de la FTP – DHCP

À ce stade, on a vérifié avoir quelque chose comme ceci (les adresses IP peuvent changer):



FTP : File Transfer Protocole : c'est un protocole de communication, qui utilise le réseauTCP/IP, dans un but d'échanger et de modifier les fichiers entre terminaux.

- Sur notre PC, on commence par installer « xinetd tftpd tftp », il s'agit d'un open source qui gère les connexions entre serveurs.
- On crée le fichier de configuration du protocole dans « /etc/xinetd.d/tftp .

- Ensuite on crée le dossier « tftpboot » pour contenir les fichiers à implémenter sur la carte

  $ sudo mkdir /srv/ftp

  $ sudo chmod 777 /tftpboot

- Pour tester la bonne marche du réseau FTP, on commence par brancher le câble ETHERNET, ensuite on crée un fichier '' test '' dans le fichier « tftpboot ».[3][9]
- Sur la carte on écrit :

  #tftp 192.168.1.10 # IP de mon pc

  #tftp> get test

  Sent 159 bytes in 0.0 seconds

  #tftp> quit

  #cat test

Remarque : l'adresse IP pourra être retrouvée au menu '' ifconfig → IPv4 ''. On en profite par ailleurs pour extraite d'avantage d'informations :

Adresse de broadcast : 192.168.101.255

Adresse de masque réseau : 255.255.255.0

vii)Manipulation avec la carte

1) Hello world

- Dans le répertoire « kit_armadeus/target/demos », et grâce à un éditeur, on écrit le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    printf( "APF says: Hello World ! ;-)\n" );
    return(0));
}
```

- Ensuite, on le compile avec « arm-linux-gcc » pour en extraire le 'hello.o'.
- Après ceci, on le copie dans le serveur TFTP, qui est '/tftpboot/' , pour qu'on puisse l'envoyer à la carte.
- L'extraction du fichier 'hello.c' se fait à travers les commandes suivantes :
  #tftp 192.168.1.10 // IP de notre pc
  #tftp> get hello.o
  Sent 211 bytes in 1.4 seconds
  #tftp> quit
- Finalement, on lui attribue les droits d'exécution :

```
# chmod a+x /usr/bin/hello
```

- afin de pouvoir le compiler et de l'exécuter

```
# /usr/bin/hello
APF says: Hello World ! ;-)
#
```

2)manipulation : allumer et éteindre une LED commandé le switch

a) LED

- Linux nous permets de contrôler n'importe quel ports et l'exploiter comme ressource. Pour une LED sur l'APF28, il s'agit d'un GPIO (General Purpose Input/Output = = entrée/sortie pour un usage général)[10]

- Linux a la spécificité de modifier les GPIO grâce à des drivers, qui sont plus performant qu'un code implémenté fonctionnant sur les LED.
  Il référence chaque GPIO par un chiffre précis, que l'on va convertir à un nombre entier.

- Pour l'APF28, la formule est la suivante :
  (Bank number * 32) + pin number)

Exemple : GPIO_0_21= (0*32)+21 = 21

Installation

Configurer le noyau Linux :
$ Make linux- menuconfig
Activer l'interface sysfs du GPIOlib Si ce n'est pas déjà fait :

```
Device Drivers  --->
    -*- GPIO Support  --->
        [*]   /sys/class/gpio/... (sysfs interface)
```

Afin de tester la GPIOlib avec la LED de l'utilisateur et SWITCH de la carte que vous devrez désactiver les GPIO_keys et GPIO_LEDS pilotes qui sont le conducteur ordinaire pour un tel dispositif.

```
Device Drivers  --->
    Input device support  --->
        [*]   Keyboards  --->
            < >   GPIO Buttons (be sure GPIO Buttons is unchecked)
```

```
Device Drivers  --->
    [ ] LED Support  --->  (be sure "LED Support" or "GPIO connected LEDs" (hereafter) is unchecked)
        < >   LED Support for GPIO connected LEDs
```

Enregistrez votre nouvelle configuration et reconstruire votre noyau (si nécessaire)
Make $

Et rafraichir votre noyau (si nécessaire).
Arrivant à cette étape il faut trouver le bon numéro GPIO de la LED

```
# export LED=174     (APF27Dev)
or
# export LED=2     (APF51Dev)
or
# export LED=21     (APF28Dev)
```

Poser noyau pour exporter le GPIO :

```
# echo $LED > /sys/class/gpio/export
```

Fonctions d'accès sont créées:

```
# ls /sys/class/gpio/gpio$LED/
active_low  edge         subsystem   value
direction   power        uevent
```

Vérifier le sens GPIO et mettre en production :

```
# cat /sys/class/gpio/gpio$LED/direction
in
# echo out > /sys/class/gpio/gpio$LED/direction
# cat /sys/class/gpio/gpio$LED/direction
out
```

Jouez avec valeur GPIO à clignoter LED :

```
# echo 1 > /sys/class/gpio/gpio$LED/value
# echo 0 > /sys/class/gpio/gpio$LED/value
```

b) commander le Switch

Pour commander le Switch ça se fait de même façon que la LED on trouver le numéro de GPIO connecté au Switch le seul changement donc c'est de trouver le bon numéro de GPIO .
pour APF27Dev : PF13 - > PORTF (n ° 6) broche 13 - > (6-1) * 32 + 13 - > GPIO n ° 173.
pour APF51Dev : GPIO1_3 - > Port 1 broche 3 - > GPIO n ° 3 ((1-1) * 32 + 3)
pour APF28Dev : Banque 0 , broche 17 - > GPIO n ° 17 (0 * 32 + 17)

Pour la carte APF 28 le numéro de GPIO est donné par la relation suivante :

APF28Dev: Bank 0, pin 17 -> GPIO n° 17 (0*32 + 17)[6][7]

On définit les variables ENVT (pour la portabilité) :

```
# export SWITCH=173     (APF27Dev)
or
# export SWITCH=3       (APF51Dev)
or
# export SWITCH=17      (APF28Dev)
```

Exporter le numéro de GPIO connecté aux Switch :

```
# echo $SWITCH > /sys/class/gpio/export
```

active Löw valeur du sous-système de bord :

```
# ls /sys/class/gpio/gpio$SWITCH/
active_low  edge         subsystem   value
direction   power        uevent
```

Configuration de la direction de GPIO en entré

```
# cat /sys/class/gpio/gpio$SWITCH/direction
in
```

Affectation da la valeur 1 pour active le Switch et la valeur 0 pour le désactivé

```
# cat /sys/class/gpio/gpio$SWITCH/value
1     (released)
...
# cat /sys/class/gpio/gpio$SWITCH/value
0     (pressed)
```

3)Manipulation avec la carte Micro SD

1) le choix de la carte sd

Attention, toutes les cartes ne sont pas supportés.il faut regarder deux critères pour vérifier la compatibilité.

microSD/TransFlash

Model	Status	Speed in kB/sec (read/write)				Comments
		APF9328	APF27	APF51	APF28	
Sandisk 512 MBytes	working	3330 / 320	NT / NT	NT / NT	NT / NT	VFAT
Dane-Elec 1 GBytes	working	3415 / 130	NT / NT	NT / NT	NT / NT	VFAT
Sandisk 1 GBytes	working	3355 / 130	NT / NT	NT / NT	NT / NT	VFAT
Sandisk 2 GBytes	Not recognized	NA / NA	NT / NT	NT / NT	NT / NT	NA
Kingston 1 GBytes	working	3355 / 130	NT / NT	NT / NT	NT / NT	VFAT
Kingston 2 GBytes	working	3355 / 130	NT / NT	NT / NT	13990 / 5730	VFAT
EMTEC 2 GBytes	working	3232 / 1548	NT / NT	NT / NT	NT / NT	EXT2
Kingston 4 GBytes	working	3355 / 130	NT / NT	NT / NT	NT / NT	VFAT
Transcend 4 GBytes SDHC Class 6	working	3379 / 2668	7315 / 3805	~9600 / ~2800	~15900 / ~3950	VFAT for APF9328 test EXT3 for APF27, APF51 & APF28 one

Donc ils existent deux critères à vérifier pour la carte APF28 :

- Si la carta à deux GBytes elle doit fonctionner a 1390/5730
- Si elle est de 4 GBytes et plus elle doit fonctionner a 1590/3950.

2) Formatage sous linux

Pour définir le partitionnement, on utilise les utilitaires <<gparted>> ou <<fdisk>>.

```
fdisk /dev/diskX
```

```
Command (m for help):d {enter }
Selected partition 1
Command (m for help):n {enter}
Command action
   e  extended
   p  primary partition (1-4)
p {enter}
Partition number (1-4):1 {enter}
```

Vérifiez le détail des actions qui vont être effectuées :

```
Command (m for help): p
```

```
Disk /dev/sda: 131 MB, 131072000 bytes
X heads, X sectors/track, X cylinders
Units = cylinders of X * X = X bytes

Device     Boot Start End Blocks Id System

/dev/sda1         1   X     X     X  Linux
```

Pour terminer, valider le tout

```
Command (m for help): w [write table to disk and exit]
```

Crée ensuite le système fichier en EXT2

```
mke2fs -t ext2 /dev/diskX
```

3) Montage de la carte SD

Linux (sur carte armadeus)

Par défaut les cartes micro sd sont montées dans /dev/mmcblk0

```
mount /dev/mmcblk0 /media/mmc
```

4) Environnement linux intégré

L'objectif de cette partie sera de préparer un noyau linux précompilé et personnalisé qui pourra être copié sur une carte SD.dans notre cas on a utilisé la version 6 de Debian i386 qui pourra être installé dans un environnement physique ou virtuelle.[11]

a) Armadeus software : buidroot

1.1 prérequis

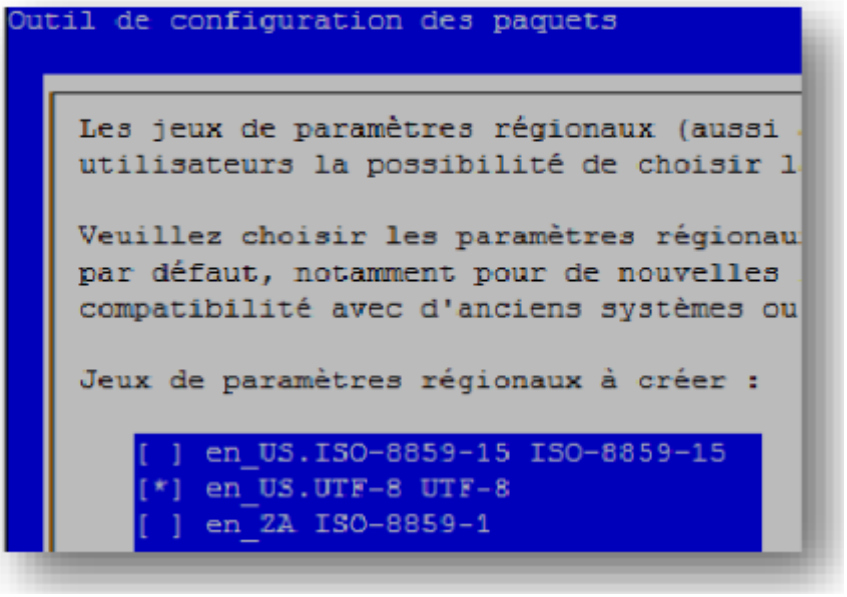
Pour pouvoir utiliser le logiciel fournit par armadeus, on installe les prérequis suivants :

```
apt-get install -y patch subversion texinfo wget git-core
apt-get install -y libncurses5 libncurses5-dev
apt-get install -y zlib1g-dev liblzo2-2 liblzo2-dev
apt-get install -y libacl1 gawk cvs curl lzma libacl1-dev
apt-get install -y uuid-dev mercurial
```

1.2 paramétre de langue

Pour instalé la distribution d'une langue différente que l'anglais on ajjoute Le support <<en_us.utf.8>>.

```
dpkg-reconfigure locales
```

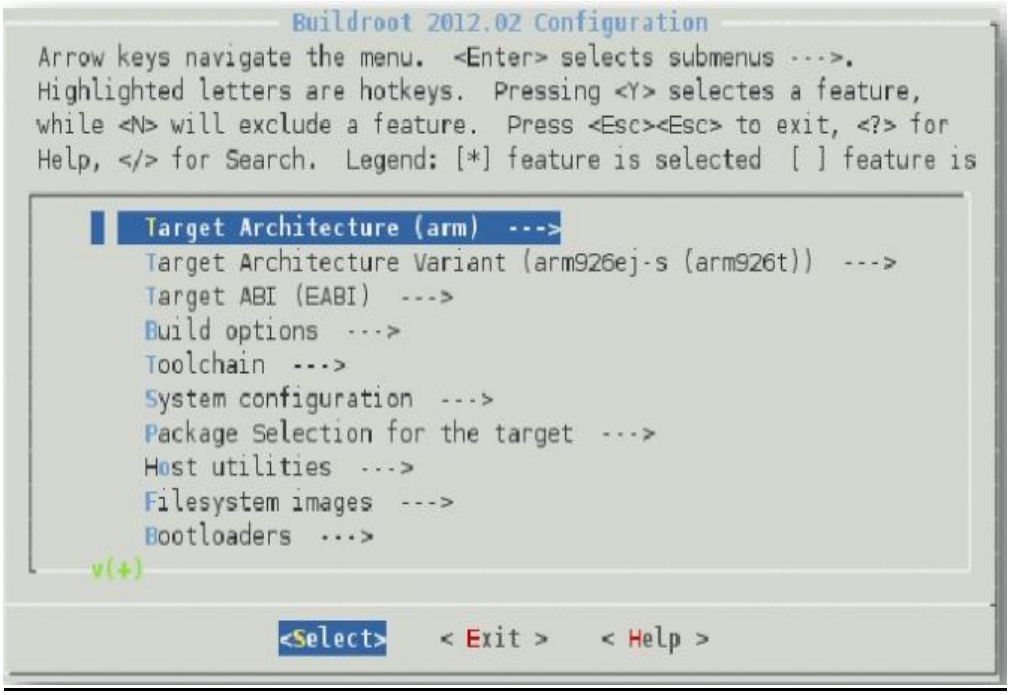


1.3 Installation

Une fois les prérequis installé on exécute le logiciel armadeus buidroot

```
tar xjvf armadeus-5.2.tar.bz2
cd armadeus-5.2
make apf28_defconfig
```

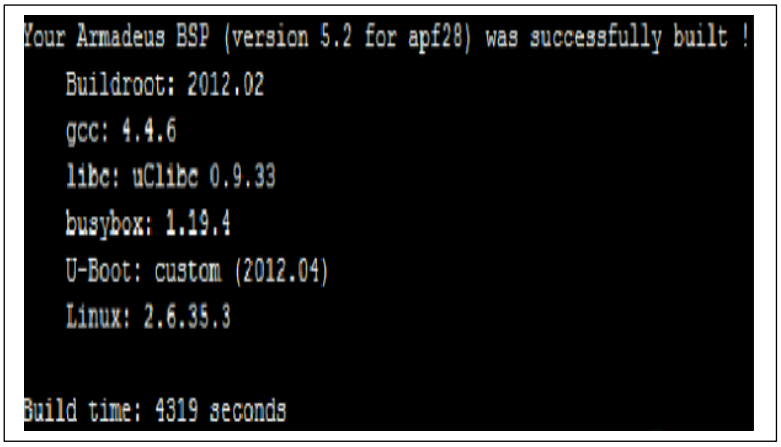
1.4 Choix des packages



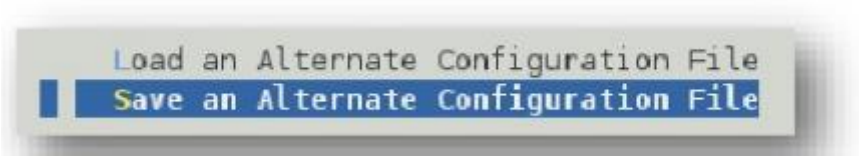
Le menu buildroot apparait on choisit le package suivant

Package choisit:

- Compressor
- Bzip
- Debugging
- Dmalloc
- Development tools
- Autoconf
- Automake
- Make
- Hardware
- USB utils



Une fois la sélection des packages est faite on enregistre dans le buildroot et on quitte le système



Enfin on pourra bien booter de la carte micro sd vers le pc et échangé les fichiers.

viii) Mesures de performances

La mesure de performance va se faire à travers le patch preempt RT installé sur notre PC. Il s'agit d'un système temps réel :

Définition d'un système à temps réel :

C'est un type d'OS multitâches qui s'applique sur les systèmes embarqués, qui facilite l'implémentation, mais ne garantit pas le fonctionnement temps réel plutôt que corriger le développement du logiciel. En général, le système adopte une attitude évènementielle : l'ordonnanceur ne change de tâche que lorsqu'un évènement de plus haute priorité a besoin de service.[11]

Les OS temps réel utilisés dans notre cas sont preempt_RT et Xenomai.

1) Mise en marche de Preempt RT

Définition de preempt RT :

Il s'agit d'un patch qui répond aux contraintes d'un système temps réel dur tout en limitant le nombre de modifications apportées. Il modifie par ailleurs certains mécanismes pour réduire les temps de latence induits par le fonctionnement du système. De plus, il met en place un mécanisme de protection contre les inversions de priorités.

Par rapport à des extensions concurrentes du noyau Linux tels que Xenomai ou RTAI, il ne fait que modifier le fonctionnement du noyau standard sans ajouter un second noyau ou une couche de virtualisation temps réel, ce qui simplifie et allège le système résultant.

Il n'ajoute aucune interface de programmation spécifique et ne requiert aucune modification d'une application existante. Un programme prévu pour fonctionner sur un noyau Linux conventionnel fonctionnera donc naturellement sur linux-rt et en tirera immédiatement certains bénéfices (temps de latence réduits) sans aucune recompilation[12].

2)Différence entre Preempt et Xenomai :

- De l'autre côté, Xenomai est une extension du noyau, qui apporte des fonctionnalités RTAI dures. Il introduit le concept de machine virtuelle en

programmation temps réel et dispose de plusieurs interfaces de programmation.[13]

Rappel :
Différence entre temps réel dur et mou :
Temps réel dur (strict): il ne tolère aucun dépassement des contraintes temporelles. À l'inverse le temps réel souple s'accommode de dépassements des contraintes temporelles dans certaines limites.
Installation du patch preempt rt :
On commence par entrer dans le répertoire '/usr/src/' afin de savoir notre version de linux installé, et puis installer le patch preempt correspondant :

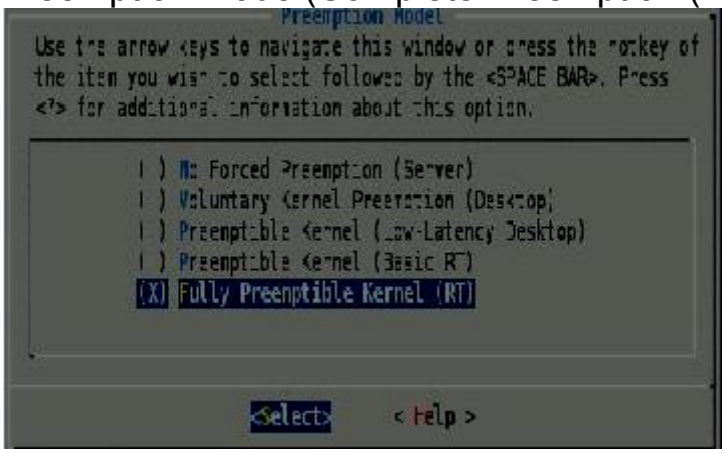
```
walid@ubuntu:~$ cd /usr/src/
walid@ubuntu:/usr/src$ ls
backfire-0.73-1                 linux-headers-3.5.0-46-generic
linux-headers-3.5.0-17          linux-headers-3.5.0-47
linux-headers-3.5.0-17-generic  linux-headers-3.5.0-47-generic
linux-headers-3.5.0-46
walid@ubuntu:/usr/src$
```

Index of /pub/linux/kernel/projects/rt

Name	Last modified	Size
Parent Directory		-
2.6.22/	08-Aug-2013 18:24	-
2.6.23/	08-Aug-2013 18:26	-
2.6.24/	08-Aug-2013 18:27	-
2.6.25/	08-Aug-2013 18:27	-
2.6.26/	08-Aug-2013 18:28	-
2.6.29/	08-Aug-2013 18:28	-
2.6.31/	08-Aug-2013 18:28	-
2.6.33/	08-Aug-2013 18:29	-
3.0/	19-Nov-2013 22:02	-
3.2/	07-Mar-2014 17:55	-
3.4/	07-Mar-2014 17:55	-
3.6/	19-Nov-2013 22:01	-
3.8/	07-Mar-2014 17:56	-
3.10/	07-Mar-2014 17:56	-
3.12/	08-Mar-2014 21:11	-

On installe la version adéquate pour notre OS Linux. On le décompresse dans le même dossier que ce dernier, puis on installe la librairie : libncurses-dev afin qu'on puisse compiler.

Ensuite on lance un make menuconfig au sein du fichier "/usr/src/patch_preempt-rt_3.6.03", on effectue les configurations suivantes :
Processor type and features —>
Preemption Mode (Complete Preemption (Real-Time))



Puis on enregistre et on quitte, se référer au manuel de résolution d'erreurs en cas de problème.[13][14]

On installe quelques modules nécessaires : $sudo make modules_install , ensuite dans le fichier du /boot/ , on lance un $ sudo make install , et finalement dans le même dossier on extrait l'image :
mkinitramfs -o initrd.img-3.6.33.7.2-rt30 3.6.33.7.2-rt30 // ces chiffres doivent être les mêmes que ceux du patch et de la version du Linux.

Après , on lance successivement les commandes :
make install & update-grub
Finalement, en redémarrant le système on lance

```
walid@ubuntu:~$ uname -v
```

Si la réponse est la suivante, ça veut dire que le patch a été installé

```
PREEMPT RT SMP Wed Jan 8 18:40:46 UTC 2014
```

3)Installer Xenomai sur la carte :

Afin d'installer un OS real time de type Xenomai sur la carte apf28, on effectue les modifications sur menu "defconfig" à partir du répertoire '/armadeus/' sur notre terminal:

```
Toolchain  --->
     Kernel Headers (Linux 2.6 (manually specified version))  --->
     (3.8) linux version
 ...
 Kernel  --->
     Kernel version (Custom version)  --->
         (3.8) Kernel version
     ...
     (40008000) load address (for 3.7+ multi-platform image)
     [*]   Device tree support
     (imx28-apf28dev) Device Tree Source file names
     ...
     Linux Kernel Extensions  --->
         [*] Adeos/Xenomai Real-time patch
         (http://download.gna.org/adeos/patches/v3.x/arm/) Adeos patch URL
         (ipipe-core-3.8-arm-1.patch) Path for Adeos patch file
 ...
 System configuration  --->
     (ttyAMA0) Port to run a getty (login prompt) on
 ...

 Package Selection for the target --->
    Real-Time --->
        [*] Xenomai Userspace
        [*]   Install testsuite
```

Ensuite on modifie quelques paramètres dans le U-boot afin qu'il prenne en compte les paramètres du nouveau type d'OS

```
buildroot/target/device/armadeus/apf28/apf28-u-boot-2013.04.h
```

et dans la ligne 153 on modifie

```
#define CONFIG_MTDMAP                    "gpmi-nand"
```

Ensuite on modifie les shells sur la carte :

```
$make busybox-menuconfig
```

et on fait les changement suivants

```
Shells  --->
        Choose your default shell (ash)  --->
    --- ash
    ---   Ash Shell Options
          ...
          [*]   Builtin getopt to parse positional parameters
```

finalement, on lance un make pour que armadeus prenne en compte le nouveau type d'OS.

```
cp -v buildroot/output/images/* /tftpboot/
```

Ensuite, on connecte la carte pour qu'on effectue la commande :
run update_all // elle nous permettra d'implémenter presque tout.
finalement, on accède au bios, en faisant reset et puis sauter le démarrage, on tombe sur le bios. Il manque juste de lancer
`BIOS> boot`

Suite à Un boot normal, on retrouve le message final :
```
Starting kernel ...
Uncompressing Linux... done, booting the kernel.
```

Par la suite on accède au Kernel afin de vérifier que Xenomai est bien installer, on exécute :
`# dmesg | grep Xeno`

Et on est censé retrouver les informations suivantes, ou presque, sur l'OS :

```
I-pipe: head domain Xenomai registered.
Xenomai: hal/arm started.
Xenomai: scheduling class idle registered.
Xenomai: scheduling class rt registered.
Xenomai: real-time nucleus v2.6.3 (Lies and Truths) loaded.
Xenomai: starting native API services.
Xenomai: starting POSIX services.
Xenomai: starting RTDM services.
```

Maintenant, afin de lancer un test bench pour essayer le nouvel environnement:
`# modprobe xeno_switchtest // installation du driver du bench test`

Après, on lance # `xeno-test`

et normalement, il nous affiche toutes les tâches, CPU, priorités qui sont en cours d'exécution.

4)Tester les performances de l'OS real time

On commence par accéder au built-in diagnostic tool, pour monter un debug du file system, à travers :

```
# mount -t sysfs nodev /sys
# mount -t debugfs nodev /sys/kernel/debug
```

l'histogramme des divers CPU qui sont en cours peuvent etre visualisé à travers

```
# ls /sys/kernel/debug/tracing/latency_hist/wakeup/CPU?
```

Ensuite on retrouve les CPU suivantes:

```
# grep -v " 0$" /sys/kernel/debug/tracing/latency_hist/wakeup*/CPU0
#Minimum latency: 0 microseconds.
#Average latency: 7 microseconds.
#Maximum latency: 39 microseconds.
#Total samples: 7069336
#There are 0 samples greater or equal than 10240 microseconds
#usecs           samples
    0            249884
    1            120023
    2            338781
    3            197834
    4            210872
    5            150366
    6             45870
    7           1053204
    8            564637
    9            273756
   10            190432
   11            483611
   12            328509
   13             44716
   14             72925
   15             59304
   16             28927
   17             10836
   18              2821
   19               543
   20               110
   21                82
   22                63
   23                61
```

Comparaison des WCL des divers terminaux:

Afin de déterminer les capacités du système, on install “Cyclictest”, l’outil spécifique qui détermine exactement les capacités d’un OS temps réel. On pourra le télécharger graphiquement à partir de la logithèque Ubuntu  .

Ensuite on télécharge les tests rt et on les compile :

```
# git clone git://git.kernel.org/pub/scm/linux/kernel/git/clrkwllms/rt-tests.git
# cd rt-tests
# make
```

Et là commence la comparaison simple du Worst case Latency entre divers terminaux :

Pour un Worst Case Latency du preempt RT installé sur mon PC, on analyse le rapport CPU du Kernel, On retrouve les résultats suivants :

```
$ cyclictest --smp -p95 -m
policy: fifo: loadavg: 0.04 0.01 0.00 1/338 31076

T: 0 (31074) P:95 I:1000 C:   4000 Min:      0 Act:   37 Avg:   31 Max:      59
T: 1 (31075) P:95 I:1500 C:   3322 Min:     10 Act:   08 Avg:   57 Max:      98
^C
```

Ce qui veut dire que WCL= 59ms et le temps moyen de réponse vaut = 31ms

Maintenant, sur la carte apf 28 avec le kernel armadeus installé de base, le WCL= 39ms et le temps moyen de réponse = 17ms

```
T: 0 ( 3431) P:99 I:1000 C: 100000 Min:      5 Act:   10 Avg:   14 Max:   39242
T: 1 ( 3432) P:98 I:1500 C:  66934 Min:      4 Act:   10 Avg:   17 Max:   39661
```

Finalement, avec le noyau Xenomai installé, les durée deviennent :

```
T: 0 ( 3407) P:99 I:1000 C: 100000 Min:      7 Act:   10 Avg:   10 Max:      18
T: 1 ( 3408) P:98 I:1500 C:  67043 Min:      7 Act:    8 Avg:   10 Max:      22
```

Ces performances sont extraites après une sorte d'"éxcitation" du système, à travers un open source efficace pour tester le worst case latency, à nommer 'Real Feel', qui crée des tâches asynchrones qui nous permettent de vraiment extraire la vrai durée et du coup savoir la réaction du système.

5)Mesurer les performances suite à une tache périodique

Afin de créer une tâche périodique sous Linux, nous utiliserons une fonction C permettant d'effectuer une attente passive d'une durée spécifiée en nanosecondes :[15]

```
extern int nanosleep (__const struct timespec *__requested_time, struct timespec *__remaining);
```

La structure représentant le temps est composée de deux champs : un pour les secondes et un pour les nanosecondes :[12]

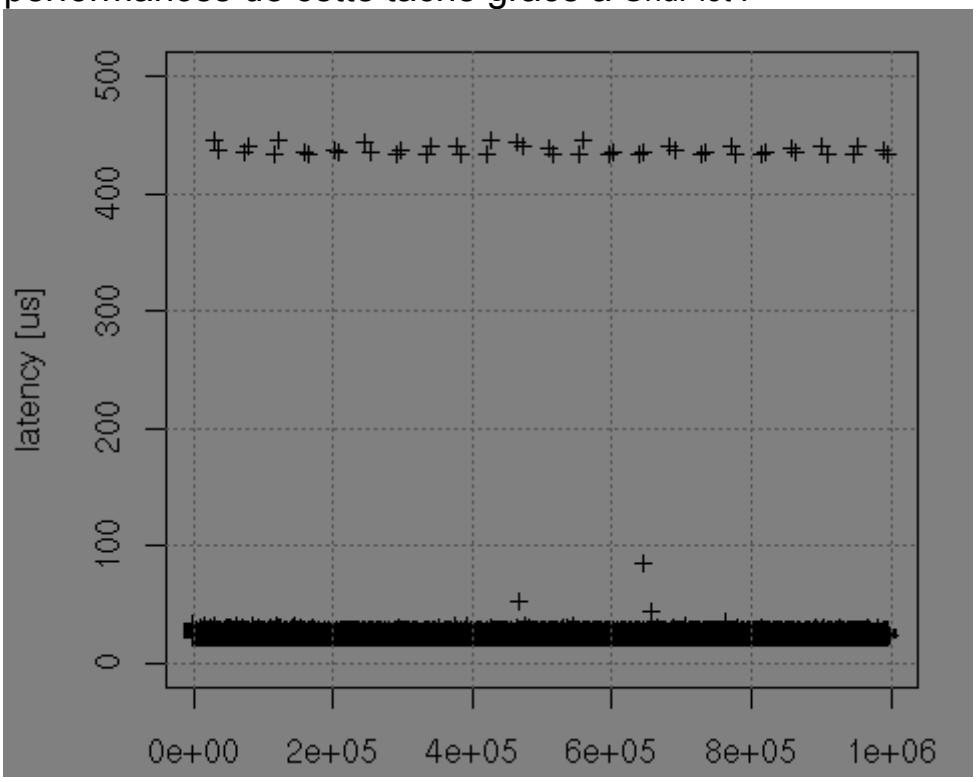
```
struct timespec {

  __time_t tv_sec; /* Seconds. */

  long int tv_nsec; /* Nanoseconds. */

};
```

Une tâche périodique sera donc de la forme :

```
for (i = 0; i < nb_iter; i++) {

  // coeur de la tâche

  nanosleep(&period, NULL);

}
```

Suite à une compilation de ce code, une implémentation sur la carte (cf. envoyer un fichier via FTP) , et une exécution, on récupère les

performances de cette tâche grâce à GnuPlot :



VII) Conclusion

En cohérence avec la problématique, et suite aux objectifs souhaités, les résultats obtenus sont les suivants:

- La mise en marche de la carte : ☺ 👍
- Édition d'un cook-book qui sert d'un tutoriel d'utilisation : ☺ 👍
- Crée de deux TP pour aider à mieux comprendre la carte : ☺ 👍
- Mesurer les performances de l'APF 28 : 😐

Comme chaque projet qui comporte la hardware et software, surtout en ce qui concerne un linux embarqué, le nombre de difficultés rencontrés est assez important :

- Bien choisir le protocole de communication entre terminal et carte, et l'actualiser à chaque connexion.
- Une difficulté liée au test d'un système sous temps réel, surtout concernant le respect des échéances : parfois on tombe sur des résultats hasardeux qui nous oblige de refaire la manipulation ou changer de méthode.
- Mise en pratique des cours du master concernant l'électronique des systèmes :
  - ❑ OS des systèmes réel.
  - ❑ gestion de projet.
  - ❑ introduction aux réseaux.
- Développement du sens d'autonomie et de polyvalence.
- Découverte de la complexité du côté opérationnel des systèmes embarqués surtout à travers la résolution au fur et à mesure des problèmes rencontrés.
- Amélioration de notre approche méthodologique vis à vis d'un projet ingénieur en respectant le cycle en V.
- Développement de notre capacité de travail en groupe.
- Une porte d'entrée pour notre recherche de stage.

X) bibliographie

[1] V. Durand, J. Birot, et R. Dupuy, « Etude et réalisation d'un driver pour Armadeus », 2013.
[2] R. Grolleau, « Configuration multilingue sous Ubuntu Linux », 2010.
[3] P. Ficheux et E. Bénard, *Linux embarqué: Nouvelle étude de cas - Traite d'OpenEmbedded*. Editions Eyrolles, 2012.
[4] J. Boukhobza, I. Khetib, et P. Olivier, « Flashmon: un outil de trace pour les accès à la mémoire flash NAND », in *Proceedings of the Embed With linux Workshop*, 2011.
[5] W. R. Nethercut, « Ille parum cauti pectoris egit opus », in *Transactions and Proceedings of the American Philological Association*, 1961, p. 389–407.
[6] L. K. Chong, C. Ballabriga, V.-T. Pham, S. Chattopadhyay, et A. Roychoudhury, « Integrated Timing Analysis of Application and Operating Systems Code », in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, 2013, p. 128–139.
[7] P. Kadionik, « La mise en oeuvre de Linux pour l'embarqué », 2004.
[8] Center for History and New Media, « Guide rapide pour débuter ». [En ligne]. Disponible sur: http://zotero.org/support/quick_start_guide.
[9] A. Marchand, « Systèmes Temps Réel embarqués: Linux pour le temps-réel et l'embarqué », *Cours Polytech'Nantes*, 2003.
[10] G. Goavec-Merou, T. de Stage, M. F. Peureux, M. de Stage, et M. J. Boibessot, « Intégration et qualification d'un systeme temps réel sur plateforme ARMadeus. »

[11] T. Knutsson, « Performance evaluation of GNU/linux for real-time applications », 2008.
[12] A. C. Heursch, A. Horstkotte, et H. Rzehak, « Preemption concepts, Rhealstone benchmark and scheduler analysis of linux 2.4 », in *Proceedings of the Real-Time & Embedded Computing Conference*, 2001.
[13] E. Landau, *Pomeranians are the Best!* Lerner Publications, 2011.
[14] W. R. Nethercut, « The ironic priest », *Am. J. Philol.*, p. 385–407, 1970.
[15] S.-T. Dietrich et D. Walker, « The evolution of real-time linux », in *7th RTL Workshop*, 2005.