

Ecole Polytechnique de l'Université de Tours

64, Avenue Jean Portalis

37200 TOURS, FRANCE

Tél. +33(0)2 47 36 14 14

Fax. +33(0)2 47 36 14 22

[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr/)

Compte-rendu du tp1

**Systèmes embarqués**

|  |  |
| --- | --- |
| Apprentis : | Enseignant : |
| Quentin Chalopin  [quentin.chalopin@etu.univ-tours.fr](mailto:quentin.chalopin@etu.univ-tours.fr) | Anthony Jouret  [anthony.jouret@univ-tours.fr](mailto:anthony.jouret@univ-tours.fr) |
| Julien Mouton  [julien.mouton@etu.univ-tours.fr](mailto:julien.mouton@etu.univ-tours.fr) |  |

Table des matières

**Tapez le titre du chapitre (niveau 1)1**

Tapez le titre du chapitre (niveau 2)2

Tapez le titre du chapitre (niveau 3)3

**Tapez le titre du chapitre (niveau 1)4**

Tapez le titre du chapitre (niveau 2)5

Tapez le titre du chapitre (niveau 3)6

# Introduction

Tout au long de ce compte-rendu, nous allons vous expliquer comment installer un système d’exploitation Linux sur une carte Armadeus **APF27** customisée selon nos envies. Par exemple, nous allons ajouter un serveur ssh ou bien encore un serveur web.

Pour réaliser cette tâche, nous allons utiliser **Buildroot.** Cet outil permet de générer facilement son « propre » Linux embarquée par l’intermédiaire de la **Cross-Compilation**.

# Matériel

Pour réaliser tout cela, il nous faudra :

* Un ordinateur doté d’un linux ou d’une machine virtuelle linux : Outil pour pouvoir compiler et envoyer les données à la carte.
* 1 carte Armadeus APF27 (celle utilisée à Polytech’) : La carte où va être implanté le système d’exploitation.
* 1 convertisseur USB 🡪 Série : Très utile si nous n’avons pas de port série directement disponible sur notre PC.
* 1 câble série : La communication série nous permettra d’interagir avec le système.
* 1 câble Ethernet : Nécessaire pour la cross-compilation via l’ordinateur.

# Mise en place

Pour mettre en œuvre tout cela, nous allons suivre le schéma suivant :

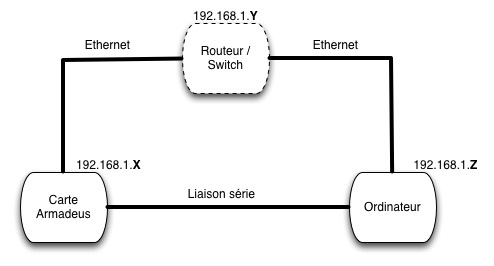


Figure Architecture

**Remarque :** Le routeur (ou switch) n’est pas nécessaire. Une connexion point à point fera l’affaire. Nous expliquerons par la suite comment configurer les adresses IP de chacun des modules.

Pour la **communication série**, aucun logiciel n’est recommandé. En ce qui nous concerne, nous allons utiliser **Hercules**.

**Remarque :** Si vous utilisez un adaptateur USB 🡪 Série, veillez à ce que vos pilotes soient bien à jour.

Si vous utilisez une machine virtuelle pour l’utilisation de Linux, veillez à ce que la connexion entre votre propre système d’exploitation et votre VM soit établie avec votre carte réseau **filaire** et non avec votre carte WI-FI (par exemple).

Lorsque tout cela est prêt, nous pouvons alors alimenter notre carte et voir l’exécution du système sur Hercules.

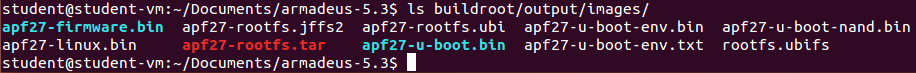
**Utile :** Un tutoriel est disponible sur le wiki d’Armadeus. Celui-ci sera très utile pour connaître les commandes importantes et la configuration de votre carte. La carte Armadeus utilisée est une APF27.

# Buildroot

L’objectif est configurer la carte afin d’inclure un serveur SSH.

Afin d’inclure un serveur SSH dans la configuration de notre carte, nous lançons l’utilitaire de configuration grâce à la commande : *make menuconfig.*

Sur le wiki de la carte Armadeus, nous avons repéré que le serveur SSH se nommait « Dropbear ». Nous l’activons donc et nous compilons avec la commande : *make*.

La sortie de la compilation se trouve dans buildroot/output/images.

# Déploiement

L’objectif est de trouver et de configurer les outils permettant de communiquer entre notre pc et la carte via la liaison Ethernet.

Afin d’envoyer les images compilées de notre bootloader, de notre noyau et de notre système de fichiers, nous avons besoin d’un serveur TFTP. On l’installe donc grâce à la commande : *sudo apt-get install tftpd xinetd*. On définit ensuite le répertoire qui contiendra les images à envoyer à la carte : /tftpboot.

Nous définissons ensuite l’architecture réseau que nous mettons en place. Pour tout le reste du tp, voici la configuration réseau.

IMAGE DE LA CONFIGURATION RESEAU

# Flash de la carte

L’objectif est de flasher la carte avec le bootloader, le noyau et le système de fichier que nous avons compilé afin de pouvoir se connecter en SSH sur la carte depuis le pc.

La première manipulation à faire est de copier les fichiers en sortie de compilation dans le repertoire /tftpboot. Il faut ensuite configurer l’environnement de la carte afin que cette dernière puisse communiquer avec le pc. Pour cela, nous devons définir l’adresse IP, le masque et l’adresse IP du serveur TFTP. Nous configurons donc la carte grâce aux commandes suivantes de manière à respecter l’architecture réseau définie précédemment : *setenv addrip 192.168.1.3*, *seten ipmask 255.255.255.0* et *setenv serverip 192.168.1.2*. A noter que ces commandes sont effectuées sur le BIOS de la carte Armadeus.

Afin de télécharger l’image du bootloader compilée sur notre pc et de flasher cette image sur la carte, il existe une commande simple : *run update\_uboot*. On effectue également cette commande dans le BIOS de la carte Armadeus. Si tout se passe correctement, on peut redémarrer la carte avec la commande : *reset*. L’environnement ayant changé, nous devons le réinitialiser : *run flash\_reset\_env*.

Nous pouvons maintenant démarrer la carte sur notre bootloader. Nous arrêtons le boot pour pouvoir reconfigurer le réseau (adresse ip de la carte, masque et adresse ip du serveur). Ceci nous permet de pouvoir télécharger le noyau et le système de fichiers compilé sur notre pc. Pour cela, nous utilisons les commandes : *run update\_kernel* et *run update\_rootfs*.

Nous pouvons maintenant redémarrer la carte sans interrompre le bootloader. La carte va donc lancer le linux que nous avons compilé ainsi que le système de fichiers. On peut voir que la génération de la clé de cryptage SSH prend du temps. Cela est dû au fait que SSH repose sur la cryptographie asymétrique. Ce sont donc les générations des clés privées et publiques qui prennent le plus de temps car elles sont générées par des évènements (clics de souris, entrées clavier, ticks processeurs,…).

Dans le cas où nous voudrions ajouter de nouveaux packages, il nous faudrait mettre à jour le noyau et le système de fichiers.

# Connexion SSH

L’objectif est de vérifier que nous arrivons bien à nous connecter sur la carte via un client SSH.

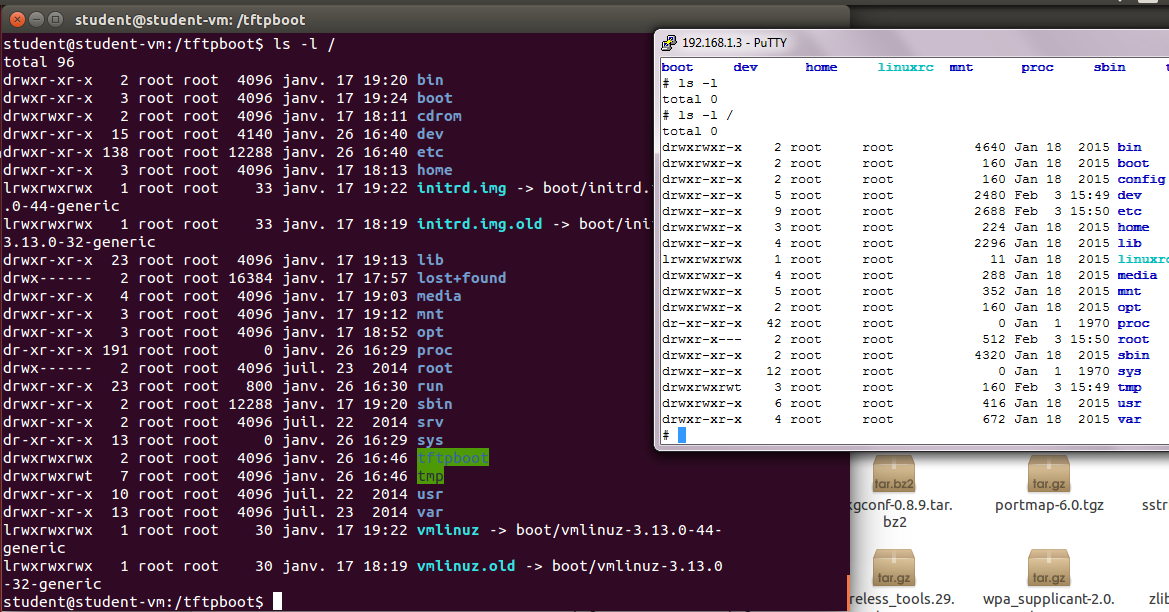
Afin de communiquer entre le pc et la carte via le protocole SSH, nous avons besoin d’un client SSH. Un des plus connus est Putty, nous choisissons donc ce logiciel.

Afin de pouvoir se connecter, nous devons commencer par configurer le mot de passe. Sur la carte, nous définissons le mot de passe de l’utilisateur root grâce à la commande : *passwd root*.

Ensuite sur le logiciel Putty, nous ouvrons une session SSH sur l’adresse IP 192.168.1.3 (adresse IP de la carte). Nous pouvons renseigner le mot de passe de l’utilisateur root afin de valider la connexion.

SCREEN DE LA CONNEXION SSH ?

En naviguant dans le système de fichier, on s’aperçoit que nous sommes bien connectés sur la carte. On valide donc la compilation du noyau et du système de fichiers avec le serveur SSH.

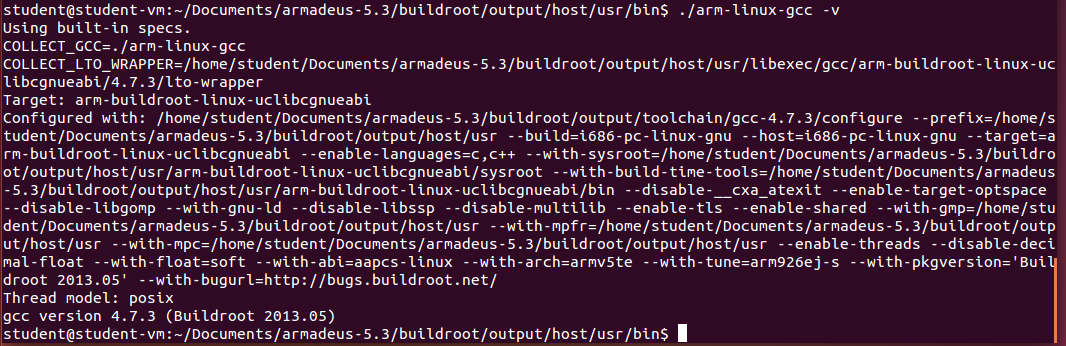


Sur l’impression écran ci-dessus, on peut voir à gauche le contenu de la racine d’Ubuntu dans la machine virtuelle et à droite le contenu de la racine de la carte. On remarque que très peu de répertoires changent. Côté machine virtuelle, les dossiers *cdrom, lost+found, run* et *srv* sont présents en plus. Côté carte, seul le dossier *config* est en plus. Ce dernier permet de configurer la carte. XXXXX peut etre incomplet ce passage.

# Hello Armadeus

L’objectif est d’utiliser la cross-compilation pour compiler un exécutable que nous enverrons sur la carte pour l’exécution.

Le gcc que nous allons utiliser pour cross-compiler n’est pas le gcc présent pas défaut sur Ubuntu car ce dernier compilerai pour une cible Ubuntu. Il faut utiliser le compilateur gcc se trouvant dans *~/Documents/armadeus-5.3/buildroot/output/host/usr/bin.*



Sur l’impression écran ci-dessus, on remarque que la cible est bien embarquée (suffixe eabi : Embedded Application Binary Interface).

On créé ensuite un simple fichier main.c qui sera chargé d’afficher « Hello World !» à son exécution (un simple print dans un main avec les bonnes bibliothèques).

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(void)

{

printf("Hello World !\n");

return 0;

}

HelloWorld.c :

Il s’agit ensuite de créer le Makefile qui va compiler ce fichier main.c en utilisant le bon compilateur gcc.

Makefile :

#

# Variables:

#

# For a desktop use

#CC = gcc

#For the Armadeus using buildroot

CC = ./../../buildroot/output/host/usr/bin/arm-linux-gcc

BIN = HelloArm

SRC = main.c

OBJ = main.o

#Include for a desktop compile

#INC=/usr/include

#Include for armadeus

INC=./../../buildroot/output/host/usr/include

#

# Flags d'erreurs:

#

ERROR\_CFLAGS = -Wall -W -pedantic

#

# Flags pour le compilateur:

#

CFLAGS = $(ERROR\_FLAGS)

#

# Flags pour l'editeur de liens:

#

LDFLAGS = $(ERROR\_FLAGS)

#

# Construction du programme:

#

all: $(BIN)

$(BIN): $(SRC)

$(CC) $(CFLAGS) -I${INC} -c $(SRC)

$(CC) $(LDFLAGS) -I${INC} -o $(BIN) $(OBJ)

#

# Nettoyage:

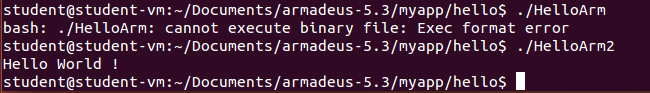
#

clean:

rm -f \*.o \*~ core $(BIN)

EXPLICATIONS DU MAKEFILE

On compile notre main.c grâce à la commande *make*. Lorsque l’on essaye d’executer la sortie de compilation, une erreur survient : « ./HelloArm: cannot execute binary file: Exec format error ». Cela est dû au fait que notre exécutable est compilé pour être exécuté sur la carte et non sur la machine virtuelle. Si on compile notre fichier main.c avec le compilateur gcc de notre machine virtuelle et que l’on exécute la sortie de compilation, on obtient : « Hello World !».



Ci-dessus l’illustration de ce que nous expliquions. HelloArm étant l’exécutable compilé pour la carte et HelloArm2 étant l’exécutable compilé pour la machine virtuelle.

# Déploiement de notre premier programme

Afin d’envoyer nos sorties de compilations sur la carte, deux moyens s’offrent à nous :

* Les télécharger grâce au serveur TFTP comme pour mettre à jour le bootloader
* Les inclure au sein du système de fichier que l’on flashera par la suite sur la carte.

Pour cette dernière méthode, il faut placer les fichiers obtenus en sortie de compilation dans un dossier de l’arborescence : *~/Documents/armadeus-5.3/buildroot/output/target.* De cette manière, lorsque nous effectuerons un make dans le dossier *~/Documents/armadeus-5.3/buildroot,* le système de fichier contiendra nos fichier là on les a placés.

En essayant d’exécuter les fichiers, on voit que seul le fichier compilé pour la cible armadeus peut s’exécuter.

SCREEN DE L’EXECUTION SUR LA CARTE

POURQUOI ?

Interet des deux méthodes ?

# Installation d’un serveur web

Le serveur web disponible sur la carte Armadeus est appelé Boa, c’est celui-là que nous implémenterons.

Afin de le configurer afin que nous puissions afficher une simple page html qui affiche « Hello Armadeus », nous avons suivi le protocole suivant.

Les commandes suivantes sont réalisées sur la machine virtuelle linux.

On commence par effectuer la commande *make shell\_env && source armadeus\_env.sh.* Cette dernière permet d’accéder au répertoire racine de la carte Armadeus grâce au simple mot clé $ARMADEUS\_ROOTFS\_DIR.

On créé un fichier vide index.html dans l’arborescence : *$ARMADEUS\_ROOTFS\_DIR/var/www.* Ce denier contient juste de quoi afficher un titre et une simple phrase. Voici son contenu :

<html>

<head>

<title> Hello Armadeus !!! </title>

</head>

<body>

<h1> Welcome on the Armadeus development board ! </h1>

</body>

</html>

On créé maintenant le répertoire de log grâce aux commandes suivante :

*ls -l $ARMADEUS\_ROOTFS\_DIR/var/log*

*rm -f $ARMADEUS\_ROOTFS\_DIR/var/log && mkdir $ARMADEUS\_ROOTFS\_DIR/var/log/*

*mkdir $ARMADEUS\_ROOTFS\_DIR/var/log/boa*

Il ne nous reste qu’à créer le système de fichiers en effectuant un make puis en flashant la carte avec le nouveau système de fichiers. Une fois le système flashé, la commande *boa* permet de démarrer le serveur web. On teste que cela fonctionne bien en accédant au serveur web depuis le navigateur de notre pc.



On voit qu’en tapant l’adresse de notre carte Armadeus dans la barre d’URL de notre navigateur, on obtient la page HTML précédemment créé et flashée sur notre carte. Le serveur web est donc bien opérationnel.