

TP5 OSAE - BUILDROOT

Nous allons pour ce TP devoir installer notre propre environnement de développement croisé afin de profiter de toutes les possibilités de customisation de la carte.

INSTALLATION DE BUILDROOT¹



Pour ce TP, il sera demandé un rapport comprenant toute la procédure que vous allez réaliser avec la définition des éléments utilisés (package, protocoles, commandes, etc.) et la réponse aux questions posées.

Pour cette première partie, nous allons réutiliser la VM du dernier TP.

Nous commençons par créer un répertoire pour Buildroot. Vous pouvez commencer à vous documenter sur <https://buildroot.org/>. Aussi, il existe plusieurs tutoriels très intéressants sur le sujet.

Afin que buildroot puisse fonctionner, il faudrait installer un certain nombre de packages, nous allons le faire (sur la VM évidemment) avec le gestionnaire de package d'Ubuntu.

```
$ sudo apt-get install sed make binutils gcc g++ bash patch gzip  
bzip2 perl tar cpio python unzip rsync wget libncurses-dev libssl-dev
```



A quoi sert chacun des paquets plus haut (environ une ligne chacun) ? (Cherchez les réponses pendant l'installation des paquets qui prend un peu de temps).

Nous allons utiliser buildroot une option possible est de cloner le code source du dépôt Git (il faut installer git si ce dernier n'est pas présent sur votre système), puis :

```
$ git clone git://git.busybox.net/buildroot
```

On préférera télécharger une version du site <https://buildroot.org/downloads> , cela prendra moins de temps.

Vous pouvez utiliser la version buildroot-2022.02.9.tar.xz (vous pouvez vous aventurer à utiliser une version plus récente, mais celle donnée a été testée et validée).

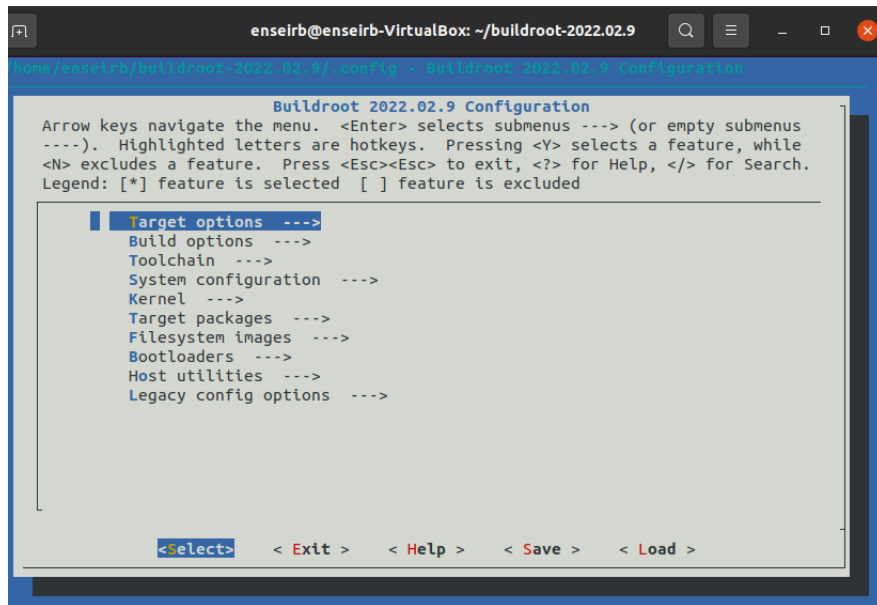
Nous allons à présent parcourir le menu de configuration de Buildroot.

¹ Une part importante de cette section a été reprise du « Buildroot Training » de FreeElectrons (actuellement « bootlin »)

Dans le répertoire racine de Buildroot (une fois décompressé), faites un :

\$ make menuconfig

Vous devriez voir apparaître le menu suivant



Nous allons explorer rapidement ce menu. Vous pouvez (devez) prendre plus de temps pour l'étudier en détail. Vous en aurez besoin pour la suite des TP's. Nous utiliserons exclusivement cet environnement pour les développements embarqués.

Dans la partie « Target Options » vous retrouverez :

- La plateforme utilisée par la BeagleBone Black est une plateforme ARM (little endian)
- Cette carte utilise un SoC Texas Instruments (TI) AM335x qui est basé sur un ARM Cortex-A8 (qu'il faudrait sélectionner comme « Target Architecture Variant »).
- Le format binaire utilisé est l'ELF, pour l'unité flottante VFPv3-D16 est une option utilisable et l'utilisation de l'ensemble d'instructions ARM est à utiliser par défaut (même si l'ensemble d'instruction Thumb-2 est possible aussi et plus compact)

Pour ce qui est de la chaîne de compilation (menu « Toolchain ») :


- Ici on peut sélectionner la chaîne créée par buildroot ou une chaîne externe comme celle de Linaro (reneignez-vous sur internet).

Configuration système (menu « system configuration »):

- Vous pouvez modifier ici le nom de l'hôte, le mot de passe root, etc.

Pour le menu « Kernel menu » :

- Vous pouvez sélectionner une version du noyau à utiliser. On peut, par exemple, choisir le plus récent
- On peut choisir le format binaire

	TP5 – OS Avancés et embarqués Construire son propre système	jalil.boukhobza@ensta-bretagne.fr
---	--	-----------------------------------

...

Explorez ces options et référez-vous, en cas de besoin, au site de Buildroot : <https://buildroot.org/downloads/manual/manual.html> ou sinon sur <http://nightly.buildroot.org/manual.pdf>

Nous pouvons manuellement sélectionner l'ensemble des options qui nous intéressent, mais cela peut être laborieux. Généralement, pour les cartes très utilisées comme la BeagleBone Black, il existe une configuration par défaut. Vous pouvez aller dans le répertoire « ./board », vous y trouverez un répertoire des cartes supportées. Allez dans le répertoire « ./beaglebone » et lisez le fichier « readme.txt ».

Il suffit donc de faire (à partir de la racine de buildroot) :

```
$ make beaglebone_defconfig
```

Ceci permet de sélectionner l'ensemble des options permettant de construire un environnement complet pour la carte dont nous disposons.

Une fois cela réalisé, vous pouvez refaire un : **\$ make menuconfig** afin de voir les options qui ont été sélectionnées.

Pour faire en sorte que la compilation prenne moins de temps (même si elle en prendra beaucoup), on va sélectionner la dernière version du noyau plutôt que le répertoire git. Allez dans le sous menu « Kernel », et sélectionnez dans « Kernel version » l'option « Custom version », puis dans « kernel version » taper « 5.3 ».

Il faut aussi changer les fichiers entête (header), allez sur le menu « Toolchain », puis sur « Kernel Headers » et sélectionnez « Same as kernel being built » et sur « Custom kernel headers series » choisissez « 5.3.x ».

Il faut prendre l'habitude de sauvegarder les fichiers « .config » générés afin de pouvoir facilement redéployer une configuration donnée. Copiez donc ce fichier puis renommez-le. Vous pouvez aussi rajouter un commentaire en entête afin de savoir à quoi il correspond.

Nous pouvons à présent lancer la compilation de l'ensemble des éléments constituant notre environnement embarqué. Faites donc un : **\$ make**

La construction de l'environnement Linux embarqué pour la beaglebone prend du temps. Si vous le souhaitez, au lieu d'attendre la fin de la compilation qui peut durer plus d'une heure, vous pouvez prendre une nouvelle machine virtuelle pour laquelle ce même environnement a été déjà compilé.

Le lien de la machine virtuelle à utiliser est sur moodle, ou sinon ici : O:\share\CSN3A\ENSEIRB_Ubuntu20_OS_2.ova.

Une fois la VM importée et lancée, faites un **\$ make menuconfig** à partir du répertoire de buildroot.

Sélectionnez la version du noyau 5.3 et la version des entêtes 5.3. Faites un **\$ make**

La compilation s'est normalement bien passée et on va pouvoir charger les fichiers sur la carte dans la section suivante.

INSTALLATION DE L'ENVIRONNEMENT EMBARQUE

Nous allons à présent installer notre « Linux embarqué » sur la carte SD afin de l'utiliser pour notre BeagleBone.

Pour se faire, il nous faut préparer la carte micro SD.

Nous devons créer deux partitions sur la carte micro SD :

- Une première partition pour le bootloader. Cette partition doit être conforme avec les exigences de la plateforme AM335x afin qu'elle puisse trouver le programme de démarrage sur la partition. On utilisera le système de fichiers FAT16. Nous allons y stocker le bootloader (MLO et u-boot.img), l'image du noyau (zImage), la *device tree* (am335x-boneblack.dtb), et un script U-Boot spécial permettant de définir la séquence de démarrage (uEnv.txt).
- La deuxième partition contiendra le système de fichiers à déployer sur la carte. On peut utiliser n'importe quel système de fichiers. Nous choisissons ext4.

La première chose à faire est d'identifier le nom de la carte micro SD une fois connectée à votre PC. Comme nous travaillons sur la VM, nous allons d'abord connecter la carte (grâce au lecteur de carte disponible à la demande) au PC, puis l'activer sur la VM (sur le menu de la VM, allez sur « Périphérique », puis « USB » et sélectionnez le lecteur de cartes – par exemple « *Generic Flash Card Reader/Writer* »).

Ubuntu devrait monter le système de fichiers automatiquement, vous pouvez donc regarder sur `/proc/partitions` pour y trouver votre carte SD. Le nom devrait commencer par **sdX²** (sdb, sdc, ...). **Attention à ne pas utiliser sda qui représente le disque de votre machine (VM dans votre cas).**

Si votre carte micro SD est représentée par le périphérique `/dev/sdb`, les partitions seront donc nommées `/dev/sdb1` et `/dev/sdb2` respectivement.

Afin de formater la carte SD, on exécutera les étapes suivantes :

- Démontez les partitions de la carte SD (par exemple : `$umount /dev/sdb1`)
- Effacez le début de la carte SD :
`$ sudo dd if=/dev/zero of=/dev/sdX bs=1M count=16` Utilisez le bon sdX en fonction de votre système.
- Créez deux partitions :
 - o Lancez l'outil **fdisk** avec : `$ sudo fdisk /dev/sdX`
 - o Créer une « nouvelle » petite partition (16Mo), primaire, avec le type « e » (W95 FAT16) et marquez cette partition comme « bootable » (ou amorçable)
 - o Créez une seconde partition, « primaire » aussi, avec le reste d'espace disponible, comme type, vous utiliserez le « 83 » (Linux).
 - o Vous pouvez « Ecrire » la table des partitions et sortir de fdisk
- Nous allons, à présent, formater les partitions, on commence par celle en FAT16
`$ sudo mkfs.vfat -F 16 -n BOOT /dev/sdX1`

² Si votre PC contient un lecteur interne, la carte SD sera probablement sur `/dev/mmcblk0`

- Puis la partition en ext4:

```
$ sudo mkfs.ext4 -L ROOTFS -E nodiscard /dev/sdX2
```

Nous pouvons à **présent débrancher la carte et la rebrancher** afin qu'elle soit détectée par Ubuntu. Normalement, vous devriez trouver une partition `/media/csn/BOOT` et une autre `/media/enseirb/ROOTFS` pour les 2 partitions sur la carte.

Nous pouvons à présent copier notre système sur la carte SD :

- Copiez les fichiers suivants se trouvant dans le répertoire « `./output/images` » de votre répertoire buildroot dans la partition « boot » : `MLO`, `u-boot.img`, `zImage`, `uEnv.txt`, et `am335x-boneblack.dtb` - Extrayez le fichier `rootfs.tar` dans la partition `rootfs` :

```
$ sudo tar -C /media/enseirb/ROOTFS/ -xf output/images/rootfs.tar
```



Pour le compte rendu du rapport, vous devrez expliquer en quelques paragraphes à quoi correspond chacun des fichiers copiés : `MLO`, `u-boot.img`, `zImage`, `uEnv.txt`, et `am335x-boneblack.dtb`. Vous devez aussi expliquer les commandes exécutées lors de ces étapes

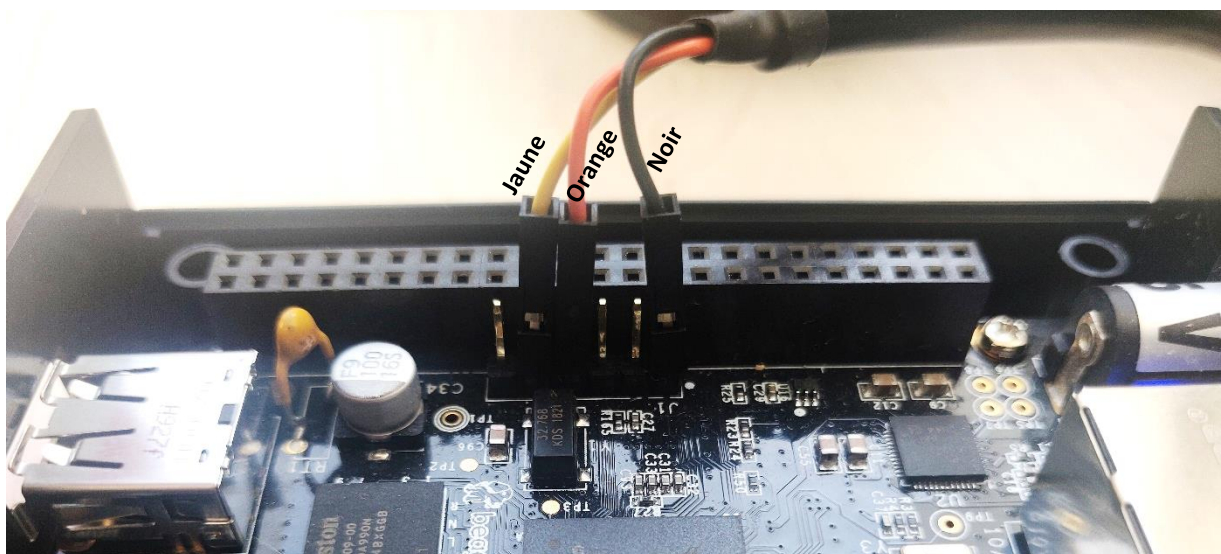
Démontez la carte micro SD et éjectez là du lecteur (`$umount /dev/sdX1` et `$umount /dev/sdX1`).

Nous allons à présent démarrer la carte BeagleBone en utilisant l'environnement nouvellement créé. Insérez la carte micro SD dans la BeagleBone (en vous assurant que cette dernière n'est pas sous tension ou branchée).

Nous allons aussi avoir besoin d'un câble série <-> USB afin de pouvoir communiquer en mode série avec la carte (accessible sur demande aussi=.



ATTENTION, la connexion doit se faire comme le montre la figure ci-dessous au risque de DETRUIRE la carte :



Rappelez-vous que la communication était possible pour le noyau précédent car le support permettant d'utiliser l'interface USB pour le réseau et le lien série avait été installé. Pour le noyau nouvellement créé, cela n'a pas été fait, par conséquent, nous utiliserons un câble série pour envoyer les commandes et un câble réseau pour le transfert de fichiers. C'est une installation classique dans le domaine de l'embarqué.

Pour démarrer sur la carte SD, maintenez le bouton S2 appuyé et branchez la carte, puis relâchez le bouton S2. N'oubliez pas d'activer la nouvelle interface sur VirtualBox afin de pouvoir communiquer en série avec la carte.



Vous pouvez lancer GTKterm. La configuration à mettre est différente de la précédente. Le port à utiliser est le ttyUSB0. Quelle différence y a-t-il entre ce port et le port utilisé précédemment (ttyACM0) ?

Maintenant que vous êtes connecté sur la carte en USB <-> série. Le login est « root » et il n'y a pas de mot de passe (nous n'avons pas demandé de création particulière).

Faites un **\$ uname -r** vous devriez avoir comme réponse la version 5.3.0 du noyau (celui de la carte SD et non celui installé sur la carte eMMC).

COMMUNICATION AVEC LA CARTE

Vous remarquerez sur cette carte que la commande scp n'est pas installée (nous verrons plus tard comment le faire). Nous allons donc explorer une autre possibilité de communication avec la carte, le protocole tftp (pour Trivial ftp).



Qu'est-ce que ce protocole ?

Nous allons donc devoir connecter la carte sur le réseau afin de pouvoir communiquer avec l'hôte.

Installation de tftp server **sur la VM** :

```
Lancez : $ sudo apt-get install tftp tftpd xinetd
```

Nous créons ensuite le répertoire de dépôt de fichiers à transmettre vers la carte, on l'appellera *tftpboot* (toujours sur la VM).

```
$ sudo mkdir /tftpboot suivi d'un $ sudo chmod 777 /tftpboot
```

Il faut aussi créer un fichier de configuration du serveur tftp (hôte) : créez le fichier de configuration **/etc/xinetd.d/tftp** et mettez-y :

```
# default: off
# description: The tftp server serves files using the trivial file
# transfer protocol. The tftp protocol is often used to boot
# diskless workstations, download configuration files to network-
# aware printers, and to start the installation process for some
# operating systems.
service tftp
{
    socket_type          = dgram
    protocol             = udp
```

```
wait = yes
user = root
server = /usr/sbin/in.tftpd
server_args = -s /tftpboot
# disable = yes
}
```

Une fois la configuration terminée, nous avons besoin de redémarrer le service xinetd (démon) :

```
$ sudo killall -HUP xinetd
```

Pour redémarrer le service il suffit d'un :

```
$ sudo service xinetd start
```

Ou :

```
$ sudo service xinetd restart
```



Vous devez bien sûr expliquer chacune des étapes précédentes dans votre rapport.

Maintenant que tftp est installé, nous allons configurer le réseau pour la VM ainsi que pour la carte.

Tout d'abord, nous devons configurer la VM en mode « ponté/accès par pont » ou « Bridged » afin qu'elle ait une adresse ip spécifique. Cela ne peut se faire qu'après extinction de la VM.

Ensuite, vous devez utiliser les adresses ip suivantes :

- VM : 192.168.1.2 (masque 255.255.255.0 et passerelle 192.168.1.1)
- Beaglebone : 192.168.1.50 pour la carte.

Configurez l'adresse ip de votre VM.


Vous devez faire pareil pour la carte à partir du terminal série, pour ce faire, on va modifier le fichier **/etc/network/interfaces**.

Mettez-y la configuration souhaitée :

```
auto eth0
iface eth0 inet static
address 192.168.1.50
netmask 255.255.255.0
gateway 192.168.1.1
```

Redémarrez la carte à présent (\$reboot) et tentez de « pinger » la carte à partir de la VM et vice versa. Si cela marche, vous pouvez continuer, sinon, trouvez l'erreur ...

Nous allons à présent tenter de récupérer un fichier de la VM et depuis la carte. Créez un fichier quelconque et mettez-le dans **/tftpboot** de la VM. Une fois cela réalisé, vous pouvez le copier dans la carte grâce à tftp (à partir du terminal série, et donc de la carte):

	TP5 – OS Avancés et embarqués Construire son propre système	jalil.boukhobza@ensta-bretagne.fr
---	--	-----------------------------------

```
$ tftp -g -r <nom_de_fichier> 192.168.1.2
```



Explorez l'utilisation de tftp et donnez en un résumé sur votre rapport.