



Table des matières

1) Création d'une image minimale pour raspberry Pi.....	1
2) Création d'une couche, d'une recette et configuration IP.....	5
3) Ajout de paquets :.....	9

1) Création d'une image minimale pour raspberry Pi

a) J'installe les paquets nécessaires à ma distribution sur le site <https://www.yoctoproject.org/docs/> dans la catégorie « Quick Build »

```
apt install [les paquets]
```

b) Je vérifie que j'ai au moins 40GB de libre et que j'ai mon locale en en_US.UTF-8 UTF-8 :

```
test@202-15:~$ df -h
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
udev              3,9G      0    3,9G   0% /dev
tmpfs             786M     11M    775M   2% /run
/dev/sda2         220G    104G   105G  50% /
```

Pour passer la locale en en_US.UTF-8 je dois me passer en root et modifier le fichier /etc/locale.gen :

```
fr_FR.UTF-8 UTF-8
en_US.UTF-8 UTF-8
```

Maintenant j'active la locale en faisant un locale-gen :

```
locale-gen
```

c) Il faut maintenant que je télécharge mon image et que je la compile (cela prend entre 1 et 2h) :

```
export LANG=en_US.UTF-8
locale
umask 022
git clone https://git.yoctoproject.org/git/poky
cd poky/
git clone https://git.yoctoproject.org/git/meta-raspberrypi
source oe-init-build-env rpi-build
bitbake-layers add-layer ../meta-raspberrypi
echo 'MACHINE="raspberrypi3"' >> conf/local.conf
echo 'IMAGE_FSTYPES+="rpi-sdimg"' >> conf/local.conf
time bitbake core-image-minimal
```

d) Je regarde la taille de mon dossier :

```
test@202-15:~/poky$ du -sh rpi-build/
33G  rpi-build/
```

e) Pour gagner un peu d'espace disque je rajoute cette ligne à la fin du fichier conf/local.conf :

```
INHERIT+="rm_work"
```

Je relance la compilation :

```
test@202-15:~/poky$ source oe-init-build-env rpi-build

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

Common targets are:
  core-image-minimal
  core-image-full-cmdline
  core-image-sato
  core-image-weston
  meta-toolchain
  meta-ide-support

You can also run generated qemu images with a command like
'runqemu qemux86'

Other commonly useful commands are:
- 'devtool' and 'recipetool' handle common recipe tasks
- 'bitbake-layers' handles common layer tasks
- 'oe-pkgdata-util' handles common target package tasks
```

```
test@202-15:~/poky/rpi-build$ time bitbake core-image-minimal
Loading cache: 100% |
ETA:  --:--:--
Loaded 0 entries from dependency cache.
Parsing recipes: 100% |#####|
Time: 0:00:28
Parsing of 833 .bb files complete (0 cached, 833 parsed). 1437
targets, 66 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "1.49.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "universal"
TARGET_SYS           = "arm-poky-linux-gnueabi"
MACHINE              = "raspberrypi3"
DISTRO               = "poky"
DISTRO_VERSION        = "3.2+snapshot-
796be0593a607938aef3941372a9238b7e895446"
TUNE_FEATURES         = "arm vfp cortexa7 neon vfpv4 thumb
callconvention-hard"
TARGET_FPU            = "hard"
meta
meta-poky
meta-yocto-bsp        =
"master:796be0593a607938aef3941372a9238b7e895446"
meta-raspberrypi      =
"master:e4f5c32925fec90ff688e51197cb052fe12af82e"

Initialising tasks: 100% |#####|
Time: 0:00:02
Sstate summary: Wanted 2 Found 0 Missed 2 Current 1029 (0% match,
99% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 3275 tasks of which 2623 didn't
need to be rerun and all succeeded.

real 1m47,471s
user 0m1,195s
sys 0m0,159s
```

Je vérifie combien de place j'ai gagné :

```
test@202-15:~/poky$ du -sh rpi-build/
16G  rpi-build/
```

La place a été multiplié par 2

f) Je regarde la taille du fichier :

```
test@202-15:~/poky/rpi-build/tmp/deploy/images/raspberrypi3$ ls -l
core-image-minimal-raspberrypi3-20201215085757.rootfs.rpi-sdimg
-rw-r--r-- 1 test test 54525952 déc. 15 09:59 core-image-minimal-
raspberrypi3-20201215085757.rootfs.rpi-sdimg
```

g) Je copie ce fichier dans une carte sd et je la met dans une raspi et je l'alimente :

```
test@202-15:~/poky/rpi-build/tmp/deploy/images/raspberrypi3$ df -h
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
udev                3,9G      0   3,9G   0% /dev
tmpfs               786M     11M   775M   2% /run
/dev/sda2           220G     87G   123G  42% /
tmpfs               3,9G     94M   3,8G   3% /dev/shm
tmpfs               5,0M     4,0K   5,0M   1% /run/lock
tmpfs               3,9G      0   3,9G   0% /sys/fs/cgroup
/dev/loop3          72M      72M      0 100% /snap/lxd/18546
/dev/loop4          98M      98M      0 100% /snap/core/10185
/dev/loop8          34M      34M      0 100% /snap/multipass-gui/16
/dev/loop5          221M     221M      0 100% /snap/multipass/3062
/dev/loop0          56M      56M      0 100% /snap/core18/1885
/dev/loop7          56M      56M      0 100% /snap/core18/1932
/dev/loop1          98M      98M      0 100% /snap/core/10444
/dev/loop2          70M      70M      0 100% /snap/lxd/18520
/dev/loop6          221M     221M      0 100% /snap/multipass/3044
/dev/sda1           511M     5,2M   506M   1% /boot/efi
tmpfs               786M     28K   786M   1% /run/user/1000
tmpfs               1,0M      0     1,0M   0% /var/snap/lxd/common/ns
/dev/sdb1           253M     53M   200M  22% /media/test/boot
/dev/sdb2           15G      1,5G    13G  11% /media/test/rootfs
```

```
test@202-15:~/poky/rpi-build/tmp/deploy/images/raspberrypi3$
```

```
umount /dev/sdb1
```

```
test@202-15:~/poky/rpi-build/tmp/deploy/images/raspberrypi3$
```

```
umount /dev/sdb2
```

```
test@202-15:~/poky/rpi-build/tmp/deploy/images/raspberrypi3$ sudo
dd if=core-image-minimal-raspberrypi3.rpi-sdimg of=/dev/sdb bs=4M
conv=fsync
```

```
[sudo] Mot de passe de test :
```

```
13+0 enregistrements lus
```

```
13+0 enregistrements écrits
```

```
54525952 octets (55 MB, 52 MiB) copiés, 4,70058 s, 11,6 MB/s
```

h) Je peux définir le nom de l'hôte dans le fichier conf/local.conf :

```
hostname_pn-base-files="yocto-15"
```

Une fois fait je refais le source oe... et le bitbake pour la compilation du nouveau raspi

Maintenant je redémonte mes partitions usb et je refait la copie de l'image dans la carte sd

Au lancement de la raspi je vois bien que mon hostname a changé

2) Création d'une couche, d'une recette et configuration IP

a) Je regarde sur mon pc Linux dans le fichier /poky/meta/recipes-core/init-ifupdownw/init-ifupdown-1.0/interfaces et je regarde que le contenu soit identique au fichier /etc/network/interfaces du Raspi :

```
# /etc/network/interfaces -- configuration file for ifup(8),
ifdown(8)

# The loopback interface
auto lo
iface lo inet loopback

# Wireless interfaces
iface wlan0 inet dhcp
    wireless_mode managed
    wireless_essid any
    wpa-driver wext
    wpa-conf /etc/wpa_supplicant.conf

iface atml0 inet dhcp

# Wired or wireless interfaces
auto eth0
iface eth0 inet dhcp
iface eth1 inet dhcp
```

Le contenu est bien identique entre les deux fichiers

b) Pour personnaliser le paramétrage réseau je dois surcharger le contenu du fichier interfaces :

```
test@202-15:~/poky/meta/recipes-core/init-ifupdown/init-ifupdown-1.0$ cd /home/test/poky/rpi-build/
test@202-15:~/poky/rpi-build$ bitbake-layers create-layer ../meta-XXX
```

```
NOTE: Starting bitbake server...
Add your new layer with 'bitbake-layers add-layer ../meta-XXX'
test@202-15:~/poky/rpi-build$ bitbake-layers add-layer ../meta-XXX
NOTE: Starting bitbake server...
test@202-15:~/poky/rpi-build$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer                path
priority
=====
=====
meta                  /home/test/poky/meta                5
meta-poky             /home/test/poky/meta-poky           5
meta-yocto-bsp        /home/test/poky/meta-yocto-bsp       5
meta-raspberrypi      /home/test/poky/meta-raspberrypi     9
meta-XXX              /home/test/poky/meta-XXX            6
```

c) Je modifie le fichier meta-XXX/conf/layer.conf et je change la priorité de la couche réseau :

```
BBFILE_PRIORITY_meta-XXX = "10"
```

Je ré-affiche maintenant la liste des priorités des couches réseau :

```
test@202-15:~/poky/rpi-build$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer                path
priority
=====
=====
meta                  /home/test/poky/meta                5
meta-poky             /home/test/poky/meta-poky           5
meta-yocto-bsp        /home/test/poky/meta-yocto-bsp       5
meta-raspberrypi      /home/test/poky/meta-raspberrypi     9
meta-XXX              /home/test/poky/meta-XXX            10
```

La priorité de meta-XXX est passé de 6 à 3

d) Je crée une recette de mise à jour du fichier interfaces :

```
test@202-15:~/poky/rpi-build$ RCP=../meta-XXX/recipes-core/init-
ifupdown
test@202-15:~/poky/rpi-build$ mkdir -p $RCP/files
test@202-15:~/poky/rpi-build$ cat << EOF > $RCP/init-
ifupdown_1.0.bbappend
> FILESEXTRAPATHS_prepend:="\${THISDIR}/files:"
> EOF
test@202-15:~/poky/rpi-build$ cat << EOF > $RCP/files/interfaces
> # /etc/network/interfaces -- See ifup(8), ifdown(8)
```

```
> # The loopback interface
> auto lo
> iface lo inet loopback
> # Wired interface
> auto eth0
> iface eth0 inet static
>     address 10.202.15.2
>     netmask 255.255.0.0
>     gateway 10.202.255.254
> EOF
```

e) Je copie la nouvelle image sur la carte sd (en recompilant le kernel avec source oe et bitbake)
puis je démonte les partitions sdb et je copie :

Je vérifie directement sur le pc linux dans le dossier etc puis network et enfin dans le fichier interfaces que ma conf a bien été changé :

```
# /etc/network/interfaces -- See ifup(8), ifdown(8)
# The loopback interface
auto lo
iface lo inet loopback
# Wired interface
auto eth0
iface eth0 inet static
    address 10.202.15.2
    netmask 255.255.0.0
    gateway 10.202.255.254
```

f) J'alimente mon Raspi et je vérifie que j'ai bien l'IP que j'ai donné sur l'interface eth0 :

```
ip a

2: eth0 : .....
    link/ether ....
    inet 10.202.15.2/16 scope global eth0
        valid_lft forever preferred_lft forever
```

Je vois bien que mon IP est sur l'interface eth0

g) Je peux changer le mot de passe par défaut pour l'utilisateur root et ajouter un nouvel utilisateur iot dans le fichier conf/local.conf :

```
INHERIT+="extrausers"
EXTRA_USERS_PARAMS="useradd iot; \
                    usermod -P abc123 iot; \
```

```
usermod -P secretPwd root; \  
"
```

Je dois maintenant recréer l'image pour que tout se mette à jour

Je vérifie le contenu du fichier /etc/shadow dans l'image créée. Je vois bien que j'ai mon utilisateur root et iot avec leur mot de passe hashés :

```
root:$6$fvBbV/  
mCouQWv$00ccXP3ZoQDj38UqLgPwMJyfo8eitfS55uonmcFCCqA5mrdqCfiM2PU...  
iot:$6$zQyuL/  
hc3t$80KV7GdLc5VyWYRPq8w2VXxBJ4ajwGA0cgDDCF6TEJj6AP7RbMwyqWddAvD..
```

Je peux vérifier le mot de passe en utilisant la commande openssl et le salt de l'utilisateur concerné (le salt se trouve dans shadow entre le 2ème et le 3ème \$: voir surlignage au dessus) :

```
root@202-15:/home/test/poky/rpi-build/tmp/deploy/images/  
raspberrypi3# openssl passwd -6 -salt fvBbV/mCouQWv  
Password:  
$6$fvBbV/  
mCouQWv$00ccXP3ZoQDj38UqLgPwMJyfo8eitfS55uonmcFCCqA5mrdqCfiM2PUeDN  
zUmihV90e6wqTvaPA4pqE4Sf7St.
```

Pour le mot de passe je met celui que j'ai donné à root dans le fichier local.conf et je vérifie que le mot de passe hashé soit bien le même que le hash dans le fichier shadow

h) Pour augmenter la sécurité au niveau du fichier de configuration (local.conf) au lieu de mettre le mot de passe en clair on peut mettre directement le mot de passe hashé. Je commence par générer le hash du mot de passe torototo :

```
root@202-15:/home/test/poky/rpi-build/tmp/deploy/images/  
raspberrypi3# openssl passwd -6 torototo  
$6$5NxJeREdw783MNP/$oQXdFSYzqHzAlomk84D5F1E6rREk/  
pUMP0l0uC99vN9HPWFGGrYaow5sPadMbDHrfJlaDTujg7V0izdS8UNBa4/
```

Je place le hash du mot de passe dans le fichier de configuration :

```
usermod -p  
'\$6\$5NxJeREdw783MNP/$oQXdFSYzqHzAlomk84D5F1E6rREk/pUMP0l0uC99vN  
9HPWFGGrYaow5sPadMbDHrfJlaDTujg7V0izdS8UNBa4/' iot; \  
"
```

Attention : il faut mettre un \ devant chaque dollars car sinon il est interprété comme une variable et il supprime tout ce qui est après

i) Maintenant je peux relancer la compilation de l'image et ensuite retourner dans le fichier shadow et vérifier que le hash est bien le même que celui mis dans le fichier de configuration (local.conf) :

```
iot:$6$5NxJeREdw783MNP/$oQXdFSYzqHzAlomk84D5F1E6rREk/  
pUMP0l0uC99vN9HPWFGGrYaow5sPadMbDHrfJlaDTujg7V0izdS8UNBa4/::0:99999  
:7:::
```

Le hash est bien le même

3) Ajout de paquets :

a) J'ajoute les directives de création d'un historique de fabrication au fichier de configuration (local.conf) :

```
INHERIT+="buildhistory"  
BUILDHISTORY_COMMIT="1"
```

Je recompile l'image et je vérifie que le dossier buildhistory a bien été créé :

```
test@202-15:~/poky/rpi-build/tmp/deploy/images/raspberrypi3$ cd  
/home/test/poky/rpi-build/  
test@202-15:~/poky/rpi-build$ ls  
bitbake-cookerdaemon.log  cache  downloads  tmp  
buildhistory              conf    sstate-cache  
test@202-15:~/poky/rpi-build$ cd buildhistory/
```

Je peux voir que le dossier contient les logs de mes images et les données de mon raspi :

```
test@202-15:~/poky/rpi-build/buildhistory$ ls  
images  metadata-revs
```

b) J'affiche la liste des recettes existantes et je vérifie que example existe bien :

```
test@202-15:~/poky/rpi-build$ bitbake -s |grep example  
example                               :0.1-r0  
gst-examples                          :1.18.1-r0
```

c) J'ajoute la variable IMAGE_INSTALL_append dans le fichier de configuration (local.conf) pour ajouter un paquet à ma distribution :

```
IMAGE_INSTALL_append="openssh"
```

Je compile mon image et je vérifie le contenu du répertoire buildhistory :

```
test@202-15:~/poky/rpi-build/buildhistory$ ls
images  metadata-revs  packages
```

Je vois maintenant qu'un dossier packages s'est créé et dedans il contient le paquet openssh et toutes ses dépendances :

```
test@202-15:~/poky/rpi-build/buildhistory/packages/cortexa7t2hf-
neon-vfpv4-poky-linux-gnueabi$ ls
libjitterentropy  openssh  rng-tools  sudo  sysfsutils
```

d) Pour changer le nom de la recette exemple je vais dans le répertoire meta-XXX/recipes-example et je modifie tous les noms qui sont dedans (dossiers et fichiers) :

```
test@202-15:~/poky/meta-XXX/recipes-example$ ls
example
test@202-15:~/poky/meta-XXX/recipes-example$ sudo mv example/ rgb-
sensor/
test@202-15:~/poky/meta-XXX/recipes-example$ ls
rgb-sensor
test@202-15:~/poky/meta-XXX/recipes-example$ cd rgb-sensor/
test@202-15:~/poky/meta-XXX/recipes-example/rgb-sensor$ ls
example_0.1.bb
test@202-15:~/poky/meta-XXX/recipes-example/rgb-sensor$ sudo mv
example_0.1.bb rgb-sensor_1.0.bb
test@202-15:~/poky/meta-XXX/recipes-example/rgb-sensor$ ls
rgb-sensor_1.0.bb
```

e) Je fixe le contenu du fichier rgb-sensor_1.0.bb avec les instructions suivantes :

```
DESCRIPTION="RGB sensor application"
SECTION="Examples"
LICENSE="GPLv2"
LIC_FILES_CHKSUM="file://${COMMON_LICENSE_DIR}/GPL-
2.0;md5=801f80980d171dd6425610833a$
PR="r0"
SRC_URI="file://${BPN}-${PV}.tar.gz"
SRC_URI[md5sum]="e574da4da3f63804ab3509888e516a52"
SRC_URI[sha256sum]="24a6a3cc96027ff52d3670ad867df7e05e09fdd22c9eac
3c78961b1d7e61b58d"
EXTRA_OECMAKE=""
inherit cmake
```

Je crée le dossier files dans ma recette et j'y copie le fichier .tar.gz dans ce dossier

Je lance la fabrication manuellement :

```
test@202-15:~/poky/meta-XXX/recipes-example$ bitbake rgb-sensor
```

f) Pour ajouter ce paquet à ma distribution je dois modifier le fichier de configuration (local.conf) :

```
IMAGE_INSTALL_append+="rgb-sensor"
```

Je dois maintenant compiler mon image et la remettre sur la carte micro sd. Une fois fais dans le dossier usr/bin je dois voir le fichier rgb-sensor qui s'est ajouté dedans :

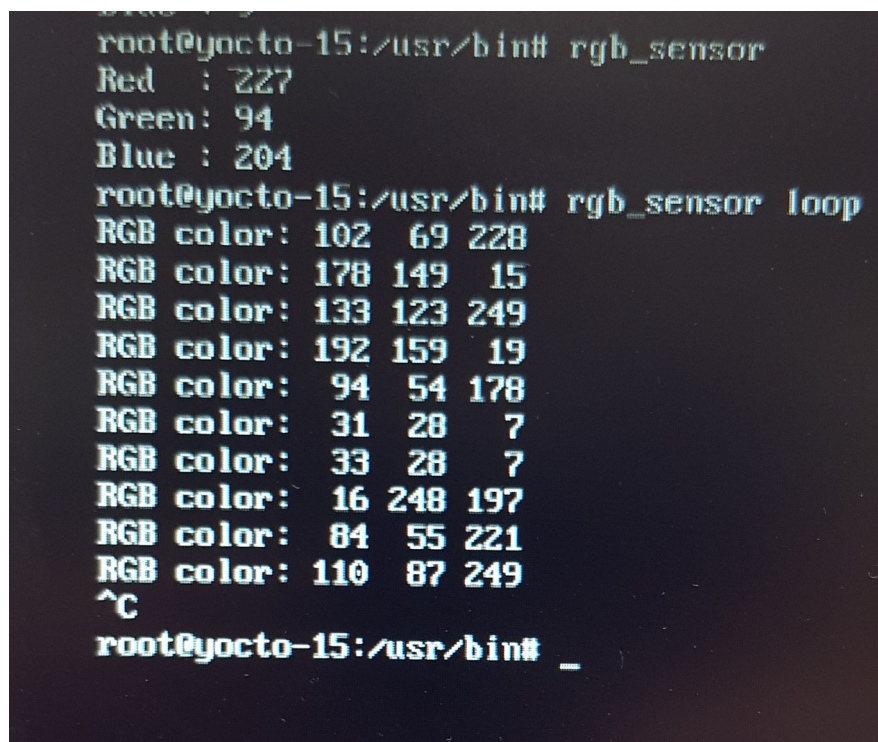
```
root@202-15:/home/test/poky/rpi-build/tmp/deploy/images/  
raspberrypi3# ls /media/test/48918d19-5b45-4e04-8933-d43a6436a0a3/  
usr/bin/ |grep rgb  
rgb_sensor  
setvtrgb
```

g) Je dois activer l'i2c sur mon rapsi. Pour cela je modifie le fichier de configuration (local.conf) :

```
ENABLE_I2C="1"  
IMAGE_INSTALL_append+="i2c-tools kernel-modules"  
KERNEL_MODULE_AUTOLOAD+="i2c-dev"
```

Je recompile l'image et je le remet dans la carte micro sd

h) Je dois maintenant tester le programme rgb_sensor sur mon raspi (avec le module connecté sur les pins de mon raspberry) :



```
root@yocto-15:/usr/bin# rgb_sensor  
Red : 227  
Green: 94  
Blue : 204  
root@yocto-15:/usr/bin# rgb_sensor loop  
RGB color: 102 69 228  
RGB color: 178 149 15  
RGB color: 133 123 249  
RGB color: 192 159 19  
RGB color: 94 54 178  
RGB color: 31 28 7  
RGB color: 33 28 7  
RGB color: 16 248 197  
RGB color: 84 55 221  
RGB color: 110 87 249  
^C  
root@yocto-15:/usr/bin# _
```