TP Projet Yocto

1. Préparations des layers

* Pour créer les layers on utilise les commandes suivantes :

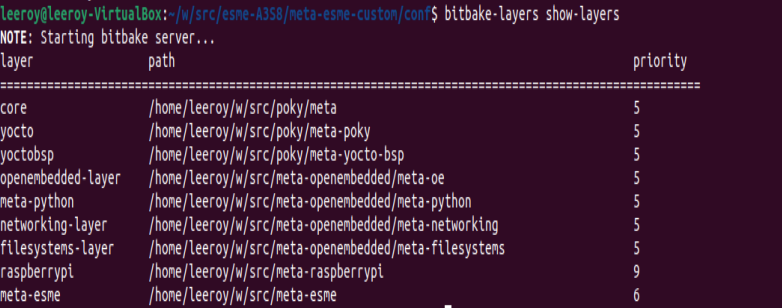
$bitbake-layers create-layer meta-esme-images

$bitbake-layers create-layer meta-esme-custom

* Pour vérifier si les layers sont activés ou non on utilise la commande suivante :

$bitbake-layers show-layers

On obtient la sortie suivante :



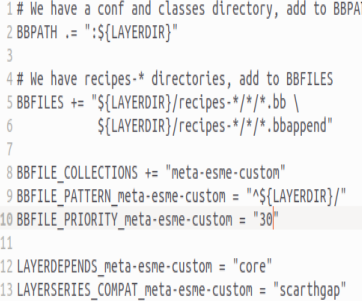
* Pour définir les priorités on modifie le fichier layer.conf grâce à la commande suivante :

$ gedit layer.conf

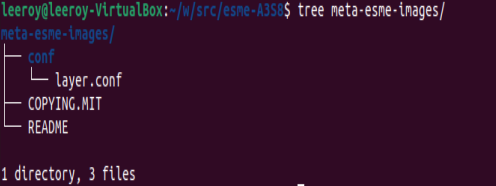
* Puis on modifie la ligne « BBFILE\_PRIORITY\_meta-esme-images = "20" » pour le premier layer :



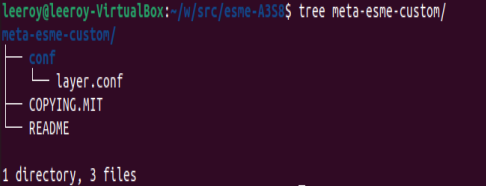
* Puis « BBFILE\_PRIORITY\_meta-esme-custom = "30" » pour le second layer pour avoir respectivement des priorités de 20 et 30.



* L’arborescence du premier layer, qui justifie sa création, est la suivante :

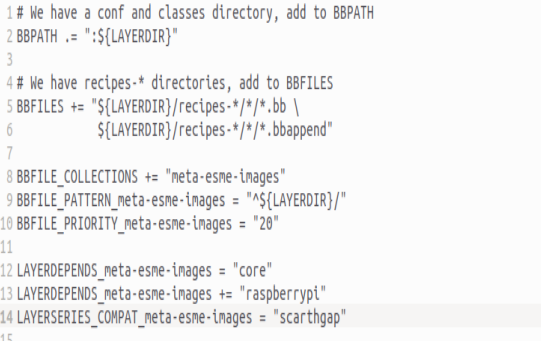


* L’arborescence du second layer est la suivante :



1. Préparation des layers

* Pour modifier la configuration du layer « meta-esme-images » de manière à exprimer la dépendance du layer « raspberrypi », on ajoute la ligne LAYERSDEPENDS\_meta-esme-images += “raspberrypi” en accédant au fichier grâce à la commande $ gedit layer.conf

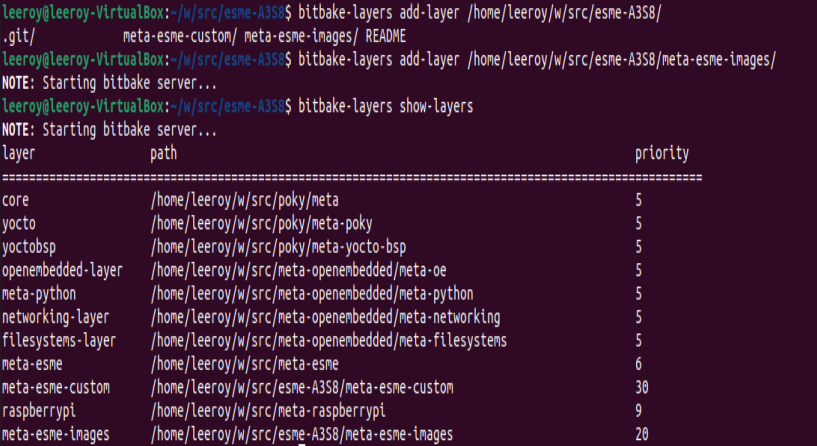


1. Activation des layers

* $bitbake-layers add-layer /home/leeroy/w/src/esme-A3S8/meta-esme-images/

$bitbake-layers add-layer /home/leeroy/w/src/esme-A3S8/meta-esme-custom/

$bitbake-layers add-layer /home/leeroy/w/src/meta-raspberrypi/



1. Liste des layers pour la configuration du build

* Les deux fichiers pour l’emplacement de configuration du build et du fichier de configuration des layers pour le build open-embedded sont les fichiers build.conf et bblayers.conf qui se situent dans le build-scarthgap/conf/.

1. Surcharge de la recipe “rpi-config”

leeroy@leeroy-VirtualBox:~/w/build-scarthgap$ recipetool newappend /home/leeroy/w/src/esme-A3S8/meta-esme-custom/ rpi-config

NOTE: Starting bitbake server...

Loading cache: 100% |################################################################################################| Time: 0:00:01

Loaded 4688 entries from dependency cache.

WARNING: No bb files in default matched BBFILE\_PATTERN\_meta-esme-images '^/home/leeroy/w/src/esme-A3S8/meta-esme-images/'

WARNING: No bb files in default matched BBFILE\_PATTERN\_meta-esme-custom '^/home/leeroy/w/src/esme-A3S8/meta-esme-custom/'

Summary: There were 2 WARNING messages.

/home/leeroy/w/src/esme-A3S8/meta-esme-custom/recipes-bsp/bootfiles/rpi-config\_git.bbappend

leeroy@leeroy-VirtualBox:~/w/src/esme-A3S8/meta-esme-custom/recipes-bsp/bootfiles$ gedit rpi-config\_git.bbappend

On ecrit la ligne suivante : ENABLE\_UART = “1”

1. Vérification de la mise à jour de la recipe rpi-config

leeroy@leeroy-VirtualBox:~/w/src/esme-A3S8$ bitbake -e rpi-config | grep UART

# $ENABLE\_UART [2 operations]

ENABLE\_UART="1"

# UART support

echo "# Enable UART" >>$CONFIG

bbfatal "Invalid value for ENABLE\_UART [1]. The value for ENABLE\_UART can be 0 or 1."

bbfatal "Invalid configuration: RPI\_USE\_U\_BOOT requires to enable the UART in config.txt for raspberrypi0-wifi"

echo "# U-Boot requires UART" >>$CONFIG

7)

bitbake rpi-test-image

Loading cache: 100% |################################################################################################| Time: 0:00:01

Loaded 4686 entries from dependency cache.

WARNING: No bb files in default matched BBFILE\_PATTERN\_meta-esme-images '^/home/leeroy/w/src/esme-A3S8/meta-esme-images/'

NOTE: Resolving any missing task queue dependencies

Build Configuration:

BB\_VERSION = "2.8.0"

BUILD\_SYS = "x86\_64-linux"

NATIVELSBSTRING = "universal"

TARGET\_SYS = "arm-poky-linux-gnueabi"

MACHINE = "raspberrypi0-wifi"

DISTRO = "poky"

DISTRO\_VERSION = "5.0.4"

TUNE\_FEATURES = "arm thumb vfp arm1176jzfs callconvention-hard"

TARGET\_FPU = "hard"

meta

meta-poky

meta-yocto-bsp = "scarthgap:200d12b6a58ad961d60a7774ca0f7a9d29498724"

meta-oe

meta-python

meta-networking

meta-filesystems = "scarthgap:72018ca1b1a471226917e8246e8bbf9a374ccf97"

meta-esme-custom = "scart:d22182cc5355069b5095a24987a9c9669b2d95c3"

meta-raspberrypi = "scarthgap:6df7e028a2b7b2d8cab0745dc0ed2eebc3742a17"

meta-esme-images = "scart:d22182cc5355069b5095a24987a9c9669b2d95c3"

Sstate summary: Wanted 3 Local 0 Mirrors 0 Missed 3 Current 2403 (0% match, 99% complete)################## | ETA: 0:00:00

Removing 3 stale sstate objects for arch raspberrypi0\_wifi: 100% |###################################################| Time: 0:00:00

NOTE: Executing Tasks

NOTE: Tasks Summary: Attempted 4901 tasks of which 4894 didn't need to be rerun and all succeeded.

leeroy@leeroy-VirtualBox:~/w/build-scarthgap/tmp/deploy/images/raspberrypi0-wifi$ sudo bmaptool copy --bmap rpi-test-image-raspberrypi0-wifi.rootfs.wic.bmap rpi-test-image-raspberrypi0-wifi.rootfs.wic.bz2 /dev/sdb

bmaptool: info: block map format version 2.0

bmaptool: info: 113357 blocks of size 4096 (442.8 MiB), mapped 52083 blocks (203.4 MiB or 45.9%)

bmaptool: info: copying image 'rpi-test-image-raspberrypi0-wifi.rootfs.wic.bz2' to block device '/dev/sdb' using bmap file 'rpi-test-image-raspberrypi0-wifi.rootfs.wic.bmap'

bmaptool: info: 100% copied

bmaptool: info: synchronizing '/dev/sdb'

bmaptool: info: copying time: 54.7s, copying speed 3.7 MiB/sec

8) Programme en C permettant d’allumer la led de la carte par la GPIO 17 :

#include <gpiod.h>

#include <stdio.h>

#include <unistd.h>

#define GPIO\_CHIP "/dev/gpiochip0" // Chemin du contrôleur GPIO

#define GPIO\_PIN 17 // Numéro de la GPIO à utiliser

#define TOGGLE\_INTERVAL 1 // Intervalle de basculement (en secondes)

int main() {

struct gpiod\_chip \*chip;

struct gpiod\_line \*line;

int value = 0; // Initialisation de la valeur (0 : éteint, 1 : allumé)

// Ouvrir le contrôleur GPIO

chip = gpiod\_chip\_open(GPIO\_CHIP);

if (!chip) {

perror("Erreur : Impossible d'ouvrir le contrôleur GPIO");

return 1;

}

// Obtenir la ligne GPIO

line = gpiod\_chip\_get\_line(chip, GPIO\_PIN);

if (!line) {

perror("Erreur : Impossible d'obtenir la ligne GPIO");

gpiod\_chip\_close(chip);

return 1;

}

// Configurer la ligne GPIO en sortie

if (gpiod\_line\_request\_output(line, "gpio-toggle", 0) < 0) {

perror("Erreur : Impossible de configurer la ligne GPIO en sortie");

gpiod\_chip\_close(chip);

return 1;

}

printf("Contrôle de la LED sur GPIO#%d (Ctrl+C pour arrêter)\n", GPIO\_PIN);

// Boucle infinie pour basculer l'état de la LED

while (1) {

value = !value; // Inverser la valeur (0 -> 1 ou 1 -> 0)

if (gpiod\_line\_set\_value(line, value) < 0) {

perror("Erreur : Impossible de modifier la valeur GPIO");

break;

}

printf("GPIO#%d est maintenant : %s\n", GPIO\_PIN, value ? "ALLUMÉE" : "ÉTEINTE");

sleep(TOGGLE\_INTERVAL);

}

// Libérer les ressources

gpiod\_line\_release(line);

gpiod\_chip\_close(chip);

return 0;

}

9) Fichier Makefile :

# Répertoires

INSTALL\_DIR ?= ./.install/

# Fichiers

PROG := gpio-toggle

SCRIPT = esme-led

# Fichiers objets

OBJS := $(subst .c,.o,$(shell ls \*.c))

# Options de compilation et d'édition de liens

CFLAGS += $(shell pkg-config --cflags libgpiod)

LDLIBS += $(shell pkg-config --libs libgpiod)

# Cible par défaut

all: $(PROG)

$(PROG): $(OBJS)

# Cible d'installation

install: $(PROG) $(SCRIPT)

# Créer le répertoire d'installation s'il n'existe pas

mkdir -p $(INSTALL\_DIR)/usr/bin

# Copier le programme dans le répertoire d'installation

cp $(PROG) $(INSTALL\_DIR)/usr/bin

@echo "$(PROG) installé dans $(INSTALL\_DIR)/usr/bin"

# Cible pour nettoyer les fichiers générés

clean:

-$(RM) -rf $(PROG) $(OBJS)

@echo "Nettoyage effectué"

10) création d’un script de démarrage

#!/bin/sh

### BEGIN INIT INFO

# Provides: esme-led

# Required-Start: $remote\_fs $time

# Required-Stop: $remote\_fs $time

# Default-Start: 3 4 5

# Default-Stop: 0 1 2 6

# Short-Description: ESME LED GPIO#17 toggle service

### END INIT INFO

# Variables

DAEMON="/usr/bin/gpiod"

NAME="esme-led"

PIDFILE="/var/run/${NAME}.pid"

case "$1" in

start)

echo "Starting $NAME..."

if start-stop-daemon --start --quiet --background --make-pidfile --pidfile "$PIDFILE" --exec "$DAEMON"; then

echo "$NAME started."

else

echo "Failed to start $NAME."

fi

;;

stop)

echo "Stopping $NAME..."

if start-stop-daemon --stop --quiet --pidfile "$PIDFILE"; then

echo "$NAME stopped."

else

echo "$NAME is not running."

fi

;;

restart)

echo "Restarting $NAME..."

$0 stop

$0 start

;;

status)

if [ -f "$PIDFILE" ] && ps -p $(cat "$PIDFILE") > /dev/null 2>&1; then

echo "Status of $NAME: running with PID=$(cat $PIDFILE)"

else

echo "Status of $NAME: stopped"

fi

;;

\*)

echo "Usage: $0 (start | stop | restart | status)"

;;

esac

exit 0

11)Enrechissement du Makefile

# Répertoires

INSTALL\_DIR ?= ./.install/

# Fichiers

PROG := gpio-toggle

SCRIPT = esme-led

# Fichiers objets

OBJS := $(subst .c,.o,$(shell ls \*.c))

# Options de compilation et d'édition de liens

CFLAGS += $(shell pkg-config --cflags libgpiod)

LDLIBS += $(shell pkg-config --libs libgpiod)

# Cible par défaut

all: $(PROG)

$(PROG): $(OBJS)

# Cible d'installation

install: $(PROG) $(SCRIPT)

# Créer le répertoire d'installation s'il n'existe pas

mkdir -p $(INSTALL\_DIR)/usr/bin

# Copier le programme dans le répertoire d'installation

cp $(PROG) $(INSTALL\_DIR)/usr/bin

@echo "$(PROG) installé dans $(INSTALL\_DIR)/usr/bin"

# Créer le répertoire d'installation s'il n'existe pas

mkdir -p $(INSTALL\_DIR)/etc/init.d

# Copier le script esme-led dans /etc/init.d et ajuster les permissions

cp $(SCRIPT) $(INSTALL\_DIR)/etc/init.d/

chmod 0755 $(INSTALL\_DIR)/etc/init.d/$(SCRIPT)

@echo "$(SCRIPT) installé dans $(INSTALL\_DIR)/etc/init.d avec chmod 0755"

# Cible pour nettoyer les fichiers générés

clean:

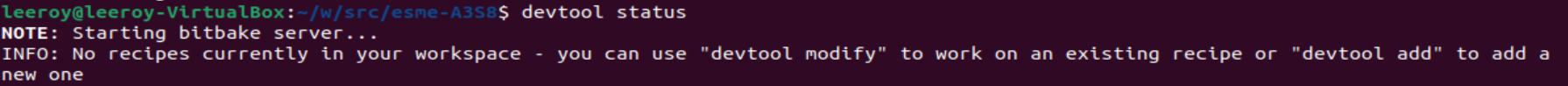
-$(RM) -rf $(PROG) $(OBJS)

@echo "Nettoyage effectué"

12) DEVTOOL : création d’un workspace

devtool create-workspace --create-only ~/w/src/esme-A3S8/workspace

devtool status





13 ) devtool add esme-gpio ~/w/src/gpio-toggle/

recipetool edit esme-gpio

(inherit pkgconfig

DEPENDS += "libgpiod (<2.0)")

bitbake esme-gpio

Build Configuration:

BB\_VERSION = "2.8.0"

BUILD\_SYS = "x86\_64-linux"

NATIVELSBSTRING = "universal"

TARGET\_SYS = "arm-poky-linux-gnueabi"

MACHINE = "raspberrypi0-wifi"

DISTRO = "poky"

DISTRO\_VERSION = "5.0.4"

TUNE\_FEATURES = "arm thumb vfp arm1176jzfs callconvention-hard"

TARGET\_FPU = "hard"

meta

meta-poky

meta-yocto-bsp = "scarthgap:200d12b6a58ad961d60a7774ca0f7a9d29498724"

meta-oe

meta-python

meta-networking

meta-filesystems = "scarthgap:72018ca1b1a471226917e8246e8bbf9a374ccf97"

meta-esme-custom = "scart:d22182cc5355069b5095a24987a9c9669b2d95c3"

meta-raspberrypi = "scarthgap:6df7e028a2b7b2d8cab0745dc0ed2eebc3742a17"

meta-esme-images = "scart:d22182cc5355069b5095a24987a9c9669b2d95c3"

workspace = "<unknown>:<unknown>"

Sstate summary: Wanted 7 Local 0 Mirrors 0 Missed 7 Current 295 (0% match, 97% complete)################### | ETA: 0:00:00

Removing 6 stale sstate objects for arch arm1176jzfshf-vfp: 100% |###################################################| Time: 0:00:00

Removing 1 stale sstate objects for arch raspberrypi0\_wifi: 100% |###################################################| Time: 0:00:00

14) Mise à jour de la version préférée de libgpiod

on se rend dans notre dossier de build du projet yocto et on execute la cmd bitbake-layers show-recipes libgpiod :

leeroy@leeroy-VirtualBox:~/w/build-scarthgap$ bitbake-layers show-recipes libgpiod

NOTE: Starting bitbake server...

Loading cache: 100% |#################################################| Time: 0:00:01

Loaded 4686 entries from dependency cache.

Parsing recipes: 100% |###############################################| Time: 0:00:00

Parsing of 2779 .bb files complete (2778 cached, 1 parsed). 4687 targets, 438 skipped, 0 masked, 0 errors.

WARNING: No bb files in default matched BBFILE\_PATTERN\_meta-esme-images '^/home/leeroy/w/src/esme-A3S8/meta-esme-images/'

Summary: There was 1 WARNING message.

=== Matching recipes: ===

libgpiod:

meta-oe 2.1.2

meta-oe 1.6.4

on edit le dossier conf/local.conf dans build-scarthgap et on ajoute la ligne PREFERRED\_VERSION\_libgpiod = "1.6.4"

15) dans la recipe on modifie la ligne du do-install comme ceci :

oe\_runmake install INSTALL\_DIR=${D}