Y.N.O.V.

PROJET DE MASTER EN PARTENARIAT AVEC THALÈS

Intégration d'un driver sur une OpenRex Basic

Auteurs :
Alain AIT-ALI
Martin LAPORTE
Clément AILLOUD
Romain PETIT

Superviseurs :
David COUÉ

Rapport hebdomadaire présentant l'avancement du projet sur l'intégration d'un driver de la Raspi Cam v2 sur un imx6 OpenRex Basic

au sein du

Département Aéronautique & Systèmes Embarqués

23 janvier 2018

Table des matières

Table des figures

Chapitre 1

Organisation de l'équipe et planning

1.1 Organisation de l'équipe

Afin de se concentrer sur la "Solution n°2" lundi 8 nous nous sommes tous lancés dans l'analyse du code c du driver et du déroulé de son fonctionnement. Le code réparti entre chacun, nous avons cherché à comprendre les utilité des structures et de l'organisation d'un client kernel. Nous avons donc chacuns poursuivit les appels aux fichiers systemes, en mutualisant les informations oralement.

Grâce à cette agile organisation nous avons pu préciser la source du problème avant d'en chercher la solution.

Par la suite Romain suivi par Alan se sont lancés dans le second objectif d'évolution, la compilation d'un noyau en version 4.14. Ce afin de précéder le portage du driver à ce kernel.

1.2 Planning

Ci-dessous, un planning des tâches effectuées en 2018.

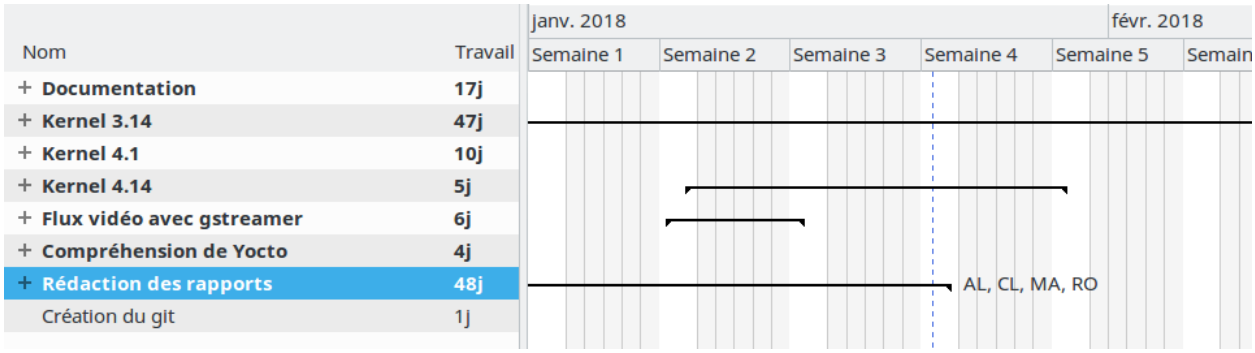


FIGURE 1.1 – Avancement général du projet

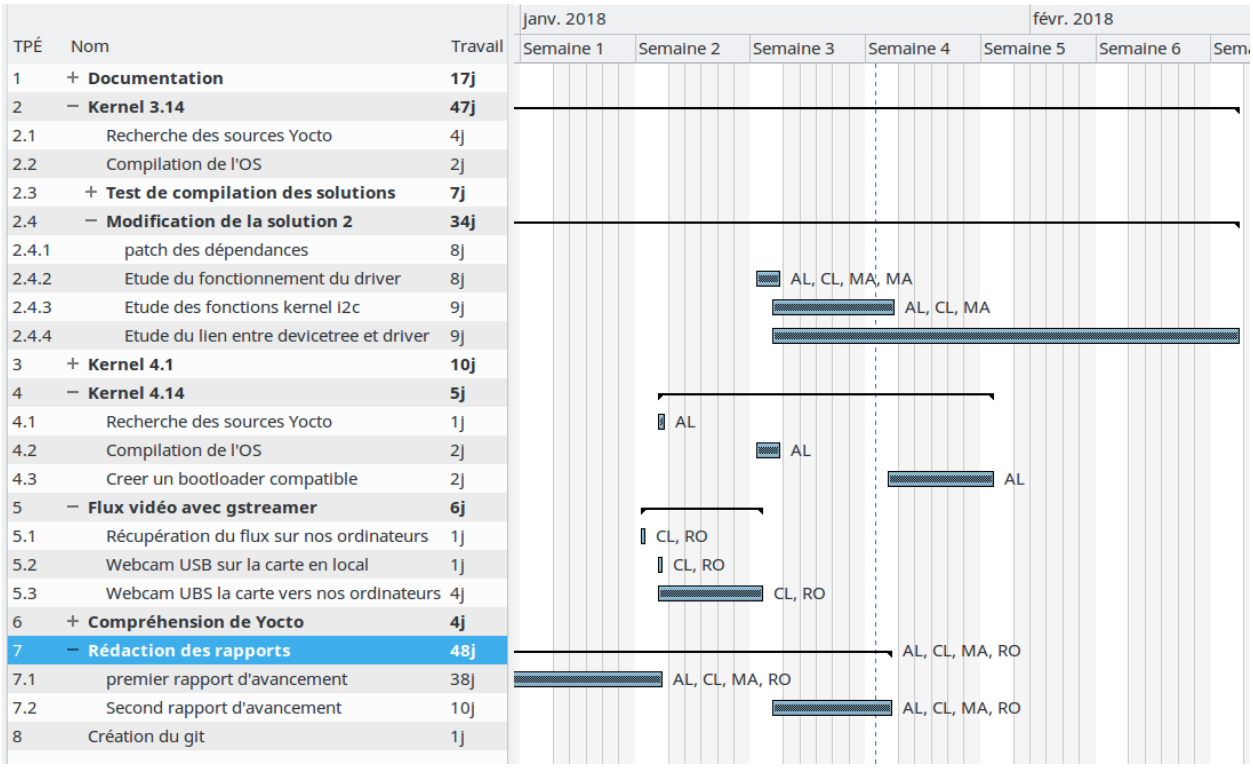
AL = Alan Ait-Ali

MA = Martin LAPORTE

RO = Romain Petit

CL = Clément Ailloud

La totalité du planning est accessible sur notre git, sur la branche "presentation" à l'emplacement "meta-openrexpicam/presentation2/gantthales.planner" sous forme de fichier à ouvrir avec le logiciel "planner" (sudo aptitude install planner).



TPÉ	Nom	Travail
1	+ **Documentation**	**17j**
2	− **Kernel 3.14**	**47j**
2.1	Recherche des sources Yocto	4j
2.2	Compilation de l'OS	2j
2.3	+ **Test de compilation des solutions**	**7j**
2.4	− **Modification de la solution 2**	**34j**
2.4.1	patch des dépendances	8j
2.4.2	Etude du fonctionnement du driver	8j
2.4.3	Etude des fonctions kernel I2c	9j
2.4.4	Etude du lien entre devicetree et driver	9j
3	+ **Kernel 4.1**	**10j**
4	− **Kernel 4.14**	**5j**
4.1	Recherche des sources Yocto	1j
4.2	Compilation de l'OS	2j
4.3	Creer un bootloader compatible	2j
5	− **Flux vidéo avec gstreamer**	**6j**
5.1	Récupération du flux sur nos ordinateurs	1j
5.2	Webcam USB sur la carte en local	1j
5.3	Webcam UBS la carte vers nos ordinateurs	4j
6	+ **Compréhension de Yocto**	**4j**
7	− **Rédaction des rapports**	**48j**
7.1	premier rapport d'avancement	38j
7.2	Second rapport d'avancement	10j
8	Création du git	1j

FIGURE 1.2 – Avancement détaillé du projet

Chapitre 2

Piste n° 1

Version du Kernel : 3.14

Code de travail : imx219.c (fichier de la solution 2 sur le rapport du 09/01/2017)

Suite à l'exploration de 3 pistes, nous avons choisi de nous concentrer sur la solution n°2 dont le driver compile sans erreur mais ne réussit pas le chargement au sein du kernel.

2.1 Erreur

Constat : Le driver apparait à la commande lsmod mais est inutilisé.

```
root@openrexpicam:~# modprobe imx219
root@openrexpicam:~# lsmod
Module                  Size  Used by
mxc_v4l2_capture       25109  0
ipu_bg_overlay_sdc      5242  1 mxc_v4l2_capture
ipu_still               2312  1 mxc_v4l2_capture
ipu_prp_enc             5872  1 mxc_v4l2_capture
ipu_csi_enc             3743  1 mxc_v4l2_capture
v4l2_int_device         2913  2 ipu_csi_enc,mxc_v4l2_capture
ipu_fg_overlay_sdc      6068  1 mxc_v4l2_capture
imx219                  7888  0
mxc_dcic                6543  0
galcore               225000  0
evbug                   1871  0
```

FIGURE 2.1 – Chargement du module

Après recherche d'indices pour le debug, on retient le message d'erreur suivant :

```
imx219 1-0064: IMX219: missing platform data!
```

FIGURE 2.2 – Debug probe platform data

Résultat de l'analyse : l'erreur provient des premières lignes de la première fonction probe du driver. La sécurité testant le premier paramètre d'appel du driver s'est déclanchée. le paramètre est une structure contenant des informations sur les périphériques i2c. Il y a donc problème de compatibilité entre le driver imx219 et le gestionnaire i2c. Selon l'avis de notre professeur de yocto qui a été appuyé par nos recherches, le driver en question utilise les interfaces de fonctionnement nommées platform_data, technologie remplacée progressivement depuis 2011 par le device-tree et sa gestion des compatibilités.

2.2 Solution

Aujourd'hui mardi 16 janvier nous avons cherché à porter le driver vers une compatibilité avec le device-tree.

Ajout de la structure type of_device_id dans le driver

```
1  static const struct of_device_id imx219_of_match[] = {
2    { .compatible = "sony,imx219", .data = 0 },
3    {}
4  };
5
6  MODULE_DEVICE_TABLE(of, imx219_of_match);
```

Completion de la structure de manipulation du driver

```
1  static struct i2c_driver imx219_i2c_driver = {
2    .driver = {
3      .name = "imx219",
4      .of_match_table = of_match_ptr(imx219_of_match),
5    },
6    .probe    = imx219_probe,
7    .remove = imx219_remove,
8    .id_table = imx219_id,
9  };
```

Suppression de la sécuritée platform data

```
1    if (!ssdd) {
2      dev_err(&client->dev, "IMX219: missing platform data!\n");
3      return -EINVAL;
4    }
```

2.3 Conclusion

Une nouvelle erreur est apparue cependant les travaux qui concernent le device tree ont fonctionné correctement car on peut remarquer au démarage que le driver est chargé automatiquement.

2.4 Erreur

Au chargement de la nouvelle image nous obtenons les erreurs suivantes :

```
imx219 1-0064: Error -19 getting clock
i2c 1-0064: Driver imx219 requests probe deferral
```

FIGURE 2.3 – Debug probe clk

L'erreur est relative à la fonction getclk du driver qui est une sécurité tout comme la fonction supprimer précedement. Elle confirme que nous avons des dificultés à communiquer avec l'i2c.

2.5 Solution

Notre strategie est de supprimer cette nouvelle fonction de securité, pour avoir plus d'informations sur la façon dont le driver fonctionne.

Suppression de la securité d'horloge

```
1     if (IS_ERR(priv->clk)) {
2         dev_info(&client->dev, "Error %ld getting clock\n",
3         PTR_ERR(priv->clk));
4         return -EPROBE_DEFER;
5     }
```

2.6 Conclusion

Le driver est maintenant chargé "correctement" par le kernel.

```
root@openrexpicam:~# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- UU -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@openrexpicam:~# lsmod
Module                  Size  Used by
mxc_v4l2_capture       25109  0
ipu_bg_overlay_sdc      5242  1 mxc_v4l2_capture
ipu_still               2312  1 mxc_v4l2_capture
ipu_prp_enc             5872  1 mxc_v4l2_capture
ipu_csi_enc             3743  1 mxc_v4l2_capture
v4l2_int_device         2913  2 ipu_csi_enc,mxc_v4l2_capture
ipu_fg_overlay_sdc      6068  1 mxc_v4l2_capture
imx219                  7716  1
mxc_dcic                6543  0
galcore               225000  0
evbug                   1871  0
root@openrexpicam:~#
```

FIGURE 2.4 – Chargement du module

On peut voir que l'i2c peut maintenant utiliser le driver imx219 et qu'il est utilisé par un autre module. Nous ne sommes toujours pas en mesure d'utiliser le driver car les informations sur son utilisation nous manquent. De plus de nouvelles erreurs apparaissent au chargement du driver.

2.7 Erreur

Au chargement du driver nous obtenons le message de debug kernel suivant :

```
Modules linked in: ipu_prp_enc ipu_csi_enc v4l2_int_device ipu_fg_overlay_sdc imx219(+) mxc_dcic galcore(O+) evbu
g
CPU: 0 PID: 156 Comm: udevd Tainted: G           O 3.14.61-yocto+g336bc38 #1
task: 863a2d00 ti: 864ae000 task.ti: 864ae000
PC is at soc_camera_power_init+0x0/0xc
LR is at imx219_probe+0xa8/0x424 [imx219]
pc : [<804a09b0>]    lr : [<7f053b0c>]    psr: 80010013
sp : 864afd88  ip : 864afcad  fp : 7f054680
r10: 86164220  r9 : 00000000  r8 : 86164200
r7 : 00000000  r6 : 00000000  r5 : 8655ce10  r4 : 7f053f48
r3 : 8655ce30  r2 : 00000020  r1 : 00000000  r0 : 86164220
Flags: Nzcv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment user
Control: 10c53c7d  Table: 164b0059  DAC: 00000015
Process udevd (pid: 156, stack limit = 0x864ae238)
Stack: (0x864afd88 to 0x864b0000)
fd80:                   8684edc8 80af0698 86164220 86164200 7f05451c 00000005
fda0: 00000000 80af0698 86164220 00000000 7f05451c 00000005 00000000 00000001
fdc0: 7f054680 80353e10 86164220 7f05451c 86164254 80a6b248 8684fd48 80354014
fde0: 00000000 7f05451c 80353f88 8035248c 860c185c 8614cb34 7f05451c 8643a180
fe00: 00000000 80353590 7f054400 7f056000 7f054644 7f05451c 7f056000 7f054644
fe20: 7f054638 8035460c 7f0544f8 7f056000 7f054644 80463cc4 864ae000 800088a0
fe40: 8bc278c0 10000000 8bc278c0 80a25598 00000000 80a95a80 00000000 800aa54c
fe60: 00012e80 800d85bc 00000001 800cedc8 00000093 00000004 00012e84 800d85bc
fe80: 00000001 8008b080 8684fd40 864aff58 00000001 864aff58 00000001 7f054644
fea0: 7f054638 8684fd48 8684fd40 8008b0b8 7f054644 00007fff 800885b4 864d0b40
fec0: 00000000 864aff58 864ae030 80088750 a0d65d6b 00000000 0000007c 80708710
fee0: a0d63000 00003780 000000b1 00000000 0b300002 00000000 00000000 00000000
ff00: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
ff20: 00000000 00000000 00000000 00000000 0000001c 00000000 00000005 76f11dc4
ff40: 0000017b 8000e5a4 864ae000 00000000 01665a48 8008ba38 a0d63000 00003780
ff60: a0d66280 a0d64f67 a0d65870 00001790 00001b90 00000000 00000000 00000000
ff80: 0000001e 0000001f 00000018 00000015 00000013 00000000 00000000 0164d008
ffa0: 00000000 8000e420 00000000 0164d008 00000005 76f11dc4 00000000 00000000
ffc0: 00000000 0164d008 00000000 0000017b 0164d2d8 00000000 0002ea98 01665a48
ffe0: 7ea18c00 7ea18bf0 76f0ac30 76e93760 60010010 00000005 1bf5e821 1bf5ec21
[<804a09b0>] (soc_camera_power_init) from [<7f053b0c>] (imx219_probe+0xa8/0x424 [imx219])
[<7f053b0c>] (imx219_probe [imx219]) from [<80353e10>] (driver_probe_device+0x118/0x24c)
[<80353e10>] (driver_probe_device) from [<80354014>] (__driver_attach+0x8c/0x90)
[<80354014>] (__driver_attach) from [<8035248c>] (bus_for_each_dev+0x68/0x9c)
[<8035248c>] (bus_for_each_dev) from [<80353590>] (bus_add_driver+0x148/0x1f0)
[<80353590>] (bus_add_driver) from [<8035460c>] (driver_register+0x78/0xf8)
[<8035460c>] (driver_register) from [<80463cc4>] (i2c_register_driver+0x30/0xb8)
[<80463cc4>] (i2c_register_driver) from [<800088a0>] (do_one_initcall+0xfc/0x158)
[<800088a0>] (do_one_initcall) from [<8008b0b8>] (load_module+0x1794/0x1f90)
[<8008b0b8>] (load_module) from [<8008ba38>] (SyS_finit_module+0x80/0x90)
[<8008ba38>] (SyS_finit_module) from [<8000e420>] (ret_fast_syscall+0x0/0x38)
Code: e1a00007 e3481092 ebfac115 eaffffe1 (e5912024)
V4L2 device 'Mxc Camera' on IPU1_CSI0 registered as video0
V4L2 device 'Mxc Camera' on IPU1_CSI1 registered as video1
ERROR: v4l2 capture: slave not found!
ERROR: v4l2 capture: slave not found!
---[ end trace faab8b2921e8f95b ]---
udevd[152]: worker [156] terminated by signal 11 (Segmentation fault)

udevd[152]: worker [156] failed while handling '/devices/soc0/soc.0/2100000.aips-bus/21a4000.i2c/i2c-1/1-0064'
```

FIGURE 2.5 – Dmsg

2.8 Reste à faire

- Determiner la cause du segmentation fault.
- Savoir comment récuperer le flux video.

Chapitre 3

Solution n° 2

Version du Kernel : 4.14

3.1 Kernel 4.14

La version 4.14 (une des plus recente chez fslc) du kernel contient un driver imx219 main line operationel. Notre idée était de porter le suport de la meta-voipac compatible avec un kernel 4.1 sur le nouveau kernel 4.14 pour pouvoir utiliser le driver.

3.2 Travail effectué

Modifier les sources du Kernel
Dans les recettes kernel chaque version du kernel est architecturé de la meme façon :

- Créer un fichier linux-voipac_4.14.bb.
- Créer un document linux-voipac-4.14 content un defconfig et un patch du device tree Openrex.

FIGURE 3.1 – Arborescence des fichiers

Le principe de cette méthode et de copier la version 4.1 et de remplacer les sources du kernel.

Modification du fichier

```
1       SRCBRANCH = "4.14.x+fslc"
2       LOCALVERSION = "-yocto"
3       SRCREV = "${AUTOREV}"
4       KERNEL_SRC ?=
            "git://github.com/Freescale/linux-fslc.git;protocol=git"
```

3.3 Erreur

Nous obtenons quelques erreurs relative à gstreamer et alsa pendant la compilation du nouveau kernel. Ces paquets n'étant pas essentiel à l'utilisation du driver nous les avons simplement enlevé les paquets.

L'étape de compilation réussite nous obtenons une erreur lors du boot

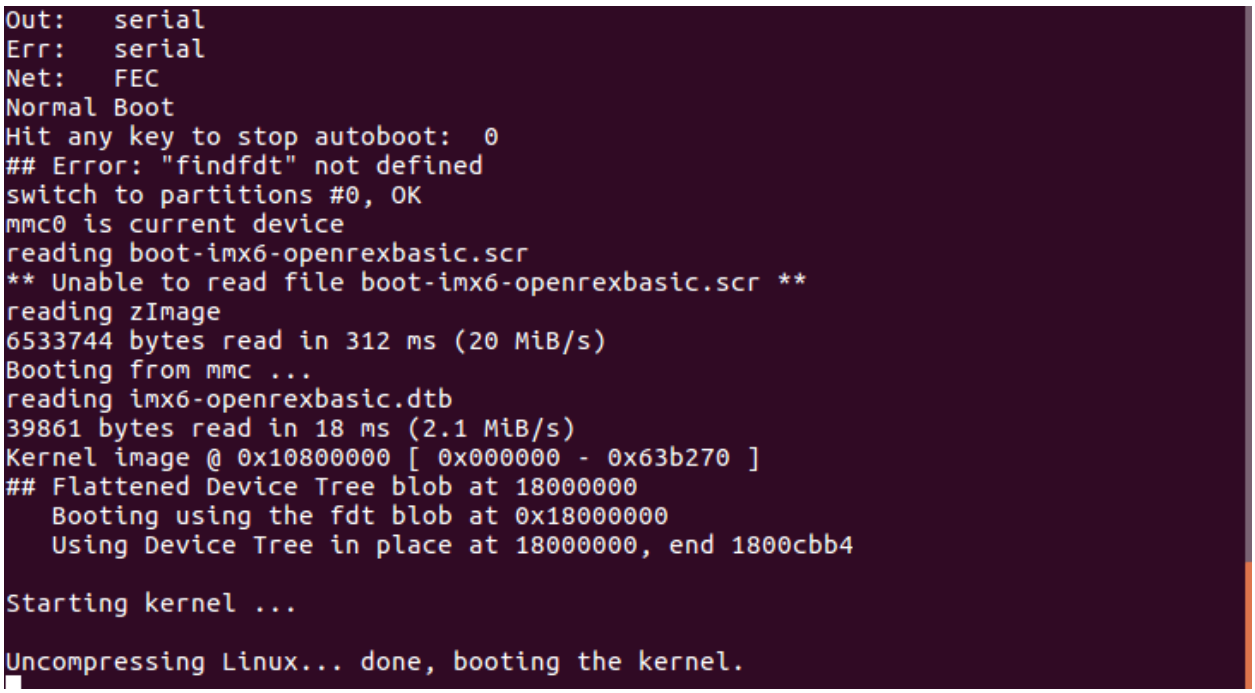


FIGURE 3.2 – Erreur du bootloader

3.4 Conclusion

U-boot n'arrive pas à lancer notre image, il y a donc une imcompatibilitée entre la version du kernel et celle du bootloader chargé dans la SPI flash. Pour parvenir à utiliser cette nouvelle image nous devons concevoir un nouveau bootlaoder en suivant la méthode présentée ici.

3.5 Reste à faire

- Créer un image 4.14 en se basant sur la meta-openrex.
- Créer un nouveau bootlaoder.
- Tester le nouveau bootlaoder.

Chapitre 4

Solution n° 3

Plateforme : Hummingboard (i.MX6)

Version du Kernel : 4.13

Source : http://git.armlinux.org.uk/cgit/linux-arm.git/commit/?h=csi-v6&id=e3f847cd37b007d55b76282414b
http://git.armlinux.org.uk/cgit/linux-arm.git/commit/?h=csi-v6&id=4bd8e1231a2e6eca6a65b565176ea97

Les sources étant sous forme de patch nous allons cette fois directement les intégrer à notre recette.

4.1 Contenu des sources

Kconfig : ajoute l'imx219 au menuconfig, pour demander la compilation du driver, utilisation :

1. Modifier le menuconfig
2. Récupérer le defconfig
3. Ajouter à notre recette.

Makefile : ajoute la compilation du driver (pourra être activée par le defconfig)

Imx219.c : driver qui contient l'ensemble des configurations spécifiques à la caméra.

Device-tree pour Hummingboard : annonce les périphériques avec lesquels le driver aura des interactions (i2c, csi, gpio, clk).

4.2 Problèmes rencontrés

Des libraires système sont inexistantes, nous les avons donc identifiées et ajoutées au système.

Les versions des librairies existantes ne sont pas compatibles avec notre fichier imx219.c. Nous utilisons une version de kernel 3.14 ou bien la version nécessaire est bien plus récente nous l'estimons à 4.12. Pour s'approcher de la versions 4.12 nous sommes actuellement entrain d'appliquer les patchs sur une version plus ressente soit 4.1.36 (Krogoth sous Yocto).

4.3 lien GIT

Vous trouverez ici un lien qui contient l'ensemble des patchs appliqués sur la version 3.14 du kernel : https://github.com/Alanaitali/meta-openrexpicam/tree/develop/gladwistor/recipes-kernel/linux

4.4 Source des patchs

1. http://git.armlinux.org.uk/cgit/linux-arm.git/commit/?h=csi-v6&id=e3f847cd37b007d55b76282414bfcf1
2. http://git.armlinux.org.uk/cgit/linux-arm.git/commit/?h=csi-v6&id=4bd8e1231a2e6eca6a65b565176ea97

4.5 Ajout de paquets

En prévision d'un usage ultérieur de v4l-utils et/ou gstreamer comme applicatif (user-space), nous avons essayé de les compiler. Cette compilation n'a abouti ni par compilation in-tree ni out-of-tree, ni par une recette Yocto. Par la suite nous avons ajouté ces paquets à notre image via la variable IMAGE_INSTALL.

```
1  DESCRIPTION = "Basic␣image␣openrexpicam"
2  LICENSE = "MIT"
3
4  inherit core-image
5
6  IMAGE_INSTALL += "␣\
7  ␣␣␣␣gstreamer␣\
8  ␣␣␣␣i2c-tools␣\
9  ␣␣␣␣gstreamer1.0-plugins-imx␣\
10 ␣␣␣␣gst1.0-fsl-plugin␣\
11 ␣␣␣␣v4l-utils␣\
12 "
```

4.6 Conclusion

La compilation du driver n'a pas fonctionné car les librairies kernel et celles requises par le driver ne sont pas compatibles. Une autre solution serait d'essayer la même compilation sur un kernel 4.1.

Chapitre 5

État de l'art de la récupération du flux vidéo

5.1 Methodologie par la Webcam USB

Nous nous aguérissons à l'utilisation de Video4Linux-utils, de plus nous avons trouvé un journal en ligne expliquant la récupération du flux vidéo d'une webcam USB avec V4L2-utils sur imx6. Dans un premier temps, nous avons capturé le flux d'une webcam USB intégrée à un de nos ordinateurs personnels afin de comprendre les commandes et paramètres. Ensuite nous avons récupéré une webcam USB lambda afin de tester les commandes sur la carte openrex et essayer de capturer le retour vidéo. Une fois fonctionnel nous passerons sur le flux vidéo de la Raspi Cam v2.

1. Récupérer le port qui est connecté la webcam USB grâce à la commande : **lsusb -t**
2. La webcam USB est sur le bus I2C n°1 avec comme identifiant de port 1.2.
3. Il est nécessaire d'écrire le numéro du bus et l'identifiant du port dans les fichiers bind et unbind. Le fichier bind permet de « lier » un appareil USB à son driver et ainsi le rendre visible pour le système tandis que le fichier unbind lui détache le driver de son appareil USB pour le « cacher » du système.
4. La webcam USB est bien reconnu par notre système qui affiche sa référence
5. Démarrer un flux vidéo sur notre webcam USB s'effectue avec la commande : **# gst-launch-1.0 v4l2src device="/dev/video0"! video/x-raw,w idth=640,height=480! autovideosink**
6. Nous sommes persuadés de la bonne capture du flux vidéo puisque lorsque nous la lançons, le retour de la commande se fige à :

```
1 New clock GstSystemClock
```

   C'est exactement le même endroit quand nous lançons la même commande sur notre propre ordinateur avec notre webcam intégrée. De plus une petite LED verte s'allume au moment de l'envoi de cette commande (cf. Figure **??**).

5.2 Problèmes

Notre principal problème pour l'instant est de récupérer ce flux vidéo puisqu'il est impossible de l'afficher directement dans le terminal, pour tester cela nous essayons de l'afficher sur nos ordinateurs par le réseau WiFi en protocole UDP.

5.3 Point d'amélioration

Pour commencer il est nécessaire de mettre l'image de l'openrex à jour afin de se connecter à un réseau WiFi ensuite récupérer ce flux vidéo par le réseau WiFi sur nos ordinateurs en UDP.

Finalement une fois que nous aurons réussi à afficher le flux vidéo de la webcam USB sur nos ordinateurs nous essayerons d'effectuer le même processus mais pour la Raspi Cam v2, qui se pilote par le CSI et non l'USB.

5.4 Sources

Lien du blog de la récupération du flux vidéo par la webcam USB :
http://jas-hacks.blogspot.fr/2014/04/imx6-gstreamer-imx-and-usb-webcam.html

Chapitre 6

Conclusion

De part l'avancée que nous avons actuellement et l'expérience accumulée au cours du projet il nous reste plusieurs pistes à explorer. Sans contre- indication particulière nous devrons être capables d'avoir des résultats dans quelques semaines. Nous ne sommes pas en capacité de dire avec précision à quelle date il sera possible de vous fournir un driver fonctionnel.

Pour tout conseil, piste à suivre ou erreur de notre part n'hésitez pas à revenir vers nous. De plus nous tenons à nous excuser pour le dernier rapport envoyé il n'était manifestement pas du tout adapté à votre demande.