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

22 janvier 2018

Table des matières

1 Organisation de l'équipe et planning **1**
1.1 Organisation de l'équipe 1

2 Piste n° 1 **2**
2.1 Erreur 2
2.2 Solution 3
2.3 Conclusion 3
2.4 Erreur 3
2.5 Solution 4
2.6 Conclusion 4

3 Solution n° 2 **6**
3.1 Fichiers sources 6
3.2 Travail effectué 6
3.3 Compilation de driver 6
3.4 Utilisation du driver sur la cible 7
3.5 Conclusion 8

4 Solution n° 3 **9**
4.1 Contenu des sources 9
4.2 Problèmes rencontrés 9
4.3 lien GIT 9
4.4 Source des patchs 9
4.5 Ajout de paquets 10
4.6 Conclusion 10

5 État de l'art de la récupération du flux vidéo **11**
5.1 Methodologie par la Webcam USB 11
5.2 Problèmes 11
5.3 Point d'amélioration 12
5.4 Sources 12

6 Conclusion **13**

Table des figures

2.1 Chargement du module 2
2.2 Debug probe 2
2.3 Debug probe 3
2.4 Debug probe 4

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 lancés dans l'annalyse du code c du driver et du déroulé de son fonctionnement.

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 afinde précéder le portage du driver à ce kernel.

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 de solutions, nous avons choisi de nous concentrer sur la solutions n°2 dont le driver compile sans erreur mais ne réussit pas le chargement.

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

Résultat de l'annalyse : 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 concernant 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 erreur suivantes :

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

FIGURE 2.3 – Debug probe

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

2.5 Solution

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

Suppression de la securitée 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 charger "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 – Debug probe

On peut voir que l'i2c peut maintenant utiliser le dirver 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 l'utilisation du driver nous manque. Deplus de nouvelles erreurs apparaissent au chargement du driver.

Chapitre 3

Solution n° 2

Version du Kernel : 4.14
Code de travail : recette yocto des paquets à inclure.

3.1 Fichiers sources

Ce driver contient les fonctions de configuration de l'imx219 : on, off, gain, couleurs, taille de l'affichage. Il repose sur les couches de driver i2c et v4l. La configuration du périphérique imx219 effectuée par v4l-utils est automatique lors du chargement du module. Elle s'appuie sur les trois couches de driver imx219>v4l>i2c. Le flux vidéo configuré peut ensuite être récupéré par Gstreamer.

3.2 Travail effectué

- Gstreamer permettrait de streamer le flux sur le réseau à travers le protocole UDP alors que v4l-utils permettrait plutôt d'enregistrer le flux vidéo.
- Création d'une recette

3.3 Compilation de driver

Extrait de la recette : récupère le fichier imx219.c, le compile par la cross tool-chain et l'installe

```
1  SUMMARY = "driver_imx219"
2  SECTION = "examples"
3  LICENSE = "MIT"
4  LIC_FILES_CHKSUM =
       "file://${COMMON_LICENSE_DIR}/MIT;md5=c2b5f7071fdde268bf46ace1546f3c4b"
5
6  SRC_URI = "file://imx219.c"
7
8  S = "${WORKDIR}"
9
10 do_compile() {
11          ${CC} imx219.c -o imx219
12 }
13
14 do_install() {
15          install -d ${D}${bindir}
16          install -m 0755 imx219 ${D}${bindir}
17 }
```

3.4 Utilisation du driver sur la cible

Vérification de la présence du driver dans l'arborescence :
Chargement du module imx219 :
Comme nous pouvons le voir le driver n'est pas encore utilisé par v4l (Used by 0).

Chargement du module à l'adresse I2C de l'imx219 :

Erreur sur le bus i2c sur laquelle nous travaillons actuellement. Récupération du flux vidéo en local via Gstreamer :

 # gst-launch-1.0 v4l2src device="/dev/videoX"! video/x-raw,width=640,height=480! autovideosink

X' étant le numéro correspondant au numéro du driver vidéo du port CSI-2

Actuellement nous cherchons l'utilisation des commandes gst-launch-1.0 et des fichiers de configurations v4l2src (cf. Chapitre 5).

3.5 Conclusion

Cette solution est notre piste la plus concrète. En effet la compilation a réussi, nous cherchons à comprendre comment utiliser le driver à travers v4l. Une fois cette tache effectuée nous travaillerons sur le portage des versions du kernel (3.14 $\rightarrow$ 4.1).

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.