UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Smile OPEN SOURCE SOLUTIONS

Rapport de stage M2

Auteurs :
Nizar ABAK-KALI
11290569

Tuteur :
M. Pierre LAMOT
M. Fabien DUTUIT

29 août 2017

Résumé du stage

Ce stage s'est déroulé dans la société de service Smile et plus particulièrement le département Smile ECS, responsable de l'offre embarqué et IoT [1]. Ainsi, mon stage s'est d'abord déroulé sous la tutelle de Pierre Lamot, expert en traitement d'images et streaming chez Smile ECS, puis sous celle de Fabien Dutuit, expert traitement de signal.
Au cours de ce stage, j'ai eu l'opportunité de travailler sur une grandes variétés de technologie et ainsi d'avoir la chance de pouvoir engranger un maximum de connaissances et d'expériences dans le monde de l'embarqué que je compte utiliser plus tard.

Ce stage ce découpe en deux phases. Une première concentré sur l'étude du protocole MPEG-TS ainsi que de son implémentation dans Gstreamer [2] puis de l'écriture d'un patch décri plus tard dans le rapport, permettant le multipléxage de données non typé dans un flux vidéo. Puis dans un second temps, mon stage c'est plus concentré sur le projet SisselBox [3], où je devais dans un premier temps mettre à jour le système de provisionnement, puis implémenter une application afin de tester mon patch.

1. IoT : Internet of Things ou objets connectés
2. 4.2
3. 2.1

Table des matières

Table des figures

Remerciements

A l'issue de ce stage, je souhaite remercier l'équipe technique de Smile ECS [4] de m'avoir accordé une place dans leur département d'embarquer et de m'avoir permis d'effectuer mon stage de fin d'étude au sein d'une équipe compétente et conviviale. Je remercie également M. Pierre Lamot, expert technique ECS, pour m'avoir soutenu et motivé pendant le stage, et tant apporté.Je souhaite également remercier M. Fabien Dutuit, expert technique ECS, pour m'avoir aidé dans les difficultés que j'ai pu rencontrer. Je souhaite remercier également toute l'équipe de Smile ECS qui m'a supporté durant ces 6 mois, mais qui m'a avant tout apporté énormément aussi bien sûr le plan humain que technique.

4. Embdeed Computer Science

Introduction

Dans le cadre de mon master en informatique embarqué à l'université paris 8, il est nécessaire d'effectuer un stage de fin d'étude d'une durée de 6 mois en entreprise. Après avoir prospecté plusieurs offres de stage et avoir effectué plusieurs entretient, mon choix s'est finalement porté vers la société Smile et plus particulierement le département embarqué. J'ai donc rejoint l'équipe de Smile ECS, filiale spécialisée dans l'infor-matique embarquée et le traitement des images à destination des industriels. Mon stage s'est donc déroulé du 20 février au 18 aout 2017 au sein de Smile ECS, à Asnieres.

Chapitre 1

Présentation de l'entreprise

1.1 Smile

Fondé en 1991, Smile est devenu dès 1995 un acteur du web réputé, maîtrisant l'architecture, les technologies et les outils qui permettent de construire les plus grandes plateformes de l'Internet. Depuis 2001, Smile est intégrateur de solutions open source, c'est-à-dire que le cœur de métier de Smile est la construction de systèmes d'information et de plateformes web intégrant les meilleures solutions open source du marché. Smile mène une forte action de veille afin d'identifier les solutions open source les plus matures et les plus pérennes, qui apporteront un réel bénéfice de compétitivité pour les entreprises. Smile met à disposition un échantillon de cette expertise au travers de livres blancs, librement diffusés, qui sont devenus des références dans leurs domaines. Premier intégrateur spécialisé sur les solutions open source, Smile a été choisi à de nombreuses reprises par les plus grandes entreprises et administrations pour déployer ces solutions dans le cadre de projets stratégiques. Autour du cœur de métier qu'est l'ingénierie, Smile propose une palette de services étendue, qui lui permet de prendre en charge un projet dans sa globalité : consulting en amont et en accompagnement des projets, agence interactive intervenant tant en création et webdesign qu'en conseils éditoriaux, stratégiques et e-marketing, tierce-maintenance applicative (TMA), formation, support et maintien en conditions opérationnelles, et enfin hébergement et exploitation. Smile possède également un rayonnement international grâce à ses dix-sept agences réparties dans 9 pays.

Smile organise également une veille technologiques importantes sur l'open source, avec la rédaction de nombreux livres blancs par les différents consultants sur des sujets variés allant des solutions web d'e-commerce, en passant par les middlewares et finissant par l'embarqué. Il existe actuellement une vingtaine de livres blancs,

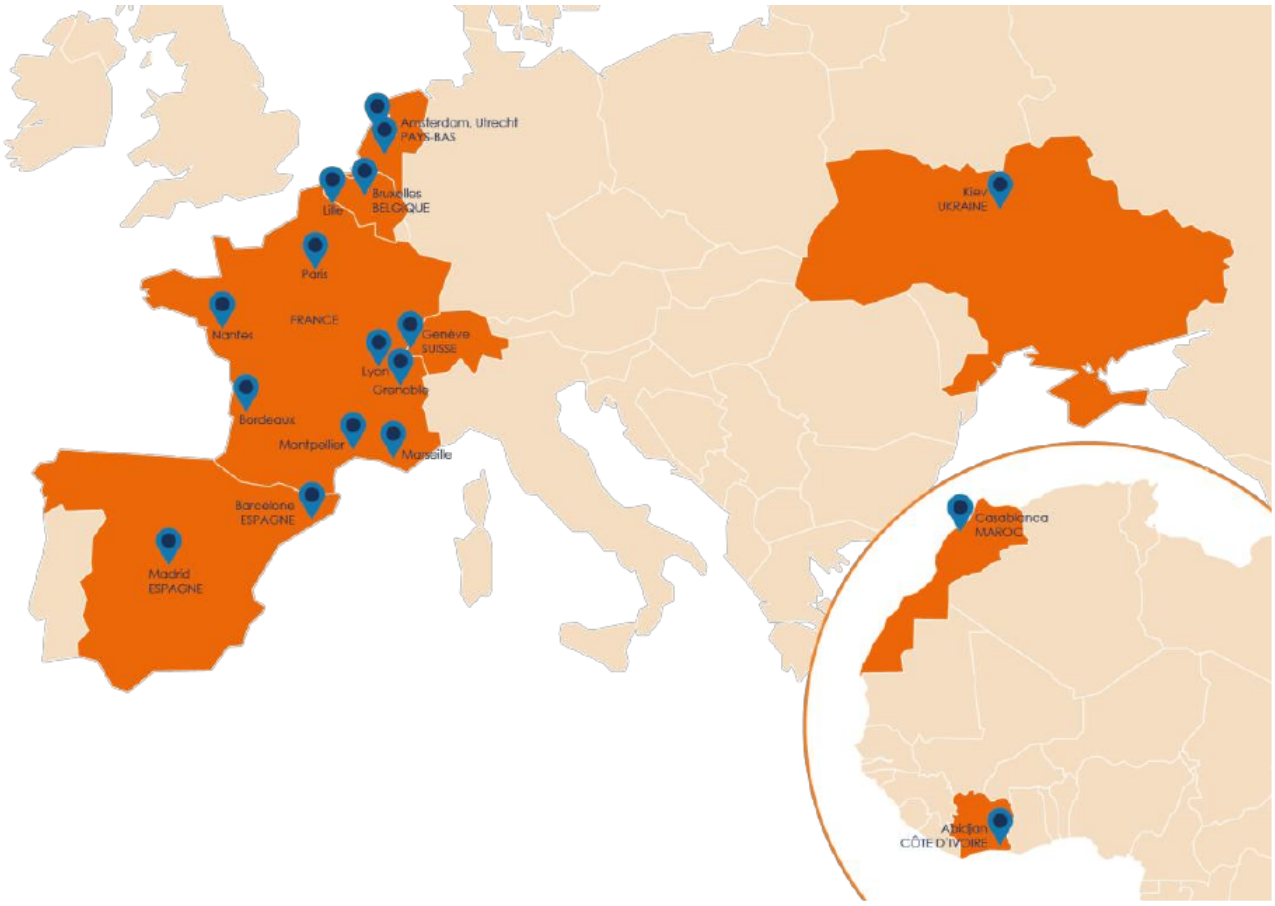

FIGURE 1.1 – Agences Smile

renouvelés tous les 2-3 ans pour inclure les nouveautés des différentes technologies. Dès 2001, Smile commence à construire son expertise des solutions open source : un choix d'avenir que beaucoup de ses concurrents n'osent pas alors entreprendre. A partir de 2004, les grandes entreprises adoptent de plus en plus souvent les solutions open source. Smile occupe une position solide de leader sur ce marché qui décolle, élargissant progressivement son offre vers de nouveaux domaines – CMS, Portails, e-Commerce, Décisionnel, Infrastructure, ERP - et créant des services associés : agence média, tierce maintenance, exploitation, système. En 2007, Smile affiche plus de 50% de croissance et inaugure trois nouvelles agences à Lyon, Nantes et Bordeaux. En 2008, Smile compte plus de 270 collaborateurs et poursuit sa croissance en se développant à l'international, notamment à Kiev en Ukraine. En 2009, Smile compte 320 collaborateurs dans le monde et poursuit son extension internationale. En juin 2009, Smile intègre Cometa Technologies S.L., basée à Barcelone et compte alors 8 agences dans le monde. En 2011, Smile regroupe 500

collaborateurs et ouvre ses agences à Bruxelles, Utrecht et Amsterdam.

En 2013, Smile compte 700 collaborateurs et 17 agences. 2014 est une année importante pour Smile, avec plus de 20% de croissance sur ces cinq dernières années et près de 700 collaborateurs dans le monde. Au fil de ces années, Smile est devenu un acteur incontournable de l'open source, leader en France et en Europe. En 2015, Smile annonce être en négociations exclusives avec la société Open Wide en vue d'un rapprochement. Ceci afin d'élargir son offre et renforcer son leadership.

1.2 Smile ECS

Smile ECS, ex-OpenWide est une société spécialisée dans le domaine de l'open source, racheté par Smile. J'ai décidé d'intégrer cette entreprise pour mon projet de fin d'étude à l'université Paris 8 afin d'approfondir mes connaissances dans le domaine du logiciel libre, et celui de l'embarqué. Les technologies Open source devenant de plus en plus prédominantes dans la paysage informatique moderne, intégrer une société spécialisée dans cette technologie était une façon pour moi de donner un cap sur l'orientation de ma carrière professionnelle. Ce rapport explicitera mon travail durant ce stage de fin d'étude, qui a duré six mois, dont le sujet a été d'améliorer le système de transmission des données, afin de pouvoir transmettre un flux vidéo, de caméras de sécurité, avec des métadonnées calculées depuis ce flux (exemple : positions des visages détectés). Puis dans un second temps, recupérer ce flux afin de le stocker pour un visionnage ultérieur. Après une explication du cadre du stage, j'essaierai de développer les technologies utilisées lors de ce stage.

Chapitre 2

Présentation du sujet

Mon sujet s'inscrit dans un projet developpé en interne, le projet SisellBox. Ansi, je vais présenter dans un premier temps le projet, puis dans un second temps expliquer ou je m'inscrit dans le projet et qu'est ce que j'y apporte.

2.1 Le projet SisellBox

FIGURE 2.1 – image de la SisellBox

La Sisell Box est un enregistreur développé par Open Wide permettant de déployer une solution complète de vidéo surveillance rapidement. Ce système peut être utilisé dans des contextes de mission très différents : surveillance longue durée, protection de sites sensibles, capture d'images hautes qualité, ainsi que surveillance ponctuelle. Fonctionnant sur batterie et créant un réseau wifi maillé de façon autonome, elle ne nécessite que peu d'intervention humaine pour l'utilisation. Il suffit de brancher la batterie, brancher les caméras et elle est autonome. L'utilisation d'algorithmes performants lui permet de détecter et d'analyser tout événement (intrusion, stationnarité, mouvement de foule) sur la base de critère précis et configurables. Le projet Sisell se découpe en partie, deux partie, d'une part le "streamer", prenant en entrée une source vidéo "live"(par exemple une caméras de sécurité) et fourni un flux de composé de vidéo et de métadonnées. La deuxième partie, le "recorder" récupère les flux vidéo et de méta- données généré par le streamer. Les méta-données sont enregistrées en base de donnée ainsi que divers informations (localisation des enregistrements vidéo etc.)

2.1.1 SisellBox Streamer

La partie streamer de Sisell Box gère la capture vidéo, l'analyse de l'image et son émission sur un réseau.

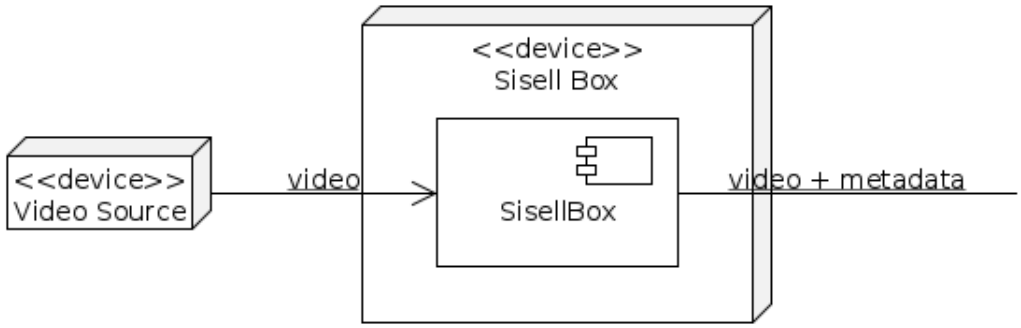

FIGURE 2.2 – Cas d'utilisation du streamer

Afin de faciliter l'inter-opérabilité de cette application, il nous faut se baser sur des protocoles de communications standard. La norme ONVIF qui a pour objectif d'uniformiser les services de vidéosurveillance préconise d'utiliser le protocole RTSP pour les émissions de flux vidéo. Le protocole RTSP (Real Time Streaming Protocol) est un protocole destiné aux systèmes de streaming multimédia. Il permet de contrôler le média à distance (jouer, mettre en pause, enregistrer, fin de

session etc.). Cela permet donc de ne pas se soucier de l'implémentation de ces fonctionnalités côté serveur et permettre le contrôle du flux via un lecteur standard type VLC sans développement supplémentaire. Le RTSP ne gère par contre pas la partie transport des données. Celles-ci sont donc émises via le protocole de transport RTP (Real-time Transport Protocol). Ainsi furent les choix technologiques pris avant mon arrivé sur le projet. Or, il s'est avéré que la norme ONVIF était trop compliqué à implémenté avec Gstreamer à des limitations technologiques. C'est ainsi qu'il a été choisi d'encapsuler le flux de données au travers de MPEG-TS puis de le transmettre via RTP.
L'image capturée par le streamer peut ensuite être analysée par une ou plusieurs brique(s) d'analyse. Une brique d'analyse est un élément interchangeable effectuant un traitement particulier sur l'image afin d'en extraire des informations (détection de visage, reconnaissance de forme, mouvement, etc..). Actuellement, seul quelques algorithmes d'analyse (tel que la détection de visage) ont été implémentés pour la nouvelle architecture.
Les informations extraites par la brique d'analyse sont mises sous la forme de métadonnées. Chaque trame vidéo se voit donc associé d'un trame de métadonnée. Celle-ci peut être vide si aucune information n'a été extraite par l'algorithme d'analyse vidéo mais est néanmoins présente. Le flux vidéo peut être encoder avec les formats H.264 et MJPEG, afin de les transmettre sur le réseau. Le streamer doit aussi gérer des sources non-live (fichiers vidéo) de différents formats (*.mkv, *.ts, *.avi).
Du côté de l'implémentation, les briques algorithmiques de traitement d'image sont développées en C++ et en Vala. Le reste est développé en Python et en utilisant les bindings Python pour GStreamer afin d'utiliser le framework et ses plugins développés en C.

2.1.2 SisellBox Recorder

Le recorder récupère les flux vidéo et de méta- données généré par le streamer. Les méta-données dont enregistrées en base de donnée ainsi que divers informations (localisation des enregistrements vidéo etc.). La vidéo est multiplexé avec les méta-données (en utilisant la piste de sous-titre) puis enregistrée sur le disque. Le recorder peut également récupérer des images (format jpeg) d'une zone de la vidéo récupérée (par exemple un visage détecté en amont) via un élément « Snapshoter ».

Le recorder est capable de :

— Recevoir des flux vidéo et de méta-données et compatible avec au minimum :
- protocoles réseau : RTSP/RTP
- codecs : MJPEG et H.264

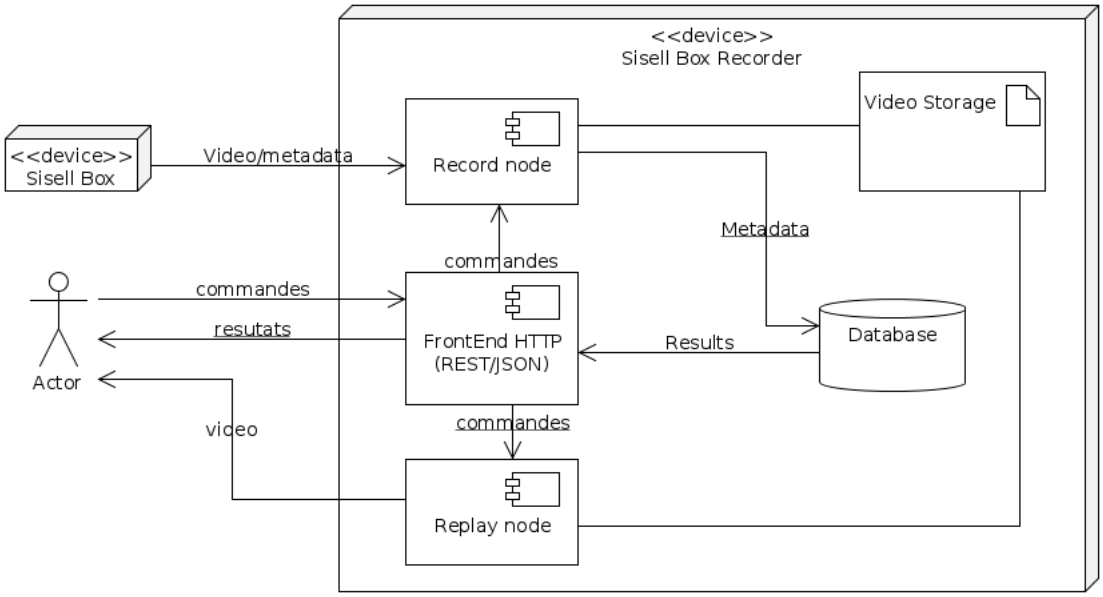

FIGURE 2.3 – Cas d'utilisation du recorder

— Enregistrer les vidéos sur disque dans un format permettant de les relire ainsi que les méta-données associés, dans notre cas, nous utilisons un conteneur MPEG-TS (fichier *.ts) avec les codecs H.264 et MJPEG pour la vidéo et du binaires formatés pour les méta-données

— Stocker les métadonnées en base de donnée

— Effectué des enregistrements sur des événements. Les événements pouvant être de type :

- Présence de méta-données
- événement de type particulier (présence de personne, ...)

Pour la base de donnée, nous utilisons PostgreSQL et Postgis (extension de PostgreSQL pour la manipulation d'informations géographique) sur une machine Debian distante. Pour la phase de développement, cette machine est virtualisée via une VM VirtualBox. Les lectures/écritures dans la base de donnée depuis le recorder se font grâce un mapping objet-relationnel (ORM) permettant de manipuler une base de donnée relationnelle de la même manière qu'une base de donnée orienté objet. Pour cela nous utilisons l'outil libre SQLAlchemy.

Chapitre 3

Les problématiques posées

Ainsi, après avoir explicité le contexte du projet, il est temps que je décrive les différentes problématique qui m'ont été posé durant ce stage. Dans un premier temps, le projet repose sur le framework multimédia Gstreamer [1], n'ayant jamais rencontré cet technologie auparavant, il m'as fallu un temps de d'auto-formation afin d'acquerir les bases pour pouvoir commencer le developpement de plugins simples. Ensuite, il m'as fallut étudier les plugins qui implémente le multiplexage et démultiplage des données dans les fichiers MPEG-TS afin de pourvoir patcher le code.

1. cet technologie est décrite plus en détaille ici 4.2

Chapitre 4

Présentation de l'environnement de travail

Dans le cadre de se stage j'ai eu l'occasion de travailler une plusieurs technologie, beaucoup d'entre-elles m'était inconnue. Ainsi , je vais explicité les différentes technologie sur lesquelles j'ai travailler.

4.1 Configuration

4.2 Gstreamer

4.3 OpenCV

4.4 Protobuf et Grpc

4.5 Outils de provisionnement

Chapitre 5

Travails effectués

5.1 Etude de MPEG-TS

– > Pourquoi ?
– > qu'est ce que c'est
– > modification des plugins pour le mux/demux

5.2 Projet SisselBox

– > Mis à jour du système de provisionnement
– > Etude de l'existant
- Sisell elements
- sboxstreamer
– > Création d'un application Test : MPEGTS[video+metadata]
- pipeline - probleme de perte de flux par perte de synchro

Chapitre 6

Conclusion

(à peu près 1 page)

? ? ? (conclusion du stage)

Quelques chose de personnel : ton sentiment, ce que tu en as tiré, ? ?

Ouverture vers la suite :

Chapitre 7

Bibliographie

How To Install and Use PostgreSQL on Ubuntu 16.04 DigitalOcean https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql- .

GStreamer-1.8.1 rtsp server and client on ubuntu Telecom R & D https://telecom.altanai.com/2016/05/20/gstreamer-1-8-1-rtsp-server-and-client-on-ub .

gstreamer - Plugins : "ugly" and "bad" - Ask Ubuntu https://askubuntu.com/questions/468875/plugins-ugly-and-bad .

grpc http://www.grpc.io/docs/quickstart/cpp.html .

grpc http://www.grpc.io/docs/quickstart/python.html .

Projects · Dashboard · GitLab https://gitlab.openwide.fr/ .

gst-plugins-base/gst-libs at master · GStreamer/gst-plugins-base · GitHub https://github.com/GStreamer/gst-plugins-base/tree/master/gst-libs .

Installing OpenCV 2.4.9 in Ubuntu 14.04 LTS – Sebastian Montabone http://www.samontab.com/web/2014/06/installing-opencv-2-4-9-in-ubuntu-14-04-l .

Installing OpenCV on Debian Linux | Indranil's world https://indranilsinharoy.com/2012/11/01/installing-opencv-on-linux/ .

grpc/INSTALL.md at master · grpc/grpc · GitHub https://github.com/grpc/grpc/blob/master/INSTALL.md .

Gstreamer basic real time streaming tutorial | Einar Sundgren http://www.einarsundgren.se/gstreamer-basic-real-time-streaming-tutorial/ .

GStreamer Debugging - RidgeRun Developer Connection https://developer.ridgerun.com/wiki/index.php/GStreamer_Debugging#Use_standard_GStreamer_debug_output_with_filter .

manual.pdf https://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/manual.pdf .

open-wide / sisellbox-ng · GitLab https://gitlab.openwide.fr/open-wide/sisellbox-ng .

Accueil - Smile Intranet https://intranet.smile.fr/portal/fr/web/guest/home .

Webmail - BlueMind https://bluemind.smile.fr/webmail/?_task=mail&_mbox=INBOX&_refresh=1 .

Vim as a Python IDE | Unlogic http://unlogic.co.uk/2013/02/08/vim-as-a-python-ide .

Real Time Streaming Protocol - Wikipedia https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol .

toto.html file:///home/niaba/Documents/sisellbox-ng/doc/architecture/toto.html .

Troubleshooting GStreamer https://gstreamer.freedesktop.org/documentation/frequently-asked-questions/troubleshooting.html .

Gstreamer cheat sheet - MyLabWiki http://wiki.oz9aec.net/index.php/Gstreamer_Cheat_Sheet .

pygst-tutorial.pdf https://brettviren.github.io/pygst-tutorial-org/pygst-tutorial.pdf .

Instantiable classed types : objects : GObject Reference Manual https://developer.gnome.org/gobject/stable/gtype-instantiable-classed.html .

GObject Reference Manual : GObject Reference Manual https://developer.gnome.org/gobject/stable/ .

Mpeg TS helper library : GStreamer Bad Plugins 1.0 Library Reference Manual https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-bad-libs/html/mpegts.html .

gst-plugins-bad/gst-libs/gst/mpegts at master · GStreamer/gst-plugins-bad https://github.com/GStreamer/gst-plugins-bad/tree/master/gst-libs/gst/mpegts .

Diapo : ces bricolages de génie qui donnent envie d'acheter un Raspberry Pi http://www.tomshardware.fr/articles/raspberry-pi-bestof-bricolage-hack,1-63056.html .

MPEG transport stream - Wikipedia https://en.wikipedia.org/wiki/MPEG_transport_stream#PCR .

Novacut/GStreamer1.0 - Ubuntu Wiki https://wiki.ubuntu.com/Novacut/GStreamer1.0 .

rubenrua/GstreamerCodeSnippets : Gstreamer Code Snippets in C, Python and Shel https://github.com/rubenrua/GstreamerCodeSnippets .

Adding Properties https://gstreamer.freedesktop.org/documentation/plugin-developme basics/args.html .

Example GStreamer Pipelines - IGEP Community Wiki http://labs.isee.biz/index.php/Example_GStreamer_Pipelines .

Base MPEG-TS descriptors : GStreamer Bad Plugins 1.0 Library Reference Manual https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-bad-libs html/gst-plugins-bad-libs-Base-MPEG-TS-descriptors.html .

Creating special element types https://gstreamer.freedesktop.org/documentation/plugin-development/element-types/index.html .

GStreamer and in-band metadata - RidgeRun Developer Connection https://developer.ridgerun.com/wiki/index.php/GStreamer_and_in-band_metadata .

Novacut/GStreamer1.0 - Ubuntu Wiki https://wiki.ubuntu.com/Novacut/GStreamer1.0#audio.2Fx-raw.2C_video.2Fx-raw .

cerbero-docs/start.md at master · centricular/cerbero-docs https://github.com/centricular/cerbero-docs/blob/master/start.md .

centricular/cerbero : GStreamer's Cerbero forked to add Meson and MSVC support https://github.com/centricular/cerbero .

GstPlugin : GStreamer 1.0 Core Reference Manual https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer/html/GstPlugin.html .

YBlog - Learn Vim Progressively http://yannesposito.com/Scratch/en/blog/Learn-Vim-Progressively/ .

The Basics of Writing a Plugin https://gstreamer.freedesktop.org/documentation/plugin-development/basics/index.html .

Things to check when writing an element https://gstreamer.freedesktop.org/documentation/plugin-development/appendix/checklist-element.html .

Shared Libraries http://tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html .

Install Qt 5 on Ubuntu - Qt Wiki https://wiki.qt.io/Install_Qt_5_on_Ubuntu .

Vim Awesome http://vimawesome.com/ .

Base MPEG-TS descriptors : GStreamer Bad Plugins 1.0 Library Reference Manual https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-bad-libs/html/gst-plugins-bad-libs-Base-MPEG-TS-descriptors.html#GstMpegtsDescriptor-s .

Base MPEG-TS sections : GStreamer Bad Plugins 1.0 Library Reference Manual https://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-bad-libs/html/gst-plugins-bad-libs-Base-MPEG-TS-sections.html .

MPEG-2 TS Overview http://cmm.khu.ac.kr/korean/files/02.mpeg2ts1_es_pes_ps_ts_psi.pdf .

Packetized elementary stream - Wikipedia https://en.wikipedia.org/wiki/Packetized_elementary_stream .

GStreamer-devel - Gstreamer-1.0 blacklisted entries http://gstreamer-devel.966125.n4.nabble.com/Gstreamer-1-0-blacklisted-entries-td4671087.html .

DevDocs — Offline https://devdocs.io/offline .

Example GStreamer Pipelines - Texas Instruments Wiki http://processors.wiki.ti.com/index.php/Example_GStreamer_Pipelines .

GStreamer Debugging - RidgeRun Developer Connection https://developer.ridgerun.com/wiki/index.php/GStreamer_Debugging .

gst-launch-1.0 https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html#environment-variables .

Basic tutorial 11 : Debugging tools https://gstreamer.freedesktop.org/documentation/tutorials/basic/debugging-tools.html#the-debug-log .

Sisselbox | Trello https://trello.com/b/20788DTs/sisselbox .

GStreamer Typefinding and Dynamic Pipelines - Part 2 · pyVision https://pi19404.github.io/pyVision//2014/12/17/21/ .

OpenVPN < Systeme/ConfigPostes < Foswiki https://wiki.smile.fr/view/Systeme/ConfigPostes/OpenVPN#Installation_client_Linux_40Debian_45_Ubuntu_41 .

Good patching practice http://en.tldp.org/HOWTO/Software-Release-Practice-HOWTO/patching.html .

7 Patch Command Examples to Apply Diff Patch Files in Linux http://www.thegeekstuff.com/2014/12/patch-command-examples .