Dupont Jade

P14 - Écran Géant Modulaire

Intitulé du projet :

Le renouvellement plus ou moins périodique de matériel informatique entraîne un nombre grandissant d'écrans d'ordinateur qui sont difficilement réutilisables compte tenu de leur petite taille. (17")
Afin de leur donner une seconde vie, ce projet a pour but de construire une plateforme permettant d'assembler ces écrans en un écran géant d'une forme "quelconque"

Objectifs :

- Ajouter une Raspberry Pi sur chaque écran afin de le rendre "intelligent".
- Créer une application permettant de prendre un flux vidéo et de le diriger vers l'écran visé.
- Permettre de déplacer facilement les écrans tout en maintenant le flux, que ce soit sur des configurations pleines en 2D ou des configurations plus complexes avec des espaces vides ou en 3D.

Matériel :

- 3 Raspberry Pi (une pour chaque écran + une faisant office de serveur)
- 3 Câbles USB alimentation Raspberry
- 1 Commutateur (5 Ports ou +)
- 3 Câbles Ethernet
- 3 Câbles VGA
- 3 Adaptateur VGA-femelle vers HDMI-mâle
- 3 écrans
- 3 Câbles alimentation écrans

Sommaire

I) Analyse du projet :

(Retour Sommaire)

1) Positionnement par rapport à l'existant & scénario d'usage du produit ou du concept envisagé

(Retour Sommaire)

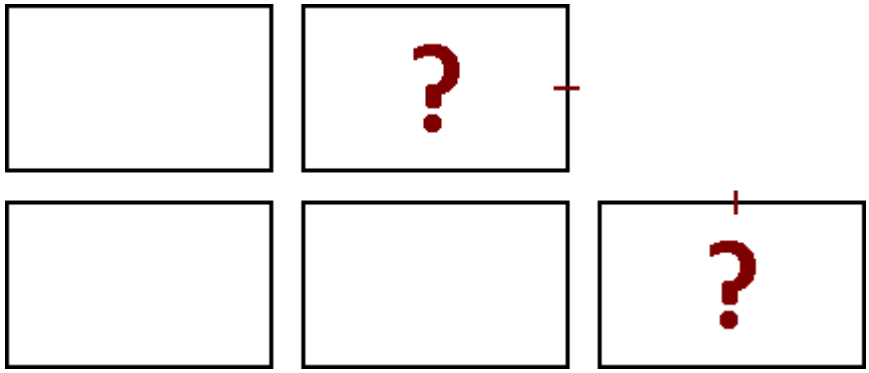
Lorsque l'on évoque un écran géant composé de plusieurs écrans plus petit, on pense généralement à une configuration à plusieurs écrans, gérés par la ou les cartes graphiques pour ce qui est des ordinateurs, ou alors aux écrans géants publicitaires qui sont parfois situés sur des façades.

Pour ce qui est de ce projet, on envisage plus une utilisation personnelle permettant de “recycler” des petits écrans afin d’afficher un flux vidéo pouvant être celui d’une caméra, d’une vidéo YouTube (ou autre) voir d’un fichier vidéo quelconque.

2) Le positionnement des écrans et la communication entre ces derniers

(Retour Sommaire)

Le premier problème qui se manifeste dans la réalisation de ce projet est de déterminer comment vont communiquer les différents écrans pour leur positionnement qui lié à quelle partie du flux devra être affichée.
Une première proposition était d’utiliser des capteurs positionnés sur les écrans et en fonction de leurs états il aurait été possible de déterminer leur position pour ensuite l’inscrire dans un fichier de configuration. Cependant cette idée fut rapidement abandonnée en raison des problèmes qui se révèleraient difficile à prendre en compte.
Le premier problème était les configuration non-régulières (en rectangle, si posée à plat)

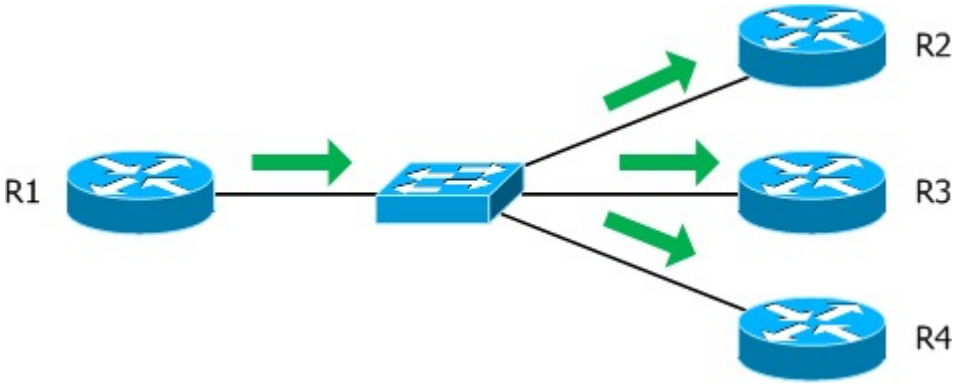


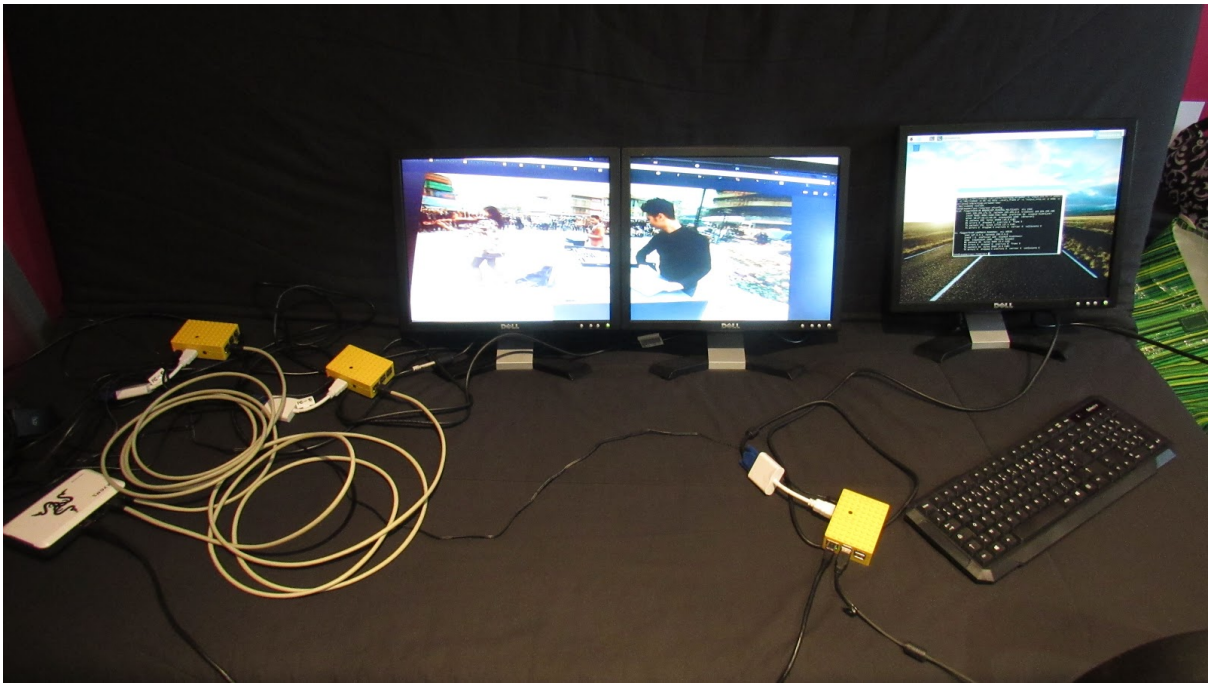
Dans cette configuration, les capteurs situés en haut et à droite des écrans indiqués seraient considéré comme étant en haut à droite d'une configuration, ainsi il deviendrait difficile - sans traitement plus élaboré - de déterminer leur véritable position en ne prenant en compte uniquement les capteurs.
De ce fait, il a été suggéré d'opter pour une solution en utilisant une R-pi équipée d'une caméra pour prendre une photo de l'ensemble des écrans où il leur serait "demandé" d'afficher une image particulière qui serait un identifiant que l'on pourrait analyser via un traitement d'image. Cette solution s'avère déjà moins coûteuse, moins encombrante et réalisable bien qu'elle nécessite d'être mise en oeuvre dans un environnement particulier pour faciliter le traitement d'image.

3) La gestion du flux vidéo côté "serveur"

(Retour Sommaire)

Le deuxième problème dans ce projet est la gestion du flux vidéo. Par exemple, du côté de la lecture du flux, il ne faut pas qu'il y ait une désynchronisation entre les R-pi.
On ne peut donc pas envoyer un fichier vidéo à chaque R-pi de lecture puisqu'il risquerait d'y avoir des décalages, pour ce faire il convient d'utiliser une R-pi pour héberger le stream et de faire en sorte que les autres R-pi s'y connectent pour y récupérer le flux.
Deux solutions ont donc été envisagées, soit utiliser un lecteur capable de diffuser un flux (VLC) soit mettre en place un serveur de streaming (mjpg_streamer). Suivant les besoins, les deux solutions sont valables.





II) Déroulement du projet :

(Retour Sommaire)

1) Mise en place d'un serveur de streaming.

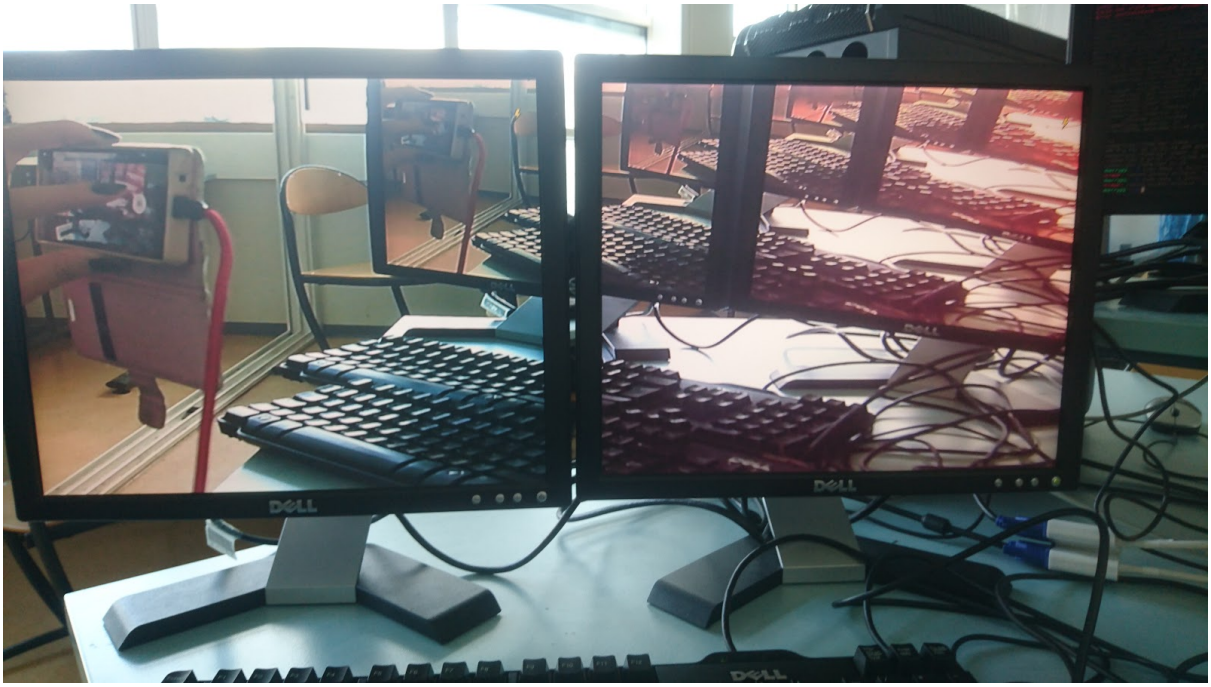
(Retour Sommaire)

Dès le départ, j'ai voulu mettre en place la Raspberry maîtresse permettant de faire office de serveur de streaming. Comme j'ai pu l'expliquer précédemment, j'ai pu tester deux méthodes, la première avec **mjpg-streamer**, l'autre avec **VLC** suite aux difficultés rencontrées pour utiliser **v4l2loopback** avec **ffmpg** pour charger une vidéo sur un dummy-point.

1.1) mjpg-streamer.

(Retour Sommaire)

mjpg-streamer est un utilitaire intéressant permettant de mettre en place facilement un serveur de stream qui utilise une caméra (par défaut **/dev/video0**, si elle est présente) et dispose d'un serveur web intégré simple à utiliser.
Dépot GIT mjpg-streamer : lien
Instructions d'installation : lien
L'installation et la mise en place n'a rien de difficile et à pu se faire rapidement, temps de téléchargement exclu.



1.2) Utilisation de v4l2loopback.

(Retour Sommaire)

Étant donné que mjpg-streamer était parfaitement fonctionnel, j'ai voulu trouver un moyen d'émuler quelque chose d'équivalent au point de montage /dev/video0 qui est la source du serveur de stream pour ensuite y charger un fichier vidéo grâce à **ffmpeg**.
Dépot GIT v4l2loopback : (lien)
L'idée principale était de pouvoir faire :

- Émulation de /dev/video10 avec v4l2loopback
- Chargement avec ffmpeg de la video sur /dev/video10
- Récupération par mjpg-streamer de /dev/video10 pour envoie sur le serveur de stream
- Lecture du stream par les r-pi de lecture

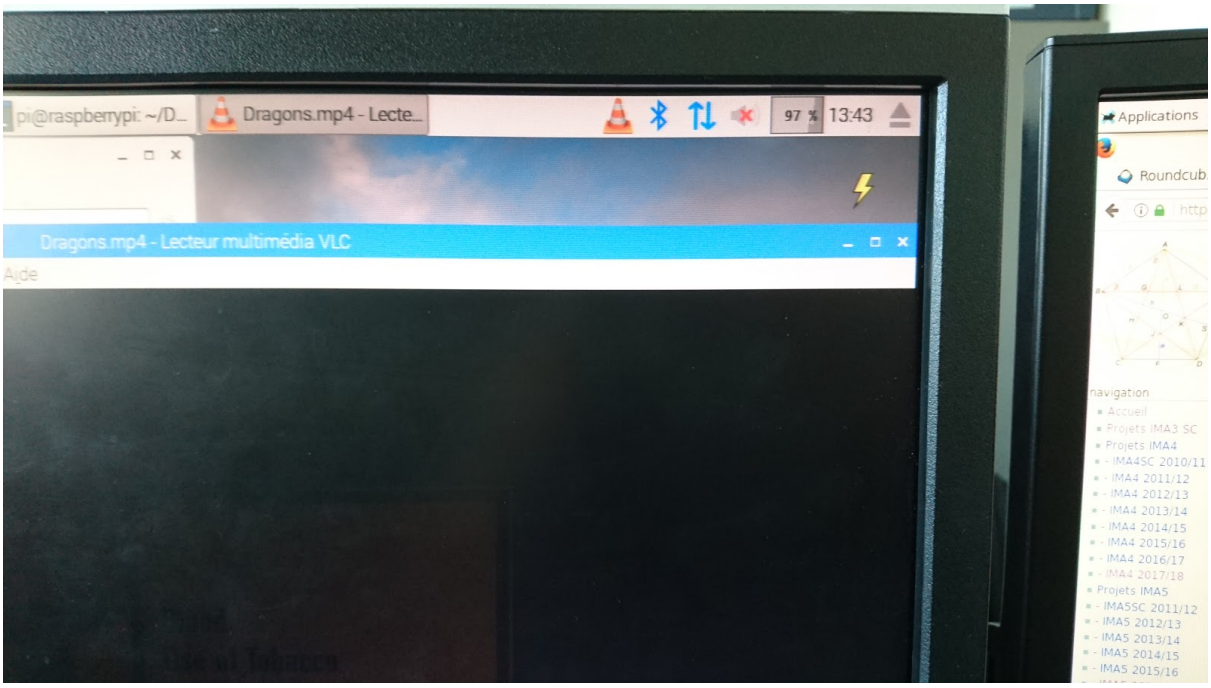
Le problème que j'ai rencontré avec v4l2loopback est situé au niveau des headers du kernel, les versions étaient parfois incohérentes et pour peu qu'on arrive à le faire fonctionner, à la moindre upgrade de la R-pi pour l'installation d'autre modules, le problème refaisait surface.

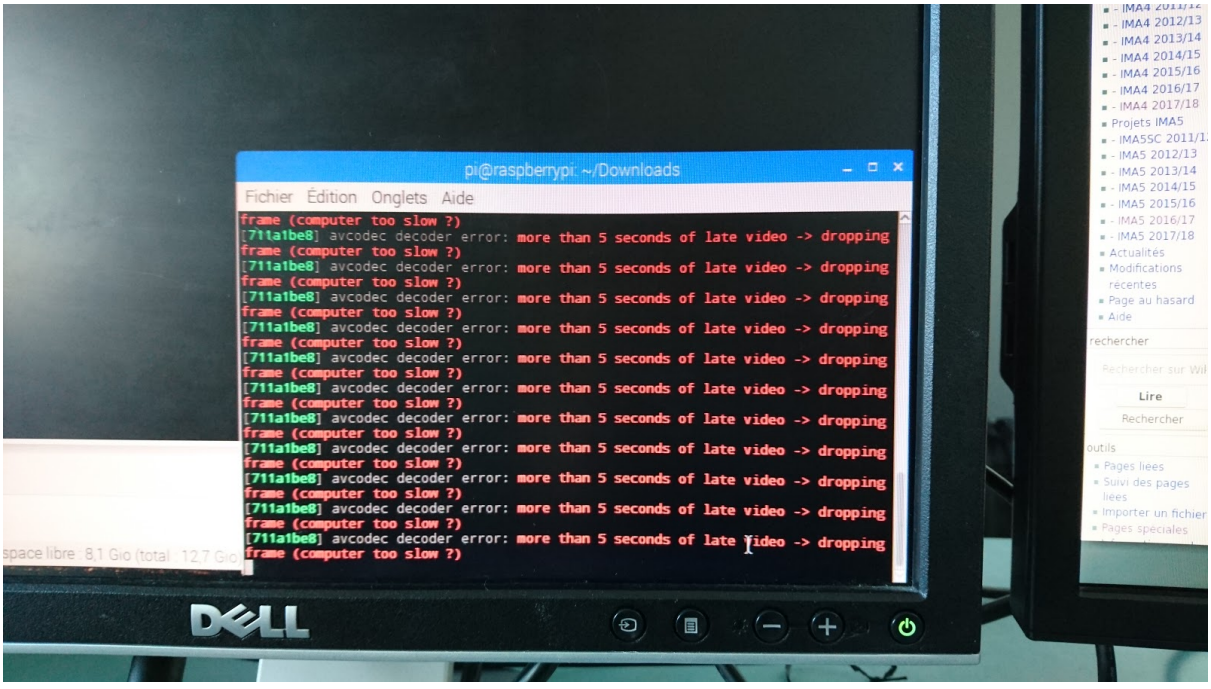
J'ai donc été contrainte d'abandonner à cette partie faute de pour résoudre le problème.

1.3) VLC.

(Retour Sommaire)

Cette méthode est une alternative qui m'a été proposée due à la difficulté que j'ai eu à mettre en oeuvre v4l2loopback. Qui a fonctionné temporairement grâce à l'aide apportée par un des encadrants mais qui a cessé de fonctionner suite à une mise à jour (sudo apt-get upgrade) nécessaire pour OpenCV.
Cependant, en mode interface, la consommation est telle qu'il est impossible de visualiser un flux stable et sans perte.





Il en va quasiment de même pour ce qui est des commandes en mode sans interface, même sans attendre les 90+ de consommation, le flux reste instable et avec un nombre considérable de pertes.
Côté diffusion : (source)

Transcode the input stream and send it to a multicast IP address with the associated SAP announce:

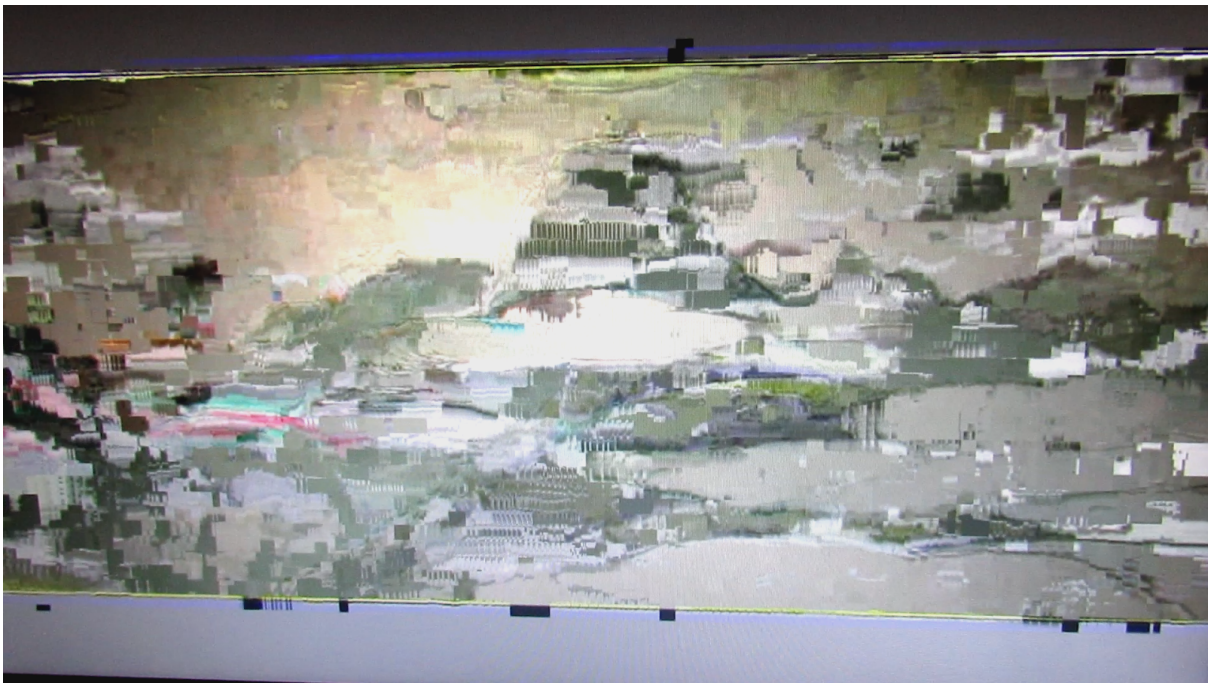
```
% vlc -vvv input_stream --sout
'#transcode{vcodec=mp4v,acodec=mpga,vb=800,ab=128,deinterlace}:
rtp{mux=ts,dst=239.255.12.42,sdp=sap,name="TestStream"}'
```

Côté lecture : (source)

To receive the input stream that is being multicasted above on a client:

```
vlc rtp://239.255.12.42
```

Je n'ai pas été en mesure de résoudre le problème des pertes et coupures.



2) Configuration et utilisation d'OpenCV + Python pour la détection des écrans.

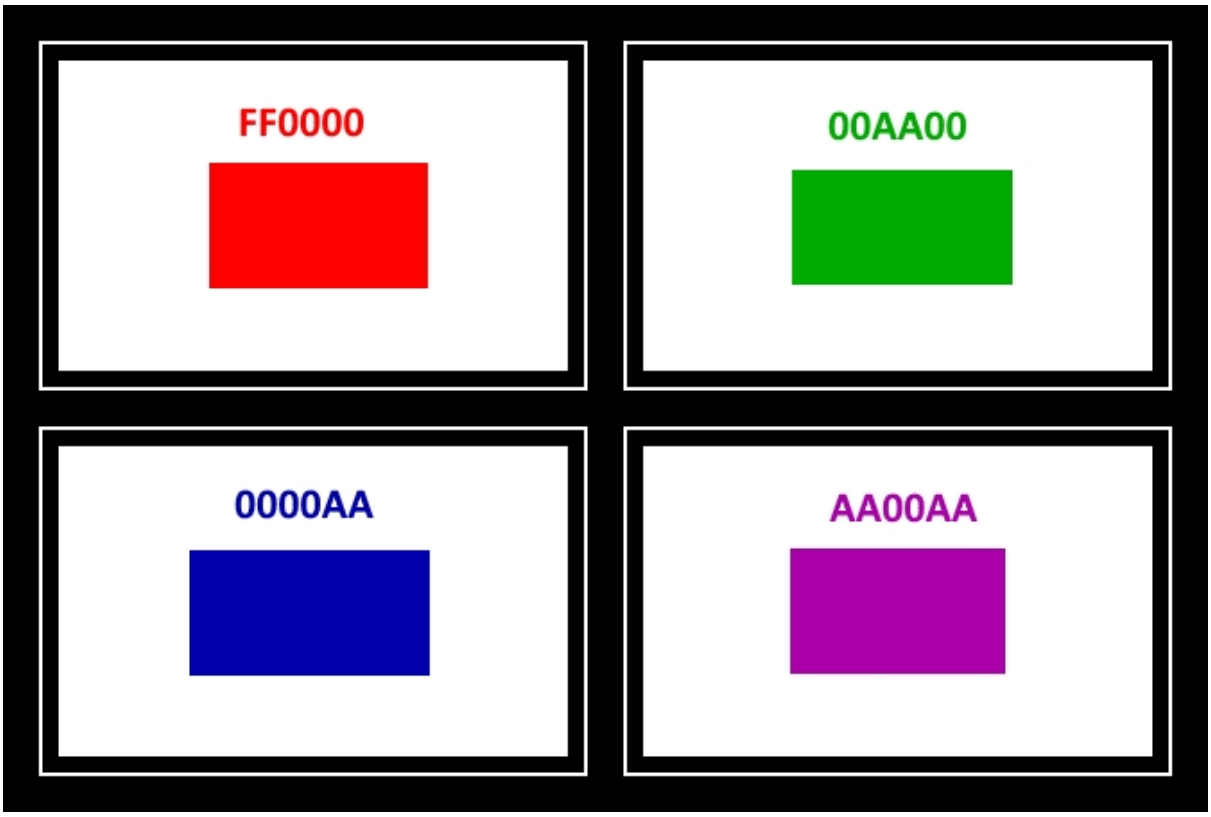
(Retour Sommaire)

Jusqu'à présent j'avais utilisé un fichier de configuration que j'avais moi même implémenté dans chaque R-pi pour chaque écran.
Toutefois, l'un des objectifs de ce projet est de permettre une configuration variable sans que l'utilisateur ait à fournir les paramètres.

La méthode que je souhaitais utiliser pour reconnaître les écrans consiste à prendre une photo avec la R-pi maîtresse (donc nécessite qu'elle soit placée à un endroit particulier en face des écrans, 1ère contrainte) pour ensuite traiter l'image mais en fonction de l'éclairage et de l'environnement, il peut y avoir des problèmes dans les étapes du traitement. (inspiré de **cet article**)
- 1ère étape : le seuillage pour pouvoir binariser une copie de l'image pour la détection des bords
- 2ème étape : une fois la copie binarisé, on peut travailler dessus pour récupérer les coordonnées des écrans sur les images

- 3ème étape : on effectue le traitement sur l'image d'origine mais avec les coordonnées relevées sur la copie.

Le cas théorique se présente de la manière suivante :



chaque écran affiche son "**image-ID**" qui est connu par la R-pi maîtresse.
Conditions requises :
- un fond sombre/noir + une pièce correctement éclairée avec aucun autre objet pouvant interférer dans la détection que les écrans
- suffisamment d'espace pour installer chaque écrans mais aussi assez de recul pour pouvoir prendre en photo la totalité du système

Points négatifs de cette solution : les couleurs sur la photo ne correspondront quasiment jamais aux codes couleurs indiqués et il faudra faire des approximations ou des intervalles de valeurs lors de la détection afin de correctement identifier un code couleur suivant la luminosité.

3) Configuration des R-pi de lecture.

(Retour Sommaire)

Pour la partie lecture j'ai, suite aux premières recherches, testé deux lecteurs, VLC et Omxplayer.
VLC est certes un lecteur connu du grand public et facile à utiliser grâce à son interface, il reste néanmoins non-adapté pour la situation. En effet, lors des test (sans serveur de stream mais avec le fichier vidéo présent localement) on observe tout de même des pics de consommation plutôt élevés.

En revanche, ce n'est pas le cas pour Omxplayer qui, bien que devant être lancé en ligne de commande, est extrêmement fluide et reste simple à utiliser.
Également, quel que soit la manière dont le fichier de configuration "position.conf" soit implémenté (manuellement ou alors automatisé avec les scripts python s'il étaient fonctionnels), le script suivant arrive à gérer l'affichage correctement.

```
#!/bin/sh

X=`grep -m 1 X= position.conf | cut -d'=' -f2`
Y=`grep -m 1 Y= position.conf | cut -d'=' -f2`

NB_X=`grep -m 1 NB_X= position.conf | cut -d'=' -f2`
NB_Y=`grep -m 1 NB_Y= position.conf | cut -d'=' -f2`

X1_tmp=1280
X1=`echo $X1_tmp"*"$X"/"$NB_X | bc`
Y1_tmp=720
Y1=`echo $Y1_tmp"*"$Y"/"$NB_Y | bc`

X2_tmp=1280
X2=`echo $X2_tmp"*(1+"$X")/"$NB_X | bc`
Y2_tmp=720
Y2=`echo $Y2_tmp"*(1+"$Y")/"$NB_Y | bc`

STREAM="http://169.254.171.72:8081/?action=stream"
echo "Fichier de lecture = "$STREAM
omxplayer --aspect-mode stretch --crop $X1,$Y1,$X2,$Y2 --live $STREAM
```

```
$sudo ./play_stream.sh
```

On récupère la position de l'écran (X et Y) et le nombre d'écrans (NB_X, NB_Y) pour ensuite sélectionner la partie de la vidéo à afficher.

NB: l'avant dernière ligne du script (echo) est normalement commentée.
De plus il serait possible d'ajouter dans position.conf les valeurs X1_tmp, Y1_tmp, X2_tmp, X2_tmp qui sont les dimensions entrées au lancement du serveur de stream/du fichier vidéo
De même qu'il serait également plus intéressant d'y faire figurer l'adresse IP du stream qui est celle de la R-pi maîtresse.

III) Bilan & Suggestions :

(Retour Sommaire)

1) [✔] Gestion d'un flux vidéo d'une caméra USB.

(Retour Sommaire)

L'utilisation de **mjpg-streamer** permet de mettre en place un processus de streaming. Il prend en entrée un flux vidéo sur **/dev/video0** (*-i input_uvc.so*)

```
input_uvc.so- UVC webcam grabber

The following parameters can be passed to this plugin:

-d | --device <DEVICE>
       Video device to open. Example: /dev/video0.
-r | --resolution <RES>
       The resolution of the video device. Can be one of the following strings:QSIF QCIF CGA QVGA CIF VGA
       SVGA XGA SXGA or a custom value like thefollowing example: 640x480.
-f | --fps <NUMBER>
       Frames per second. You must choose a value supported by your camera.
-y | --yuv
       Enable YUYV format and disable MJPEG mode.
-q | --quality <PERCENT>
       JPEG compression quality in percent (activates YUYV format, disables MJPEG).
-m | --minimum_size <SIZE>
       Drop frames smaller then this limit, useful if the webcam producessmall-sized garbage frames. This
       may happen under low light conditions.
-n | --no_dynctrl
       Do not initalize dynctrls of Linux-UVC driver.
-l | --led <on|off|blink|auto>
       Switch the LED "on", "off", let it "blink" or leave it up to the driverusing the value "auto".
```

et retransmet sur le serveur HTTP intégré (*-o output_http.so*) sur le port 8080.

```
output_http.so- HTTP output plugin

-w | --www <DIRECTORY>
       Directory that contains webpages in flat hierarchy (no subdirectories). ForopenSUSE packages this is
       /usr/share/mjpg-streamer/www.
-p | --port <PORT>
       TCP port for this HTTP server. The default port is 8080.
-c | --credentials <USERNAME:PASSWORD>
       Ask for "username:password" on connect.
-n | --nocommands
       Disable execution of commands.
```

(source)

2) [✕] Gestion d'un flux vidéo à base d'une vidéo YouTube.

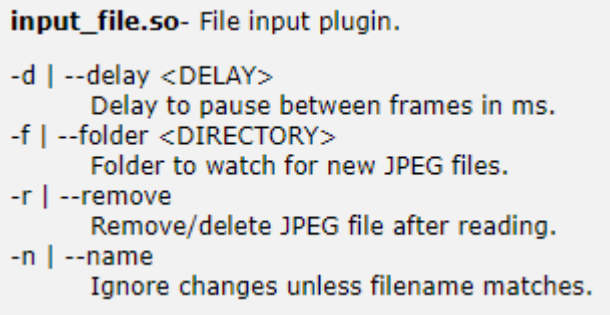
(Retour Sommaire)

Il existe bien un plugin intégré à mjpg-streamer du nom de **"input_http"** où l'on pourrait imaginer qu'il s'agit d'un moyen permettant de fournir le lien d'une vidéo qui serait ensuite redirigée sur le serveur HTTP mais il s'agit d'un plugin pour rediriger un autre serveur de stream mjpg-streamer. (source)

3) [✕] Gestion d'un flux vidéo à base d'un fichier.

(Retour Sommaire)

Il y a également un autre plugin pour mjpg-streamer, **"input_file"**, mais il fonctionne avec des images JPEG (source).



input_file.so- File input plugin.

-d | --delay <DELAY>
Delay to pause between frames in ms.
-f | --folder <DIRECTORY>
Folder to watch for new JPEG files.
-r | --remove
Remove/delete JPEG file after reading.
-n | --name
Ignore changes unless filename matches.

Afin de permettre au système de gérer les fichiers vidéo, il m'a été conseillé d'utiliser VLC pour diffuser. Mais après avoir effectué les tests, la R-pi maîtresse était incapable de suivre la cadence.

L'une des solutions que j'ai à proposer serait d'utiliser un PC avec VLC pour héberger le stream et de connecter les R-pi de lecture au réseau local où se trouve le PC pour ensuite les utiliser pour y lire le stream. (à condition bien sûr qu'Omxplayer puisse lire un stream de VLC - expérience non testée avec les R-pi en lecture mais fonctionnelle avec deux PC avec VLC. Sinon il est envisageable d'utiliser VLC en lecture sur les R-pi avec cette proposition de solution.)

4) [✔] Fonctionnement des systèmes de lecture avec un fichier de configuration donné.

(Retour Sommaire)

En entrant manuellement les coordonnées de chaque écran, le script de lecture est tout à fait capable de gérer l'affichage (dans le cas où le flux vient de mjpg-streamer). Impossible de tester la réception dans de bonnes conditions lorsque VLC est utilisé pour la diffusion en raison de la non-fluidité déjà présente côté serveur.

5) [✕] Fonctionnement des systèmes de lecture avec un fichier de configuration géré de manière autonome.

(Retour Sommaire)

Le test sur système réel n'a pas pu être effectué, seul un modèle théorique a été pensé.
Le système actuel est tel que les fichiers de configuration doivent être implémentés à la main et donc que les écrans doivent être ensuite placé correctement..

Le principal problème rencontré lors des premiers test du traitement d'image réside dans le choix des couleurs et le fond sur lequel la photo est prise.

Mais même en ayant réussi à mettre en place un traitement d'image efficace, il aurait fallu synchroniser chaque étapes.
→ la R-pi maîtresse doit récupérer l'ip de chaque R-pi de lecture dès qu'elle (une R-pi de lecture) est connecté au système (commutateur) et inversement pour ensuite pouvoir récupérer le fichier de configuration envoyé par la R-pi maîtresse et se connecter au stream.
→ Une boucle sur la R-pi maîtresse qui broadcast un message et une boucle d'écoute sur chaque R-pi de lecture qui tourne tant qu'elle ne possèdent pas leur IP respective. Ainsi, lorsqu'une R-pi de lecture est connectée, elle reçoit un message broadcast qui entraîne le lancement d'un script de réponse permettant d'envoyer son IP et le code couleur à reconnaître sur l'image-ID pour ensuite recevoir un fichier de configuration lui permettant de lire correctement le stream.

Ceci dit, le non-fonctionnement de cette partie est une bien maigre perte, la première justification vient du fait que ce projet, même s'il est utilisé à des fin domestiques/personnelles, est tout de même adressé à des utilisateurs au minimum novice vis-à-vis de ce qu'est une Raspberry-Pi.
Deuxièmement, il ne serait pas exagéré de dire que changer soi même le fichier de configuration (4 lignes tout à fait explicites) à la main pour ensuite faire un montage qui normalement ne devrait pas évoluer trop souvent, en terme de nombre d'écrans et de positionnement de ces derniers est un bien maigre effort.

Conclusion :

(Retour Sommaire)
Les choix effectués au début du projet se sont montrés trop ambitieux et une mauvaise gestion du temps, recherches de solutions aux problèmes qui se sont présentés (headers kernel, dysfonctionnements, méthodes - imprévus exclus i.e une R-pi dont la SD a flanchée en milieu de projet et qui a nécessité une réinstallation) n'ont pas été favorable.
De plus, un cruel manque de connaissance sur la mise en pratique des modules et utilitaires employés s'est avéré être un handicap majeur, malgré les conseils et aides apportés, beaucoup de choses ont été envisagées mais très/trop peu ont pu être mises en place.

Afin de finaliser ce projet, il faudrait réussir à re-faire fonctionner v4l2loopback pour émuler une entrée, puis ffmpeg pour y lire le fichier vidéo et ensuite utiliser ce même point de montage pour mjpeg-streamer afin de retransmettre le fichier sur le serveur de streaming. mais il resterait aussi l'automatisation des fichiers de configuration à implémenter qui reste quelque chose de complexe pour une fonctionnalité qui présente un nombre non-conséquent de contraintes, il serait alors plus simple de faire les fichier de configuration soi-même ou alors via un petit programme qui demande à l'utilisateur d'entrer les coordonnées.