Rapport Smart System

Projet Station TV

DELALANDRE Mathieu

ISIE4
Matteo DOYEN
Arno BALOIS

2022-2023

I-Introduction

Remerciements

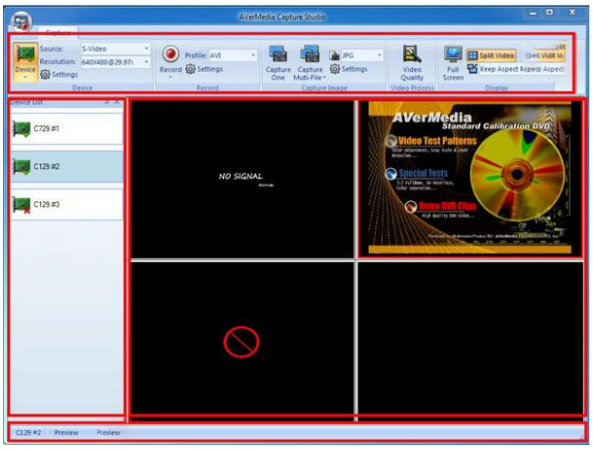
Nous tenons à remercier Mathieu DELALANDRE pour son encadrement tout au long de ce projet et pour sa pédagogie. Nous voulons remercier Van-Hao pour nous avoir renseignés et aidés à chacune de nos interventions sur la station TV et le PRD Corentin LEZE-ROBERT pour avoir répondu à nos questions rapidement.

Lexiques

Vues

Une vue correspond au retour vidéo d'un port HDMI d'une carte de capture de la Station TV.

La vue tient son nom du retour visuel du flux vidéo capturé et affiché sur le logiciel "AverMedia", ici une vue correspond à l'un des carrés de la grille.



Tuner

Un Tuner est un décodeur de télévision.

Phidget

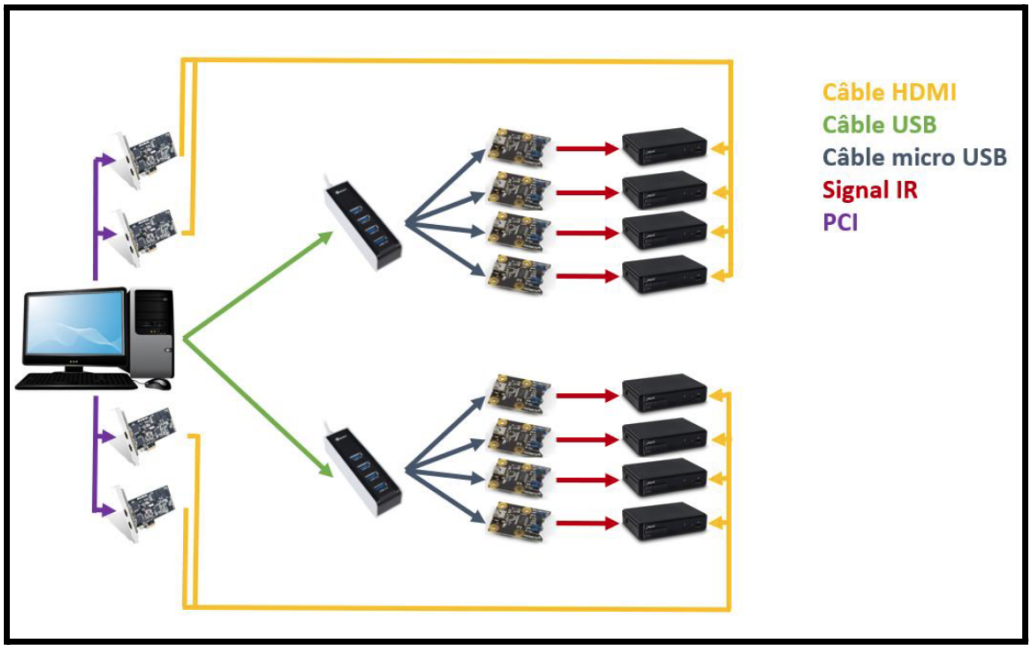
Un phidget est un émetteur infrarouge utilisé pour piloter les Tuners, il est utilisé comme une télécommande.

Contexte du projet

Ce projet est en lien avec la station TV développée depuis 2018 par l'école Polytech Tours et le laboratoire LIFAT. La station TV est une plateforme de calcul parallèle pour le traitement automatique des chaînes de télévision de la TNT. La station est équipée deux machines : DELL PowerEdge T640 pour le traitement temps-réel de flux vidéo et DELL T7610 pour la capture vidéo.
L'objectif de la station est le déploiement d'applications de traitement d'images, de vidéos et d'intelligence artificielle du laboratoire LIFAT. En dehors du développement de la station elle-même, le projet station TV recouvre aujourd'hui de multiples services pour la capture smart de vidéos et d'images jpg, la détection de publicités et de génériques TV, le

scraping Web de données TV, le fact-checking et l'annotation automatique texte/vidéo.

Ce projet est en lien avec la capture vidéo dans la station TV. La capture vidéo est supportée matériellement par des cartes Avermedia CL332-HN embarquées dans la machine DELL T7610 et reliées à un multituner. Le multituner est constitué à partir de tuners individuels de marque Astrell.



L'architecture cartes CL332-HN / multituner permet la capture de 8 flux vidéo en simultané dans la station. Les tuners Astrell au sein du multituner sont pilotés via des émetteurs IR 1055 PhidgetIR connectés par un hub USB. Ces tuners utilisent le protocole standard NEC avec le code système 0x08F6. Cette configuration matérielle (tuners Astrell / émetteurs IR 1055 PhidgetIR) permet ainsi à la station TV de "zapper" entre les 30 chaînes de la TNT.

Une solution logicielle de pilotage du multituner a été établie au cours de projets antérieurs. Cette solution permet à date une configuration par l'utilisateur à l'initialisation de la station TV pour une capture statique et en continu des chaînes. Elle supporte le contrôle des 8 tuners via un programme en ligne de commande et sous la forme d'une d'interface. Elle permet également la (re)configuration des tuners "calibrage / balayage des chaînes".

À terme, l'objectif sur le projet station TV sera de migrer cette solution vers une capture dynamique via le déploiement d'un algorithme d'IA. Cela permettra sur la station une capture ciblée par contenu, par exemple l'ensemble des émissions politiques, de divertissement, les films et séries sur une période donnée. Un préalable à la mise en place d'une telle solution et l'analyse de performance du contrôle des tuners, et en particulier celle du temps de latence (durée d'exécution de la commande d'un émetteur Phidget + traitement du tuner + remontée HDMI du flux vidéo + encodage des cartes Avermedia

CL332-HN). Il est en effet nécessaire de bien caractériser cette latence afin de la prendre en compte dans l'algorithme d'IA.

Problématique

Dans ce contexte, il nous est demandé de reprendre la solution logicielle de pilotage du multituner en version IHM et de la faire évoluer en ajoutant de nouvelles fonctionnalités.

Il est également demandé d'évaluer la latence, cette partie du projet est surnommée "Smart-Latence", afin de pouvoir permettre au enregistrement de prendre en compte celle-ci dans le changement de la chaîne.
Enfin, il nous faudra travailler sur le projet du PRD.

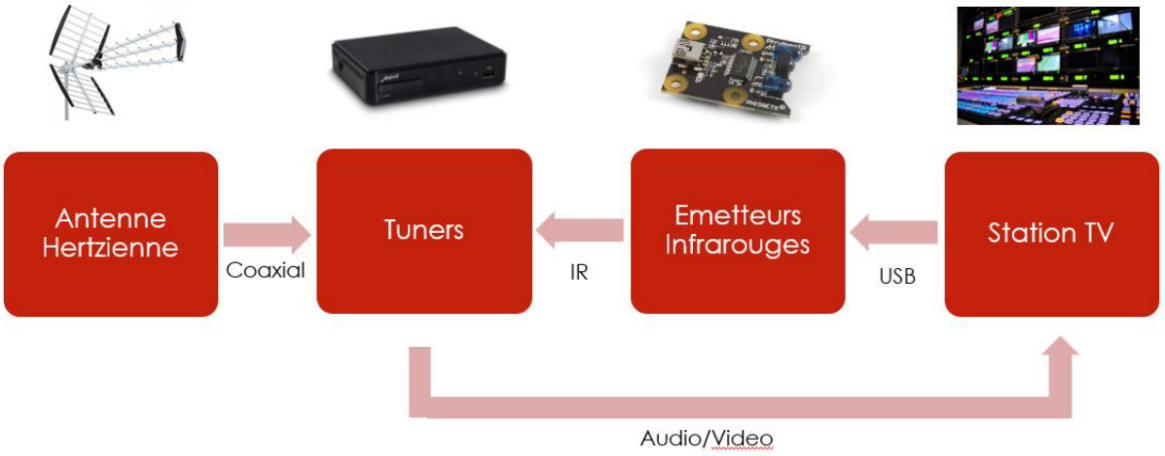
II- Existant

Architecture

La station de Polytech dispose de différents matériels pour son fonctionnement et lui permet notamment de traiter les images, capturer des morceaux de programmes télévisés ou encore d'exploiter plusieurs chaînes simultanément.
Pour cela, la station comprend les dispositifs suivants :

- Une Antenne TV Hertzienne (présente sur le toit du bâtiment)
- Un Rack station comprenant 8 Tuners TNT
- Quatre cartes d'acquisition et d'enregistrement avec deux entrées
- Deux hubs, présents dans le rack, alimentés en USB
- Huit émetteurs infrarouges Phidgets

Afin de décrire le processus de fonctionnement de ce système, un schéma représentatif de la situation actuelle au sein de la station est disponible ci-dessous.



Prise en main

Avant d'ajouter des fonctionnalités à la télécommande, nous avons tout d'abord pris le temps de manipuler un phidget (émetteur infrarouge) avec un tuner (décodeur TV) de notre côté. Pour cela, nous avons suivi le guide d'installation fourni par les anciens groupes projet. Nous avons ensuite pris en main la solution de pilotage des phidgets en ligne de commande développée les années précédentes et l'avons recompilé.

III- Expression du besoin

Cahier des charges

Comme il est évoqué plus haut dans le document, la demande du client peut être divisée en plusieurs parties, chacune séparée les unes des autres.

1) **Reprise de l'IHM**

La première partie est celle consistant à la reprise du logiciel de pilotage du multituner sous forme d'IHM et d'y ajouter des routines de contrôle (Mosaïque, Chenillard, Damier...). Mais aussi des commandes demandées par d'autres utilisateurs qui permettent de faciliter l'accès à certaines fonctionnalités du tuner (reparamétrage du tuner et recherche de chaîne).

2) **Latence Max**

La deuxième partie est l'évaluation de la latence maximale entre l'appui d'un utilisateur sur un bouton de l'IHM télécommande et le moment où le tuner reçoit la commande. Cette latence est importante à connaître, car elle permettra notamment, en étant intégré dans les temps de début d'enregistrement d'une chaîne, de ne pas avoir trop ou pas assez enregistré une émission.

3) **Normalisation de la station TV**

La troisième partie est la normalisation des tuners de la station TV. Par erreur, les années précédentes, un modèle différent de tuner a été commandé et intégré à la station TV, bien qu'il soit pilotable à travers les mêmes phidgets et que la capture de son flux vidéo fonctionne, il possède un menu différent et des temps de réaction différent aux commandes envoyé par le phidget. Afin de pouvoir travailler sur des solutions communes à tous les tuners, il faudra le remplacer par un tuner du même modèle que celui visé.

4) Zapping automatique

Enfin, la quatrième partie est celle consistant à développer un outil prenant en paramètre le fichier de sortie du projet du PRD nommé "Optimisation pour la sélection de programmes TV". Ce fichier est en format CSV et contient une liste de chaîne suivie d'une date début et d'une date de fin. Ce fichier correspond à une liste utilisant de façon optimale les 8 tuners de la station TV, chaque ligne correspond à un programme télévisé devant être enregistré et les deux dates au début et à la fin du programme. La lecture de ce fichier doit permettre le changement de chaîne automatique ainsi que l'attribution du tuner lié au programme à enregistrer.

Les trois premières parties sont les plus critiques et doivent donc être réalisées en priorité.

IV- Réalisation

IHM Télécommande

Le tableau ci-dessous contient la liste des fonctionnalités contenues dans le cahier des charges et leur état d'avancement.

Fonctionnalités	Avancement	Fonctionnement
Contrôle individuel et groupé des tuners.	**Terminée**	Les vues sélectionnées, en cochant une checkbox qui leur est liée, sont ajoutées à une liste. Lors du clic sur un bouton, tous les phidgets concernés enverrons les commandes liées au bouton.
Réglage du numéro de chaîne / volume.	**Terminée**	Envoie d'une commande à un phidget en lançant l'utilitaire exécutable à chaque appuie sur le bouton
Gestion / aide à la configuration des affectations vue / bouton de contrôle.	**Terminée**	Ajout d'un fichier de configuration contenant le numéro de la vue et le numéro du phidget qui lui est lié
Reconfiguration du paramétrage / configuration d'usine.	**Terminée**	Des touches sont envoyées à la suite afin de permettre de se balader dans les sous-menus et de simuler l'appuient sur la fonction.
Recalibrage / balayage des tuners.	**Terminée**	Des touches sont envoyées à la suite dans le but de permettre de se balader dans les sous-menus et de simuler l'appuient sur la fonction.
Gestion des sous-titres.	**Terminée**	Ajout d'une nouvelle commande dans l'exécutable, s'il reçoit la lettre "t" il envoie la commande "sous-titre".
Routine de contrôle "chenillard , mosaïque ".	**Terminée**	*Voir la partie sur le rapport*
Routine de contrôle "chenillard".	**Terminée**	*Voir la partie sur le rapport*

Commentaires :

- Le volume ne peut être qu'augmenté ou diminué , il n'est pas possible de saisir une valeur précise de volume à atteindre.
- Pour le bon fonctionnement de la reconfiguration et du recalibrage, le bouton "Menu" a dû être supprimé, car l'état du menu n'est pas remis à zéros lorsqu'on quitte le menu, ce qui posait un problème puisque les deux fonctionnalités sont des suites de touche.

- Le bouton sous-titre permet d'ouvrir le menu de sous-titre , pour activer les sous-titres l'utilisateur doit sélectionner à l'aide des flèches de direction une langue souhaitée.

Routines de contrôle :

Chenillard :

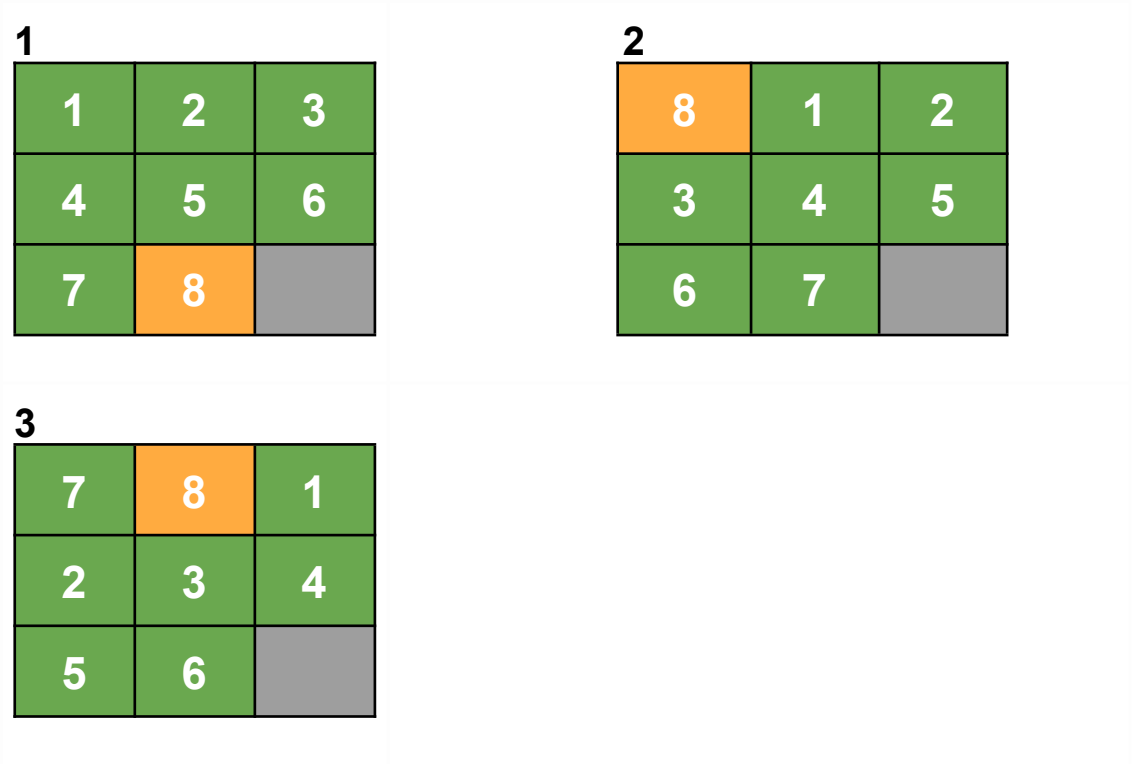
Pour le chenillard, nous avons décidé de changer progressivement les chaînes une par une de 1 à 8.

Au démarrage :

1) Nous commençons par attribuer à chaque vue la chaîne correspondant au numéro de sa vue (exemple : vue 3 = chaine 3).

En boucle :

2) Toutes les 6 secondes, chaque vue augmente son numéro de chaine. Si un tuner atteint la chaine 8 , alors au prochain tours de boucle celui-ci retourne à la chaîne 1.



Légendes :

Vues non sélectionnées

"Tête" du chenillard

Mosaïque :

Pour la mosaïque, nous avons décidé de changer aléatoirement deux vues à un intervalle de temps régulier.

Au démarrage :

1) Pour cela, nous commençons par attribuer une chaîne aléatoire aux 8 vues, cette chaîne est comprise entre la 1 et la 18 (ces chaînes sont toutes connues et se suivent sur la TNT).

En boucle :

2) Toutes les 6 secondes, deux vues sont aléatoirement choisies parmi celles qui n'ont pas été choisies au changement de chaîne précédent.

3) Deux nouvelles chaînes sont choisies aléatoirement entre la 1 et la 18 et sont envoyées aux deux tuners liés aux vues choisies.

1

4	6	13
2	10	1
9	18	

2

4	6	13
2	10	1
9	18	

3

4	6	1
12	7	1
6	18	

Légendes :

Vues non sélectionnées

Chaînes sélectionnées aléatoirement

Vues sélectionnées aléatoirement

Smart Latence

Afin d'estimer la latence maximale, plusieurs solutions était possible. La plus précise aurait été d'utiliser la reconnaissance d'image dans le but de mesurer le temps entre l'appui d'un bouton et l'apparition d'un élément reconnaissable (logo d'une chaîne, élément d'interface du tuner rappelant la chaîne sélectionné, écran noir entre les changements de chaîne).

Cette solution a été rapidement écartée par Monsieur Delalandre par manque de temps restant. Nous avions cependant développé un début d'outil capable de reconnaître l'interface de changement de chaîne du tuner et sa disparition.

Nous avons donc opté, après discussion avec Monsieur Delalandre, pour un outil de mesure empirique. Notre outil change les chaînes des 8 tuners en même temps à un intervalle de temps de données N. Ces changements de chaînes sont des augmentations et des diminutions du numéro de chaîne compris entre la chaîne numéro 1 et la 18. En laissant tourner notre solution pendant plusieurs heures, si la latence maximale était trop faible alors, nous devions observer des incohérences dans le numéro des chaînes affichées.

Le numéro de chaînes prévu étant stocké dans le programme, à la fin d'exécution de notre outil, il nous suffit de comparer les chaînes affichées par les cartes de capture et celles prévues. L'objectif ici est de répéter ces mesures en baissant notre latence maximale jusqu'à obtenir la latence maximale la plus proche de celle réelle.

Pour utiliser l'outil, il faut le lancer en ligne de commande en lui passant en paramètre la latence à tester et le nombre d'itérations voulues.

D'après notre outil, nous avons pu extraire ces résultats

Latence (en seconde)	Résultat	Nombre de mesure
1	Instable	*50*
2	Instable	*258*
2,5	Instable	*3558*
2,75	Instable	*10800*
2,9	Instable	*10800*
3	Stable	*10800*
4	Stable	*10800*

Après avoir réalisé les tests précédents, nous pouvons en conclure que la latence max est autour de 3s.

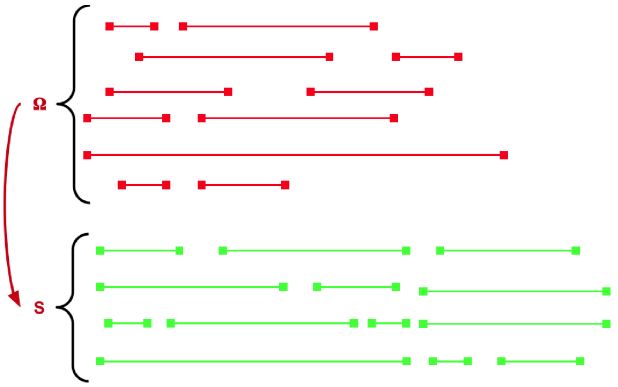
Intégration solution PRD

Le but de cette intégration est de récupérer le fichier solution généré par l'outil développé par le PRD , puis de piloter les différents tuners en fonction de la chaîne à enregistrer et du temps d'enregistrement.

Le Projet PRD consiste à utiliser l'algorithme GREEDY Alpha qui est un algorithme de sélection d'intervalles pondérés afin d'optimiser le nombre de cartes de capture utilisées pour une liste donnée de programmes télévisés à enregistrer.

Chaînes et programmes à enregistrer

Cartes de capture à utilisée



L'outil réalisé par le PRD a comme résultat un fichier CSV nommé Source qui contient 3 informations qui sont nécessaires pour le changement de chaîne. Les Informations sont les suivantes :

- Identifiant de chaîne : code qui permet d'identifier quelle chaîne doit être affiché
- Date Début : date à laquelle l'enregistrement doit commencer
- Date Fin : date à laquelle l'enregistrement se termine

Malheureusement, le fichier source précédemment présenté n'était pas prêt dans les temps, par conséquent, nous n'avons pas eu le temps de réaliser l'intégration, mais nous avons pu réfléchir à une solution possible à développer.

Solution envisagée :

Après avoir analysé le fichier solution fourni par le PRD, nous avons conclu que notre solution allait être constituée de 8 Threads qui seront créés pour contrôler les phidgets individuellement. Pour cela, 4 parties sont à développer :

- La partie que vont utiliser les threads, soit la fonction ThreadChangementChaine dont le pseudo code est le suivant :

Function ThreadChangementChaine (FichierSource *:File, mutex_file * : mutex)

```
    Variable
        chaîne : String
        debut : Entier
        fin : Entier
        WaitingTime : Entier
        new_line : string
    Début
        Tant que Vrai
            lock(mutex_file);
            fgets(new_line , 20 , FichierSource);
            unlock(mutex_file)

            parseString(new_line,chaine , debut , fin )
            WaitingTime < fin - debut

            envoieCommande(Chaîne);
            wait(waitingtime)

        Fin tant que
    Fin
```

Cette fonction prend en paramètre un pointeur vers le fichier CSV “Source” fourni par le PRD et un pointeur vers le mutex qui permet de protéger l'accès à la variable partagé qui est le File. Les fonctions ParseString et EnvoieCommande sont décrites ci-dessous

- La partie Main qui permet d’initialiser la variable partagée et le mutex nécessaire à la protection de celui-ci, mais aussi de lancer les 8 threads qui permet de contrôler la télécommande, le pseudo code suivant décrit son fonctionnement :

```
structure Argument
    mutex_file: mutex
    fichierSource : File *
end structure

Function Main
    Variable
        args : Argument
        mutex_file: mutex
        fichierSource : File *
        threads_Id[8]: pthread_t
        iter : entier
    Début
        args.mutex_file = mutex_file
        args.fichierSource = fichierSource
        iter = 0
        Tant que iter < 8 faire
```

```
            pthread_create(&threads_Id[i], NULL, &ThreadChangementChaine, *args)
            iter = iter + 1
            fin tant que

            iter = 0
            Tant que iter < 8 faire

            pthread_join(&threads_Id[i], NULL)
            iter = iter + 1
            fin tant que
        Fin
```

- La partie parseString permet de parser les informations récupérées dans le fichier CSV. Cette fonction permet de récupérer l'identifiant de chaîne à afficher, le début de l'enregistrement et la fin de l'enregistrement
- La partie envoieCommande permet de convertir l'identifiant de chaîne en numéro de chaine puis d'envoyer au bon tuner la chaîne à afficher

Normalisation Station TV

Problème :

Durant le développement de la fonctionnalité de recalibrage et de paramétrage, nous nous sommes aperçus que l'un des tuners n'était pas le même. En effet, le menu de celui-ci était différemment agencé, par conséquent les suites de commande que nous avions envoyé produisaient un résultat différent.

Solution :

Pour régler ce problème, il nous a été demandé de changer le tuner concerné par un tuner identique aux autres.
Nous avons remplacé le tuner 8 qui est associé à la vue 1.
Nous avons aussi mis à jour le fichier de configuration qui fait le lien entre les tuners, les id-phidget et les numéros de vue.

En débranchant le tuner, nous nous sommes aperçus que le tuner n'avait pas le même numéro de série que les autres : 011128.3 pour le tuner 8 et 011128.2 pour les autres.

Seul le tuner 8 a été modifié, le phidget et la carte de capture associé sont restés les mêmes, car ils étaient déjà fonctionnels.

Commentaires :

Malheureusement, après un test de conformité, nous avons découvert que deux vues sont défaillantes : les vues 3 et 5.

Nous avons essayé de changer les Tuners et les phidgets pour les deux mais sans succès. Il est possible que le problème vienne du mauvais alignement des phidgets avec les tuners.

Configuration de la Station TV

Voici l'actuelle configuration de la Station TV, le fichier de configuration reprend ces mêmes éléments et les stocke sous format CSV pour qu'il soit lisible par le code.

N° Tuner	N° phidget	N° de la Vue
1	564128	1
2	563437	2
3	563923	4
4	563968	3
5	564140	8
6	563369	7
7	564225	5
8	321005	6

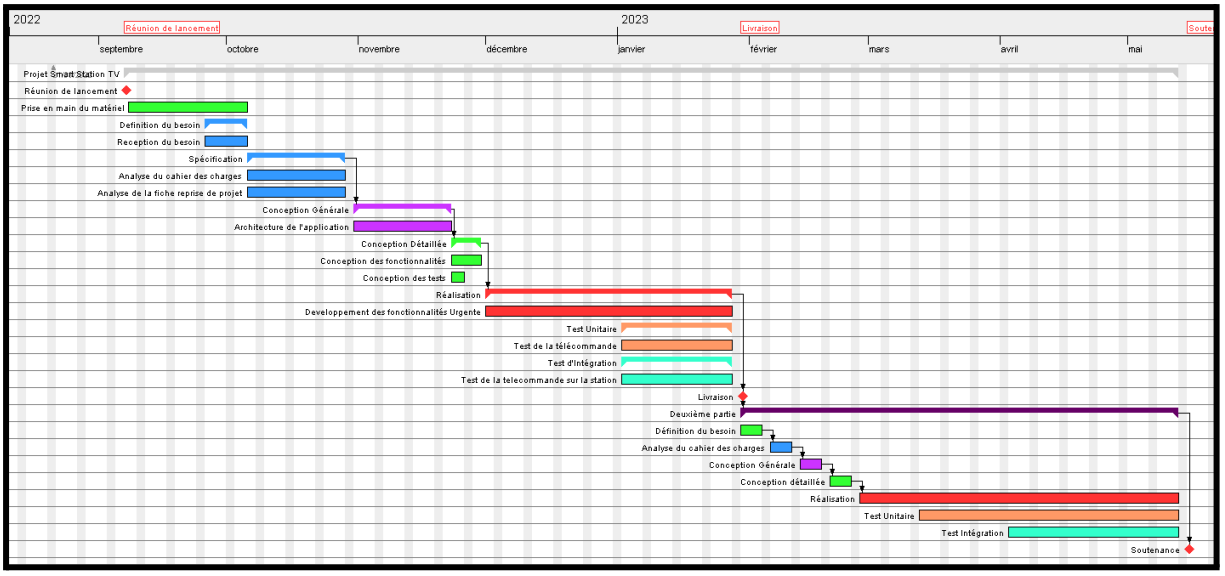
VI-Gestion de projet

Pour la partie Gestion de projet de notre projet, nous sommes partis sur un cycle de développement en V. Ce cycle de développement nous paraissait être le plus pertinent au vu de la taille et du temps que nous avions pour réaliser le projet.

Nous avons aussi réalisé une planification au début du projet et une planification en fin de projet afin de nous rendre compte de la différence entre l'attendue et le réalisé.

Ci-dessous voici la planification prévisionnelle de nos tâches :

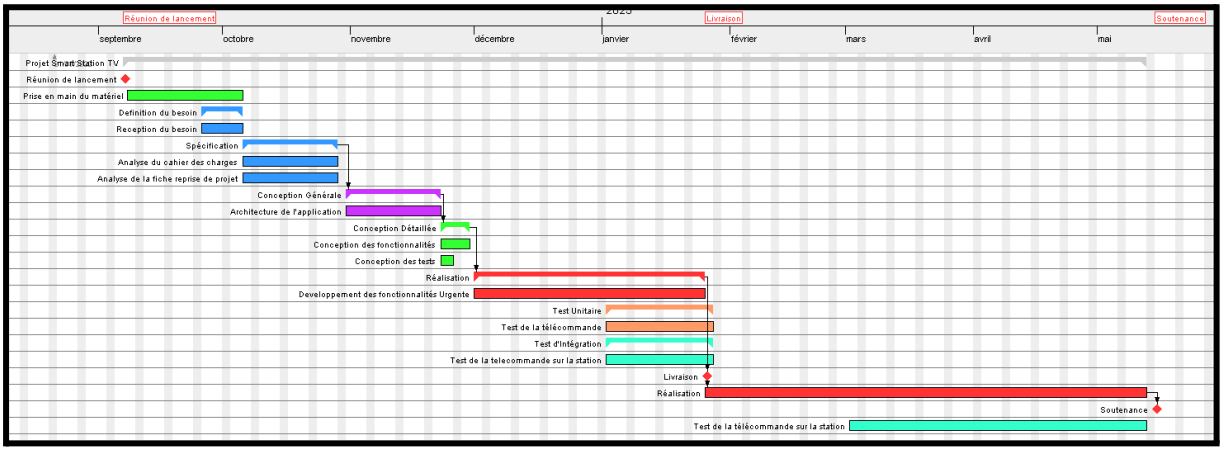
Premier Planning Prévisionnel, réalisé au début du projet :



Nous comptions réaliser deux cycles de développement complet, car deux parties avaient été identifiées en amont du projet avec le client.

Le premier cycle permettait de réaliser les nouvelles fonctionnalités à ajouter à la télécommande et le deuxième cycle permettait de réaliser la solution du PRD.

Second Planning Prévisionnel, réalisé après notre première soutenance



Malheureusement, nous n'avons pas pu réaliser deux cycles entiers, car nous avons manqué de temps, mais aussi parce que les objectifs du projet ont été modifiés au milieu du projet.

Pour la deuxième partie du projet, nous avons donc changé d'objectif pour partie sur la conception d'un outil de mesure de la smart latence et non l'intégration de la solution du PRD.

VII-Livrable

Voici les différents éléments que nous allons livrer avec le rapport.

IHM

- Un dossier contenant un exécutable de l'IHM et le fichier de configuration.
- Un dossier contenant tout le code permettant de générer l'exécutable

Télécommande

- Le nouvel exécutable de la télécommande
- Un dossier contenant tout le code permettant de générer l'exécutable

Smart Latence

- Un dossier contenant le fichier python permettant d'évaluer la latence et le fichier de configuration des tuners.

VIII-Conclusion

Pour conclure, nous avons apprécié travailler sur ce projet intéressant et fort de sens dans le contexte sociétal où il prend place. Au cours de ce projet, nous avons continuellement affiné notre compréhension du sujet et rectifié nos outils en accord.
Malgré le fait que nous ayons pris du retard sur nos échéances prévues dans notre planning prévisionnel et que certaines parties se soient révélées plus complexes que prévu, nous avons réussi à livrer les fonctionnalités les plus critiques.

Pour ce qui est de la solution du PRD, avec plus de temps, nous aurions sûrement pu livrer une fonctionnalité fonctionnelle, le travail préliminaire que nous avons effectué n'est cependant pas perdu, car il nous aura permis de nous replonger dans nos cours de système d'exploitation.

Nous sommes donc satisfaits de notre travail effectué et des connaissances que nous avons pu acquérir tout au long de ce projet.