PFE IMA5

Jukebox multi-pièces

Alexandre Jouy

Julien hérin

Table des matières

Introduction

Le but de ce projet est de réaliser un jukebox multi-pièces moderne. Concrètement, cela consiste à réaliser un dispositif pouvant jouer de la musique streamée depuis un smartphone, une tablette ou un ordinateur, et ce dans plusieurs pièces à la fois. Dans ce rapport nous vous présenterons tout d'abord la méthode de travail que nous avons choisis, puis nous vous parlerons de l'architecture du module pour finir par l'avancée actuelle du projet.

1. Méthode de travail

Dans un premier temps nous nous sommes réunis avec nos tuteurs, Rodolphe Astori et Alexandre Boé pour faire un point de départ sur le projet, ainsi que la manière dont nous allions nous y prendre.

On nous a alors proposé de travailler selon la méthode Scrum dont le principe général est de découper la durée totale du projet en plusieurs « sprints » beaucoup plus courts d'une quinzaine de jours. Un objectif est fixé au début de chaque sprint et nous devons essayer de s'y tenir, en n'avançant que dans la direction donnée. A la fin de chaque sprint une réunion est organisée de manière à faire le point sur ce qui a été fait, définir les objectifs du prochain sprint et éventuellement corriger la trajectoire.

Les deux premières semaines de notre projet ont été consacrées aux recherches préliminaires et nous avons commencé le fonctionnement par sprint la semaine du 15 octobre.

2. Architecture des modules

2.1. Présentation des modules

Pour clarifier les choses, nous appellerons « module » le haut-parleur et ses équipements embarqués, la partie applicative pour streamer la musique étant mise de côté.

Dans le cadre de ce projet, nous sommes partis sur deux types de module distincts puisque nous souhaitons proposer au public une solution modulaire et non fermée. Dans les deux solutions, nous avons choisi d'ajouter à la Raspberry Pi une carte son USB afin d'améliorer considérablement la qualité sonore que nous envoyons vers le haut-parleur.

2.2. Type de module 1

Le premier module embarque une Raspberry Pi avec MusicBox pour la réception de la musique ainsi qu'une carte son USB.

Cette solution permet à l'utilisateur de brancher n'importe quel haut-parleur actif à la carte son avec un connecteur jack 3.6 mm. Il pourra ainsi garder son propre haut-parleur s'il le désire.

2.3. Type de module 2

Le second module embarque lui aussi une Raspberry Pi avec MusicBox ainsi qu'une carte son USB. Cependant nous proposons en plus dans celui-ci un amplificateur audio auquel nous connecterons un haut-parleur passif.

L'avantage de cette solution est de maîtriser la qualité audio du haut-parleur pour notre utilisateur et de ne fournir qu'une seule « boite » avec solution toute intégrée.

Schéma de principe de la chaine de transmission

2.4. Une solution DIY pour des coûts maitrisés

Un des objectifs importants de ce projet est le coût puisque nous verrons par la suite après établissement d'un état de l'art (Sprint 2) qu'un haut-parleur multi-room dans le commerce se trouve dans une fourchette de prix allant de 100 à 500 euros. Il est donc évident que l'achat d'une enceinte par pièce revient très rapidement assez cher.

Notre but sera donc d'obtenir un module ayant un coût maximum de 80 euros pour une solution prête à l'emploi. La modularité du projet permettra à l'utilisateur de mettre le prix qu'il souhaite pour la qualité sonore de son choix (un haut-parleur actif pour le module 1, un couple amplificateur/haut-parleur pour le module 2).

Le danger majeur porte sur la qualité audio qui diminue très rapidement quand le coût baisse. La somme de 80 euros est un bon compromis pour notre projet.

Nous avons aussi vu très récemment que la Raspberry Pi Zero à 6 euros d'une puissance suffisante pour notre projet était disponible, contre une trentaine d'euros pour la solution actuelle. Elle a également l'avantage d'être beaucoup plus compacte que le modèle B+.

Cette solution permettra dans le futur de pouvoir faire diminuer le coût total du projet pour une personne voulant le réaliser par la suite.

3. Déroulement du projet

3.1. Recherches préliminaires

Les deux premières semaines ont été consacrées aux recherches préliminaires ainsi qu'à réfléchir aux solutions à mettre en place pour le mener à bien. Nous nous sommes convenus qu'il fallait développer :

- Une enceinte embarquant un amplificateur audio et un système pour recevoir le wifi
- Une application sur smartphone permettant ce choisir et streamer le flux musical
- Une application sur PC permettant la même chose

Pour l'enceinte, nous avons choisi d'utiliser une Raspberry Pi disposant d'une distribution Linux embarquée et permettant ainsi de lui ajouter simplement une clé Wifi ainsi qu'une carte son USB. Cette solution nous est naturellement venu en tête car elle est simple à mettre en œuvre et relativement peu coûteuse.

Du côté de l'amplification nous avons retenu en premier temps deux solutions envisageables. La première était un amplificateur à base de LM386. Cette solution est peu coûteuse et extrêmement simple à réaliser mais ne permet d'obtenir qu'une puissance de sortie faible et un son de qualité médiocre (annexe 1). La deuxième était un amplificateur à base de LM3886 qui permet d'avoir une bonne puissance de sortie et un son de bonne qualité au détriment d'être un peu plus coûteuse (annexe 2).

D'un point de vue logiciel nous nous sommes dans un premier temps intéressés à l'application sur PC. Nous pensions développer l'application en C++ en utilisant Qt, un framework graphique fonctionnant sur toutes les plateformes (Windows, Linux et Mac OS).

Nous avons également discuté avec nos tuteurs de manière à définir les points essentiels constituant le cahier des charges de notre projet. Le projet doit être :

- Modulaire
- Entièrement réalisable soi-même
- Relativement peu cher
- Open source

L'aspect modulaire est défini par le fait de pouvoir inter changer facilement les différentes parties comme l'amplificateur, les enceintes, la cartes son, etc… L'objectif à long terme étant un produit dont le coût est maitrisé et qui correspond à chacun.

3.2. Sprint 1

A la fin des recherches préliminaires nous avons alors défini les objectifs à atteindre en fin de sprint 1. Il s'agissait d'arriver à streamer une musique depuis un appareil quelconque, vers une Raspberry Pi Connectée à une enceinte.

Nous avons alors pris la décision de partir sur une application destinée à fonctionner sur un PC. Nous avons choisi de développer cette application en C++, en utilisant le framework Qt, car il permet la réalisation de programmes graphiques sur toutes les plates formes.

On a alors commencé à développer une application en C++ tout en cherchant des méthodes pour arriver à streamer simplement sur la Raspberry Pi. Une des voies envisagée était d'utiliser une librairie VLC intégrable à Qt disponible sur le site vlc-qt.tano.si. L'avantage de passer par VLC pour streamer un flux musical est que c'est un logiciel open source, disponible à la fois sur Windows, Linux, Mac OS et qui s'intègre parfaitement à Qt.

On a également réussi à streamer un flux musical depuis VLC, ainsi que depuis un Mac, en installant SharePort, vers la Raspberry Pi.

A la fin du sprint, on a alors effectué un retour sur ce qu'on avait fait avec nos tuteurs ainsi que Jean-Eude Laurenge. On n'a malheureusement pu faire la démonstration que du stream depuis un Mac.

On a alors rediscuté de l'avancement du projet et des objectifs futurs. En effet, la démonstration n'était pas convaincante, et assez restreinte par rapport au résultat attendu.

On a déterminé ensemble qu'il était alors nécessaire pour le sprint 2 de :

- Faire une veille technologique complète sur les solutions existantes
- Se concentrer sur l'application mobile dans un premier temps.

En effet, l'intérêt principal de pouvoir streamer de la musique dans plusieurs pièces à la fois est de pouvoir contrôler la musique tout en se déplaçant. L'intérêt de contrôler le flux musical à l'aide d'un PC est donc assez limité.

On a alors défini comme objectifs pour le sprint 2 :

- Faire un état de l'art des technologies existantes
- Arriver à créer une chaine de transmission complète du flux musical (du mobile jusqu'aux enceintes)

3.3. Sprint 2

Dans un premier temps il s'agissait de réaliser une veille technologique décrivant les solutions existantes autant au niveau logiciel et matériel DIY que celles commercialisées par des marques connues. Il fallait également arriver à streamer de la musique vers au moins une enceinte en utilisant les solutions trouvées.

Les solutions commercialisées (sous le nom d'enceintes multiroom) sont les suivantes :

- Sonos, haut de gamme, à partir de 229€
- Samsung, milieu de gamme, à partir de 99€
- Bose, haut de gamme, à partir de 199€
- LG, milieu de gamme, à partir de 179€
- Yamaha, haut de gamme, à partir de 299€
- Google Chromecast, entrée de gamme, à partir de 39€

Sonos


Samsung


Bose


LG


Yamaha

L'intérêt de la Chromecast par rapport à toutes les enceintes existantes, c'est qu'elle permet de transformer n'importe quelle enceinte en système intelligent

Nous avons également fait des recherches sur les logiciels existants (autres que les logiciels constructeurs des enceintes précédemment présentées) permettant de streamer de la musique. Ces recherches se sont portées sur les principaux systèmes d'exploitation, et plus particulièrement pour les mobiles.

En effet, nous nous sommes convenus avec nos tuteurs qu'il était judicieux de privilégier en priorité l'application mobile plutôt que commencer par développer une application pour ordinateurs. On a alors répertorié les solutions déjà existantes qui paraissaient être les plus performantes et les moins compliquées à mettre en place :

- Pour mobile :
    - Android : BubbleUPnP
    - Iphone : AirPlay (intégré)

- Pour PC :
    - Windows : Windows Media Player (intégré)
    - Mac OS : AirPlay (intégré)

Pour la Raspberry Pi nous avons choisi d'installer un OS Linux modifié nommé MusicBox. Ce système d'exploitation est prévu pour gérer les flux musicaux. En effet, il intègre un serveur DLNA, la possibilité d'y associer un compte Spotify ou encore de streamer un flux Youtube ou une musique située sur une clé USB branchée à la Raspberry. Il dispose également d'une interface web qui permet de contrôler facilement tous les flux et son code est entièrement open source. C'est donc la solution idéale pour streamer facilement de la musique depuis un mobile en utilisant le DLNA.

Lors de la réunion de fin de sprint nous avons effectué une démonstration d'une chaine complète. Nous avons réussi à diffuser deux musiques différentes depuis un même portable vers deux enceintes séparées. L'objectif fixé en début de sprint est donc atteint et nous avons pu définir de nouveaux objectifs pour le sprint suivant :

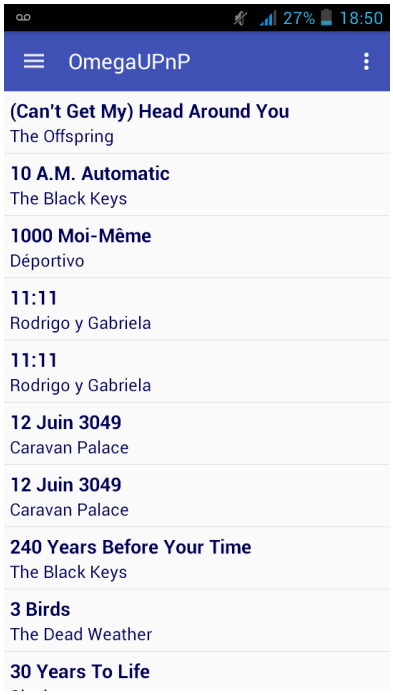
- Réaliser notre propre application et arriver à streamer une musique vers une enceinte
- Réaliser le PCB du circuit d'amplification à base de LM3886

L'intérêt de réaliser une application Android pour streamer de la musique est qu'elle soit open source, contrairement à BubbleUPnp.

3.4. Sprint 3

Pour le sprint 3 nous nous sommes concentrés sur la réalisation d'une application Android. En effet, comme iOS dispose d'une solution intégrée déjà existante nous avons jugé n'il n'était pas nécessaire de développer une application dédiée.

Nous avons choisi de développer notre application en utilisant Android Studio. Cette application est donc réalisée en Java, et nous avons fait en sorte qu'elle soit compatible avec tous les téléphones disposant de la version 4.1 (Jelly Bean) ou plus. Cela permet qu'un maximum d'utilisateurs puisse profiter de l'application tout en la rendant compatible avec les versions récentes.



Première version du lecteur

Dans un premier temps nous avons créé une activity Android standard disposant d'un panneau latéral qui nous permettra par la suite de rajouter les fonctions de recherche de serveur DLNA.

Dans un premier temps nous avons créé une classe Song qui contient les variables liées à chaque chanson et les méthodes nécessaires pour pouvoir les manipuler.

Il a ensuite été nécessaire de créer une autre classe contenant les méthodes permettant de manipuler les chansons. Cette classe, nommée MusicService, permet de lier la liste des chansons que l'on a récupérées dans l'Activity principale au lecteur de média. En effet, le

lecteur de média est un service sous Android et est donc indispensable à lier si l'on veut lire un média quelconque.

Une fois ces étapes faites, nous avons affiché cette liste dans l'activité principale et l'avons trié par ordre alphabétique. On initialise alors le service permettant de lire les médias et on lit l'action de toucher une chanson dans la liste au démarrage de ce service. On a alors une première version du lecteur musical, qui pour l'instant est limitée à la lecture en local.

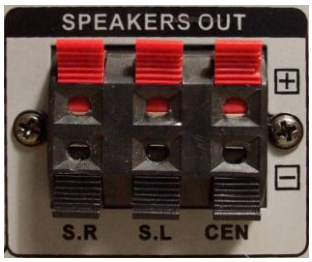
On a ensuite implanté une barre permettant de contrôler la lecture (lecture, pause, chanson suivante…). Les contrôles étant déjà présents dans le service « Media Player » il était seulement nécessaire de rajouter le widget contenant notre barre à l'activité principale puis de lier les boutons aux contrôles.

Nous avons également réalisé le layout des deux PCB du module 2 : un pour l'amplification à base de LM3886 et un pour le circuit d'alimentation. Afin de dissiper sa chaleur importante lors de l'utilisation de l'amplificateur, nous avons commandé un dissipateur thermique sur lequel nous visserons le LM3886.

Nous nous sommes fortement inspirés de l'amplificateur en annexe 2 où les informations sur le site correspondant sont très détaillées et nous permettent de comprendre l'importance de beaucoup de composants dans un amplificateur audio.

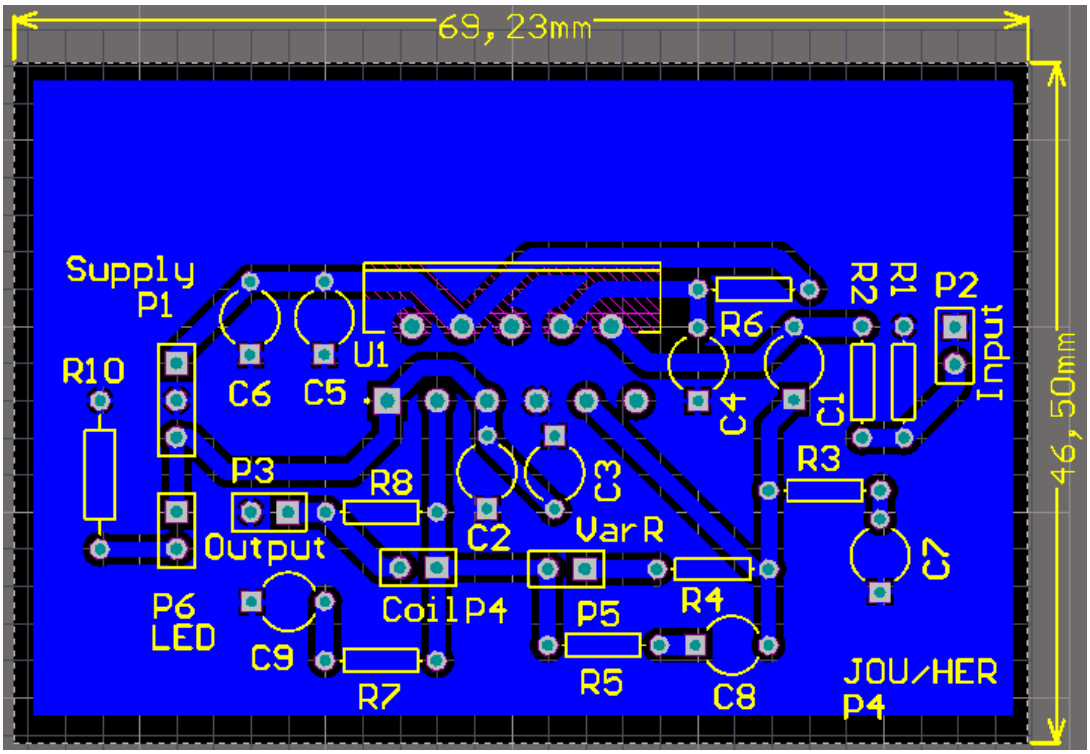
Par rapport au circuit d'origine, nous avons ajouté une LED de fonctionnement de l'amplificateur, un switch ON/OFF ainsi qu'un potentiomètre rotatif pour pouvoir agir sur le volume sonore.

Pour l'entrée audio, l'utilisateur trouvera en façade de l'amplificateur une entrée stéréo RCA ainsi qu'une entrée jack 3.6 mm. Pour la sortie, nous avons prévu un bornier standard pour connexion de haut-parleur ainsi qu'une sortie jack femelle 3.6 mm.



Bornier 2.1

La version quasiment finale du PCB est donc la suivante :



Les objectifs de ce sprint n'ont donc pas tout à fait été atteints :

- Le lecteur Android ne permet pour l'instant que de lire localement les musiques
- Le PCB n'a pas encore été tiré

Il est donc nécessaire pour le sprint suivant de corriger au plus vite ces deux points de manière à ce que le projet puisse aboutir. Les objectifs pour le sprint suivant sont donc de faire en sorte d'arriver à streamer un flux musical vers une enceinte avec notre application ainsi que de tirer les PCB, les souder et commencer les tests.

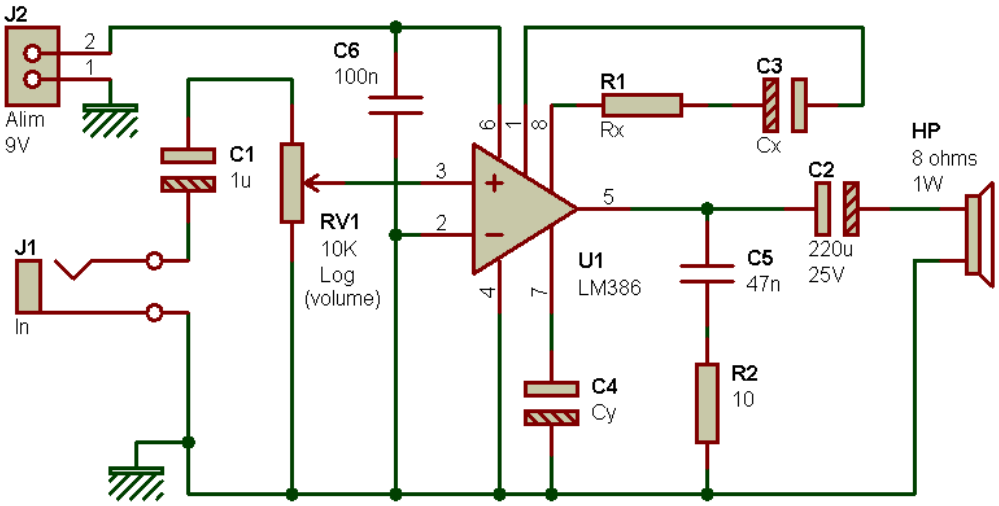
Conclusion

L'avancée par sprint est une méthode de travail que nous avons découvert au cours de ce projet, et donc qui n'est pas forcément toujours évidente pour nous à mettre en œuvre. Ainsi, nous avons pris du retard dans l'avancement du sprint 3 qui a pris beaucoup plus de temps que prévu sans pour autant arriver aux objectifs fixés. Nous avons donc pris rendez-vous avec nos tuteurs qui nous ont incité à mettre en place un planning précis pour la suite du projet (disponible en annexe).

Ce retard est également dû au développement d'une application Android, qui est à la fois nouveau pour nous et qui n'entre pas dans les parties que nous souhaiterions mettre en avant en finalité de notre projet. Nous sommes donc en train de chercher d'autres moyens d'enrichir notre réalisation et ainsi d'aboutir à un produit réfléchi et original.

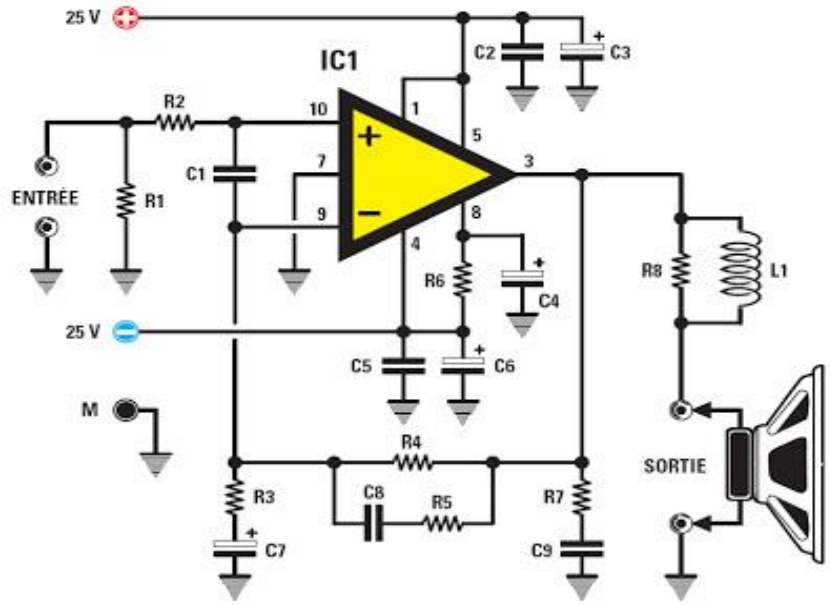
Annexes

1. Schéma de l'amplificateur à LM386



Pour un gain de 20 (26 dB) : ne rien raccorder aux broches 1 et 8 (les laisser en l'air)
Pour un gain de 20 à 200 (26 dB à 46 dB) : résistance R1 + condensateur C3 entre broches 1 et 8.
Exemples :
- Pour un gain de 50 (34 dB) : R1 = 1K2 et C3 = 10uF
- Pour un gain de 200 (46 dB) : Condensateur seul entre broches 1 et 8 - C3 = 10uF (R1 = 0)

2. Schéma de l'amplificateur à LM3886



Planning

Nom	Date de début	Date de fin
Recherche solutions existantes	21/09/15	09/10/15
Cahier des charges	21/09/15	09/10/15
SPRINT 1 : Veille Techno	05/10/15	09/10/15
SPRINT 1 : Retour/Trouver solution And...	12/10/15	16/10/15
Tests BubbleUPnP+MusicBox	19/10/15	23/10/15
Soft : Definir fonctions principales	26/10/15	06/11/15
Soft : Création liste musique+lecture lo...	09/11/15	12/11/15
Soft : Stream UPnP	16/11/15	01/01/16
Soft:Music Controller	07/12/15	11/12/15
PCB(Ampli):Réalisation	23/11/15	08/01/16
PCB(Ampli) : Conception	23/11/15	11/12/15
PCB(Ampli) : Fabrication	14/12/15	18/12/15
PCB(Ampli) : Soudure + Tests	04/01/16	08/01/16
HP : Design des modules au FabLab	11/01/16	15/01/16
Soft : Stream vers plusierus serveurs	04/01/16	15/01/16
PCB(Alimentation) : Conception et fabr...	18/01/16	22/01/16