UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et génie informatique

PROJET S5: RAPPORT FINAL

Présenté à
Jean-Baptiste Michaud

Présenté par
Alexandre Girard – gira2113
Alexandre Guay – guaa2102
Frédéric Berthelot – berf2308
Alexandre Thibeault – thia2
Gabriel Guilmain – guig2709
Luis Felipe Anillo – anil2301
Louis-Philippe Bardier – barl2407
Jean-Nicolas Crête – crej2104

21 avril 2017

Table des matières

1 INTRODUCTION

Dans le cadre du cours de projet, il était question d'approfondir trois aspects de la profession d'ingénieur électrique : les systèmes embarqués avec traitement numérique des signaux, le développement de produits et la gestion de projets. En vue de produire un système comportant une télécommande et un DSP, il a été décider de faire un prototype permettant de composer de la musique avec des traitements de signaux numériques effectués sur un piano électronique.

2 PRÉSENTATION DU PROTOTYPE

2.1 DESCRIPTION GÉNÉRALE

Notre produit est une plateforme permettant la détection de certaines notes et accords au piano. En plus de permettre la détection, celui-ci permet l'affichage temps réel des notes ou accords détectés et même l'affichage temps réel de la partition jouée. Ces fonctionnalités sont celles du prototype. Si le projet se poursuit, le mode apprentissage sera possible avec écoute du joueur et conseil, génération de mélodie, création de partitions en temps réel avec accords, etc. Finalement c'est un produit d'aide aux musiciens qui permet de composer de la musique et de s'améliorer.

Il est à noter que pour le présent prototype, ce n'est pas toutes les fonctionnalités qui sont implémentées. L'équipe s'est tout d'abord concentrée sur les fonctionnalités importantes : la détection de notes et accords (pas toutes les notes et accords possibles pour le moment) ainsi que l'affichage des partitions jouées de façon le plus minimaliste possible sur une console. Dans le vrai produit, cette dernière fonctionnalité affichera des partitions visuellement de qualité.

2.2 APPLICATION

Premièrement, plusieurs personnes composent de la musique et doivent arrêter de jouer pour écrire la partition qu'ils ont jouée afin de ne pas oublier une section ou une idée qu'ils ont eue. Notre produit s'adresse principalement à ceux-ci. En fait, il écoutera le joueur et écrira en temps réel la partition jouée !

Ensuite, notre produit peut aussi être utile pour des personnes qui veulent pratiquer l'oreille musicale. Avec la détection d'accords ou de note, le joueur peut jouer une note au hasard et essayer de l'identifier et vérifier par la suite avec le programme. Aussi pour valider que le joueur fait bien les bons accords. Par exemple, en jazz, on écrit seulement les accords et non les notes qui les composent. Alors il sera possible de vérifier en temps réel.

Finalement, le produit pourrait aussi utile pour les utilisateurs qui veulent apprendre des chansons. On peut jouer une pièce et vérifier la partition par la suite si c'est bien exécuté en la comparant à l'original. Éventuellement, cela pourrait se faire automatiquement et le programme pourrait donner des commentaires et conseils.

2.3 FONCTIONS PRINCIPALES

Plus spécifiquement, notre produit possède certaines fonctions :

- L'affichage temps réel de notes et accord joués au piano ;
- L'affichage temps réel d'une partition jouée (composée seulement de notes singulières) ;
- L'enregistrement des partitions que l'on joue ;
- Génération du son d'un métronome (tempo), et
- Réglage de la vitesse du métronome (tempo).

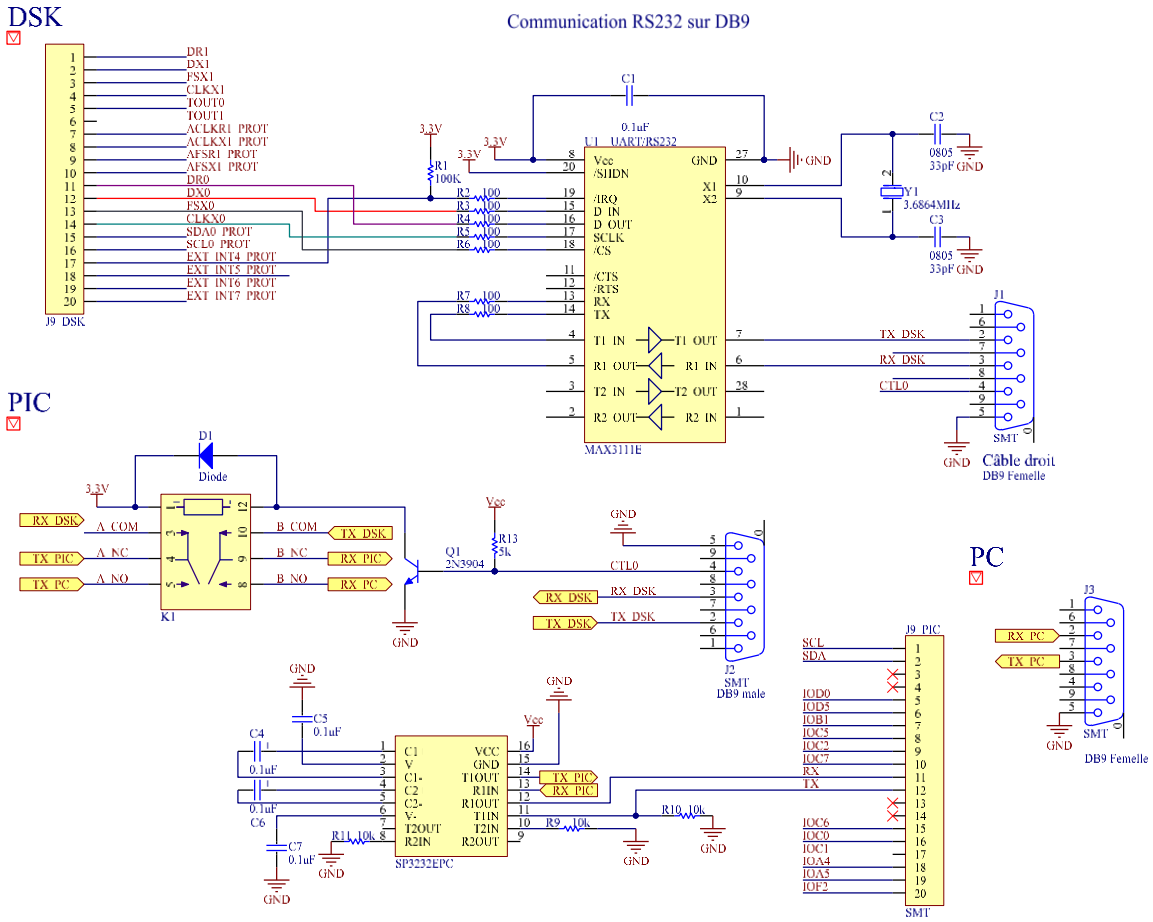
3 ASPECTS TECHNIQUES

3.1 ARCHITECTURE HAUT NIVEAU

À haut niveau, les éléments du projet qui interagissent pour créer le système global sont le DSK qui interagit avec la télécommande (PIC) via RS-232 et à une console sur ordinateur via le même protocole de communication. Le CODEC audio du DSK permet de recevoir le signal audio du micro via son ADC et de le transmettre au DSP pour effectuer l'analyse. De même, le DAC du CODEC permet d'envoyer un signal synthétisé dans le DSP pour créer un son de métronome à fréquence ajustable. Le tout est visible dans le document : L3 Diagrammes & Ordinogrammes\AUTRES\Architecture matérielle haut niveau du prototype.

L'architecture du système est relativement simple si on observe la fonction MAIN du DSK : dans la boucle, si une nouvelle donnée du RS232 est reçue, on en fait la lecture et on fait le choix de l'action à effectuer en conséquence dans la fonction suivante : L3 Diagrammes & Ordinogrammes\DSK\ RECEPTION_RS232 qui reçoit les messages de la télécommande. Dans le mode asynchrone, on vérifie deux choses : si l'interruption du CODEC envoie le flag indiquant que la trame de test d'intensité est prête, cette trame est passée dans la fonction qui vérifie l'intensité du son. Si ce test est passé, le CODEC fait l'acquisition de la vraie trame. Quand cette trame est prête, un autre flag est déclenché et le traitement pour le mode de détection de note asynchrone est effectué jusqu'à l'envoi du résultat à la télécommande. Si la télécommande décide que le traitement se fait en mode synchrone avec partitions, les données seront échantillonnées de façon parallèle aux battements du métronome dans l'interruption. De la même manière que pour le mode asynchrone, un flag sera alors déclenché et le code du MAIN activera le traitement pour ce mode jusqu'à l'envoi de la partition à la console d'ordinateur. Le tout peut être observé dans le diagramme de fonction du code principal du DSK : L3 Diagrammes & Ordinogrammes\DSK\FONCTION_MAIN.

3.2 SCHÉMA DU PROTOTYPE



Dans notre cas, le DSK doit communiquer avec deux périphériques. Nous avons utilisé une logique à relai semblable à celle de l'APP pour arriver à nos fins.

Pour permettre au DSK de communiquer avec l'écran, il faut utiliser un MAX3111 pour transformer le SPI en UART. Ce même signal peut aussi être dirigé vers le PIC avec un relais. Cependant, il faut utiliser le SP3232 qui convertira le UART en RS-232.

Le traitement de signal provenant du PIC sera similaire, le RS-232 sortant de ce dernier sera converti en UART avec le SP3232 pour communiquer avec l'ordinateur et le UART allant vers le DSK sera converti en SPI avec le MAX3111.

3.3 STRUCTURE DES INTERACTIONS

Pour l'usage de la télécommande, un menu a été conçu qui permet de choisir les modes qui seront utilisés. Dans notre cas, trois modes différents étaient présents, le mode asynchrone qui comporte le mode note et le mode accord ainsi que le mode synchrone. Pour chacune de ses commandes, le DSK doit fonctionner d'une manière différente. Par conséquent, la télécommande se doit d'envoyer des messages qui précisent le mode dans lequel le DSK doit se retrouver.

Pour se faire, la télécommande envoie un byte précis selon les modes qui est détecté par le DSK. Chacun de ses trois bytes (un par mode) a été convenu à l'interne entre les membres de l'équipe. Ces messages sont envoyés par l'UART du PIC. Ce signal est ensuite converti en RS-232. La lecture sur le DSK est faite par l'UART.

De plus, pour le mode synchrone, qui permet d'afficher la partition jouée sur la console d'un ordi, la télécommande n'a plus d'accès au RS-232 puisque celui-ci est donné à l'ordinateur par un relais. Pour savoir que la télécommande peut sortir du mode synchrone, le DSK envoie un message par SPI qui est reçu sur l'UART d'un byte qui est un code qui dit à la télécommande de changer de mode.

Aussi, lorsque le système est mode asynchrone, une communication entre le DSK et la télécommande est nécessaire pour l'affichage. La télécommande reçoit un message du DSK pour chaque note ou accord qui est identifié. Pour se faire, chaque note et accord possèdent un code bien précis qui a été décidé à l'interne par l'équipe. À chaque réception, la télécommande décode le message et affiche la note qui est jouée.

Finalement, la dernière communication qui est utilisée se produit entre l'écran LCD et le PIC. La communication SPI est utilisée pour envoyer des paquets de 3 bytes à l'écran. Selon l'action désirée, un message spécifique est demandé par l'écran.

Un ordinogramme est présent et montre les interactions entre le PIC et le DSK.

3.4 UTILISATION DES INTERRUPTIONS

Du côté DSK, les interruptions sont utilisées comme suit : l'interruption du CODEC audio (déclenchée à chaque fois que la donnée précédente a été envoyée) sert à faire l'échantillonnage pour l'enregistrement dans les trames. Le filtrage et l'application d'une fenêtre de Hamming sont également faits en temps réel. L'interruption sert également à générer le sont de métronome pour le mode synchrone et à déclencher des flags indiquant quand il faut faire l'échantillonnage en mode synchrone. Le tout peut être observé dans le ficher : L3 Diagrammes & Ordinogrammes/interruption_CODEC.

Également du côté DSK, l'interruption sur le GPIO 4 est activée par le module UART MAX3111E qui lorsqu'il reçoit une nouvelle donnée, déclenche l'interruption. Dans le code, cette interruption envoie un flag indiquant au programme principal qu'une donnée est prête.

Du côté de la télécommande, deux interruptions sont utilisées. Une première interruption provient du timer, est périodique aux 25 ms et sert à cadencer les acquisitions clavier. Sa période est assez élevée pour assurer un bon debouncing. Une seconde interruption provient du module UART lors de la réception d'un byte. Cette interruption sert à faire l'acquisition du byte à temps.

3.5 TRAITEMENT DE SIGNAL

Le traitement de signal se divise en deux grandes parties relativement similaires : la partie traitement pour le mode synchrone et la partie traitement pour le mode asynchrone. Du micro jusqu'à la console / écran LCD, le tout est décrit ci-dessous.

Pour ce qui est du traitement dans le mode synchrone, c'est-à-dire le mode qui écrit des partitions, le commence dans l'échantillonnage dans la fonction d'interruption du CODEC audio (L3 Diagrammes & Ordinogrammes\DSK\INTERRUPTION_CODEC). À certains moments précis, c'est-à-dire à chaque demi-battement du métronome, un flag active l'échantillonnage. Il est à noter que pendant l'échantillonnage, le filtrage (filtres passe-bande IIR) et l'application d'une fenêtre d'Hamming s'effectuent en temps réel. La

fonction du métronome est assez simple, à chaque coup d'interruption des compteurs s'activent et décident des moments précis où émettre les sons générer avec une table de sinus. C'est les variables de ces compteurs qui décident du moment où commencer l'échantillonnage. Lorsque l'échantillonnage est terminé, un flag indique au programme principal de démarrer le traitement. La fonction globale de traitement pour ce mode est visible ici : L3 Diagrammes & Ordinogrammes\DSK\TRAITEMENT_MODE_SYNCHRONE. On y voit les étapes du traitement : le calcul de la FFT, la détection de la note, le remplissage de buffer des notes détectées et des amplitudes moyennes des trames, l'analyse du rythme des mesures enregistrées et l'impression des partitions résultantes à la console. Pour l'analyse des notes singulières, les maximums de la FFT sont identifiés, voir la fonction ici : L3 Diagrammes & Ordinogrammes\DSK\ANALYSE_FFT_SING. Ensuite les fréquences associées à ces maximums comprenant la fondamentale et quelques autres harmoniques sont envoyées dans une seconde fonction qui en détermine la note avec quelques conditions. Voir ici : L3 Diagrammes & Ordinogrammes\DSK\ DETECTION_NOTES. Les notes et leurs amplitudes sont enregistrées comme dit plus haut. Lorsque suffisamment de notes sont enregistrées, l'algorithme détectant le temps des notes est appelé. Il se sert du vecteur des notes et du vecteur des amplitudes associées afin de déterminer si ce sont des croches, noires, blanches, rondes, ou silences. Voir l'algorithme ici : L3 Diagrammes & Ordinogrammes\DSK\ ANALYSE_RYTHMIQUE. Finalement, une fonction assure l'écriture des partitions sur la console. Son fonctionnement est très simple : elle vérifie parallèlement le vecteur des notes et le vecteur des timings généré par la fonction précédente afin de générer la ligne correspondant à chaque note. Voir ici : L3 Diagrammes & Ordinogrammes\ DSK\PRINT_PARTITION_CONSOLE.

Pour ce qui est du traitement en mode asynchrone, le tout est similaire. Tout commence par l'ISR du CODEC sauf que cette fois-ci une condition s'applique à l'échantillonnage. Une petite trame enregistrée continuellement doit répondre à un

critère d'amplitude. La structure est similaire au mode synchrone : voir ici : L3 Diagrammes & Ordinogrammes\ DSK\TRAITEMENT_MODE_ASYNCHRONE. Également, étant donné que la vitesse de traitement est moins importante dans ce mode, une autocorrélation ainsi qu'une détection de périodicité assurent que la détection soit plus béton. Voir ici : L3 Diagrammes & Ordinogrammes\ DSK\TEST_PERIODICITÉ. Ensuite dépendamment du résultat de ce test, la FFT est effectuée et dépendamment de si on se trouve en mode accords ou en mode notes singulières, le traitement est effectué. En mode singulière, les mêmes fonctions que pour le mode synchrone sont utilisées. En mode accords un algorithme légèrement différent est utilisé : voir : L3 Diagrammes & Ordinogrammes\ DSK\ANALYSE_FFT_ACCORD et L3 Diagrammes & Ordinogrammes\DSK\ DETECTION_ACCORDS. Ensuite, aucun enregistrement de vecteur de notes n'est effectué comme pour le mode synchrone. Les notes détectées sont directement envoyées au PIC pour l'affichage en direct. Voir la fonction qui effectue l'envoie des messages associés aux notes : L3 Diagrammes & Ordinogrammes\DSK\ PRINT_NOTES&ACCORDS_PIC.

4 PROCÉDURES DE TESTS SYSTÈME

Comme expliqué dans le document de l'assurance qualité de notre projet, les procédures de test de notre équipe s'effectuent en trois passes différentes. Toutefois, suite aux commentaires de la revue 2, nous avons changé un peu la troisième passe d'exécution de test. En effet, il manquait des tests d'intégration qui englobent tous les modules du projet en même temps et c'est donc cela le nouveau but de la passe 3. Alors, nous allons voir quelques exemples de tests pour les deux fonctionnalités au cœur de notre projet, soit la détection de note et la détection d'accords.

4.1 TEST UNITAIRE

Tout d'abord, il est bien important, avant d'effectuer des tests d'intégration avec le produit complet, de s'assurer que chacun des modules fonctionne correctement

individuellement. De ce fait, on a créé une panoplie de test unitaire (passe 1) pour chacun des modules qui testent le module de manière indépendante l'un des autres. On s'assure ainsi que lorsque viendra le temps de tester e produit comme un tout, chaque module fonctionne correctement (si tous les tests sont tous réussis) et on réduit les causes d'erreur si jamais les tests d'intégration échouent.

Un exemple : la spécification 2.T.2.10 requiert la détection de différent temps de note. Pour se faire, un des tests unitaires consiste à faire jouer un Do préenregistré d'une durée d'une noire suivie d'un Ré préenregistré d'une durée d'une noire et ainsi de suite pour finalement vérifier la détection de ces notes. En sortie nous devrions voir que les notes varient entre un Do et un Ré et que chacune des notes dure 1 temps.

4.2 TEST DE RÉGRESSION

Une fois que tous les tests unitaires sont réussis à 100% c'est-à-dire qu'aucun test n'a échoué, il est temps de faire des tests de la passe 2. Effectivement, les tests de la passe 2 consistent à faire des tests sur les modules précédents. Ces tests sont aussi appelés test de régression. Les tests de régression permettent de vérifier que l'implémentation de nouveaux modules n'a pas eu d'effet négatif sur les fonctionnalités d'un module le précédent.

Les tests de régressions constituent en 20% des tests d'un module. Ces tests sont les plus importants parce qu'ils sont ceux qui définissent le plus notre application et sans eux, les critères associés à chacune des spécifications ne seraient pas respectés. Ils doivent absolument fonctionner avant de passer à l'étape suivante.

Par exemple, regardons le cahier de charge, plus précisément la spécification 3.T.1 qui est la tâche « Faire l'interface du menu ». Lorsque la spécification 3.T.2 (faire les fonctionnalités du menu) est conçue, il se peut que la spécification 3.T.2 aille des effets négatifs sur 3.T.1. De ce fait, cela justifie encore une fois pourquoi les tests de régression ont importants.

En outre, on remarque aussi à cet instant quelles spécifications n'ont pas été implémentées. Ainsi, on remarque quels critères ont été laissés de côté et pourquoi. On est donc capable de réviser nos objectifs à l'aide de l'AQ, ce qui est un plus.

4.3 TEST D'INTÉGRATION

Une fois tous les autres tests terminés, il faut tester le tout ensemble, ceci consiste en la troisième passe. En effet, une fois que nous avons constaté que les modules fonctionnaient séparément et que les modules suivants n'ont pas altéré les modules précédents, on peut passer aux tests d'intégration.

Ces tests se divisent en deux catégories soit les tests avec le DSK ainsi que les tests avec intégrale. Les tests intégraux correspondent aux tests utilisant le piano, le micro, le DSK, le PIC ainsi que la console.

Les tests d'intégrations doivent absolument tous fonctionner pour assurer le bon fonctionnement de l'entièreté du projet. En effet, ce sont les tests d'intégrations qui permettent de confirmer que chacun des modules fonctionne les uns avec les autres pour former le projet complet conforme au cahier des charges.

Lors de nos tests avec le PIC, nous avons eu de la difficulté à communiquer et il a donc fallu s'assurer que le PIC envoyait les bons messages et ajouter des spécifications plus précises ainsi que des tests case pour s'assurer que l'assurance qualité couvrait le tout.

5 PROBLÈMES ET CORRECTIONS À APPORTER

La détection devait être ajustée selon le piano utilisé et le micro était de faible qualité. Nous aurions pu utiliser simplement la sortie audio du piano pour éviter les bruits ambiants.

Finalement, un code de détection de défaillance dans la communication aurait pu être implémenté pour permettre une robustesse ainsi que d'accélérer les processus de déverminage.

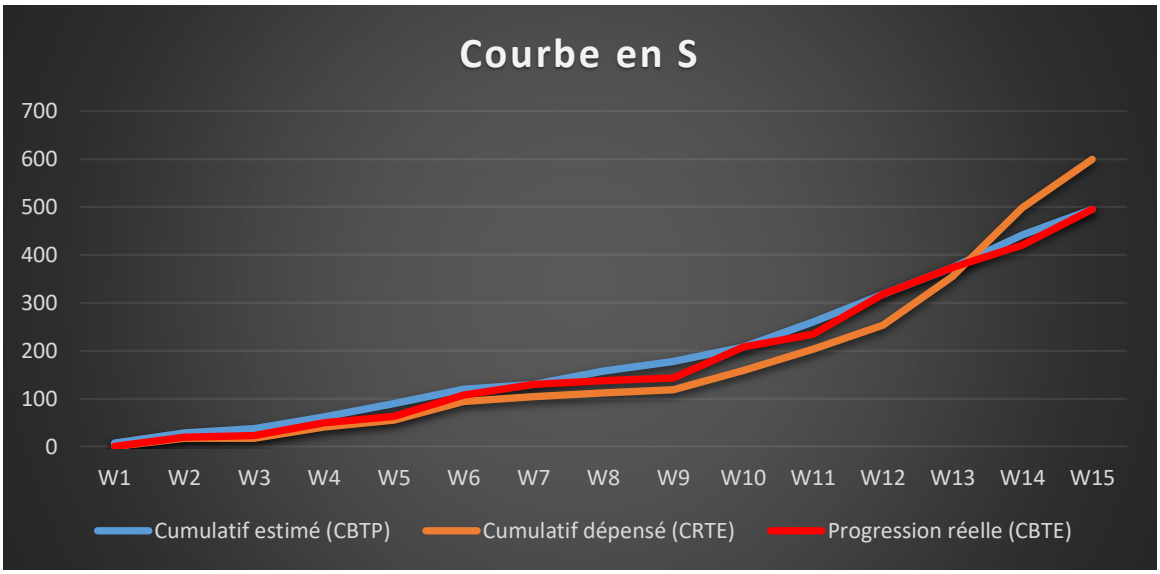
6 AMÉLIORATIONS POSSIBLES DU PROTOTYPE

Dans un but de développement complet, le prototype aurait pu être agrémenté d'autres fonctions, tel que faire rejouer des partitions ou de permettre l'apprentissage des gammes avec des options interactives.

7 GESTION DE PROJET

7.1 COURBE EN S

Afin de mesurer l'efficacité d'un travail ainsi que les déficits en ressources, la courbe en S a été utilisée. Cet outil permet de comparer les estimations en ressources versus les ressources dépensées réelles. Dans notre application, les ressources seront en fait les heures dépensées. La planification a été faite par semaine, pour une période couvrant la session complète. La courbe obtenue est la suivante :



Sur le graphique, il est possible de distinguer trois (3) courbes, soit [1] :

- Le cumulatif estimé (CBTP) ;
- Le cumulatif dépensé (CRTE), et

- La progression réelle (CBTE).

Le coût budgété du travail planifié (CBTP) sert d'estimation moyenne des ressources nécessaires à la réalisation du projet.

Le coût réel du travail effectué (CRTE) représente le coût du projet en temps réel. Elle sera très utile pour évaluer l'efficacité et l'utilisation des ressources.

Le coût budgété du travail effectué (CBTE) représente l'avancement réel du projet. Cette courbe sera utilisée pour vérifier si le projet avance au rythme prévu initialement.

Plusieurs conclusions peuvent être effectuées avec la courbe en S [2]. Tout d'abord, si la courbe de valeur acquise (CBTE) est en haut des coûts réels (CRTE), le projet réalise des économies et inversement proportionnel. Ensuite, si la progression réelle (CBTE) est supérieure au coût budgété (CBTP), le projet est en avance et inversement proportionnel.

À l'aide des courbes de CBTE et de CBTP, on observe qu'en général le projet était un peu en retard et aux présentations ou démonstrations le travail était à jour. En effet, dans un contexte d'étude avec des livrables à produire, tout le travail prévu doit être terminé dans les délais. Selon la semaine d'APP en cours, plus ou moins de temps était consacré dans le projet. L'on peut aussi voir que lors de la semaine de lecture, moins d'heures ont été consacrées au projet.

On peut voir qu'au début du projet, peu d'heures ont été dépensées versus la fin du projet. Dans notre cas, ceci ne relève rien de bien anormal parce que plusieurs compétences requises pour la complétion du projet étaient vues plus tard dans la session et que la semaine de projet a été utilisée dans le but d'avancer intensivement ce dernier. Par contre, ceci a demandé une grande utilisation des ressources en fin de parcours. Aussi, il est possible que les heures approximées pour les tâches aient été sous-estimées. C'est avec l'expérience et une bonne connaissance de son équipe qu'un gestionnaire sera en mesure de budgéter des ressources plus efficacement.

Ensuite, on peut observer que la valeur acquise (CBTE) pour la planification sur la session complète est toujours en dessous de la courbe de coût réel (CRTE) et que les deux

seront égales à la fin. Ceci signifie que le projet n'a pas réalisé d'économie. Ceci peut être dû encore une fois à une surestimation des coûts initiaux.

Finalement, pour être en mesure d'avoir un indice de performance de coût et de délais de meilleure qualité, il aurait fallu bien estimer les heures pour chaque tâche ainsi que de bien répartir le travail tout au long de la session. Il aurait été possible d'obtenir une meilleure efficacité de nos ressources en heures avec des périodes de travail similaire chaque semaine, sans prendre de semaine de repos. Il est plus difficile d'être efficace lorsqu'il faut se remémorer le travail effectué d'il y a quelques semaines et de se replonger dedans.

7.2 TRAÇABILITÉ

Pour s'assurer d'une bonne traçabilité dans toutes les sphères du projet, il a été décidé d'utiliser les logiciels « Trello Plus », « GitHub » et « Lean Testing ». Trello est une plateforme web qui permet de créer des listes contenant des cartes, qui sont en fait des tâches à effectuer. L'extension « Plus » de Trello permet d'ajouter une foule de modules à la plateforme, notamment un outil permettant la gestion des heures planifiées vs effectuées pour chaque tâche et pour le projet en entier. Dans les cartes de Trello, il est possible de taguer des personnes pour leur assigner cette tâche, de mettre une estimation de temps pour la tâche et de mettre une date de début et de fin (et ainsi de générer le Gantt). De cette manière, il est donc possible de savoir qui est associé à quelle tâche et combien d'heures il a mis à l'effectuer. De plus, il est possible de savoir le total d'heures travaillées pour chaque personne, ce qui permet d'essayer d'équilibrer les choses au courant de la session.

GitHub est une application qu'on installe sur notre ordinateur et qui fait le lien avec un serveur sur lequel tous les fichiers sont hébergés. Ce logiciel fait extrêmement bien pour la traçabilité, car il permet non seulement de savoir qui a déposé quel fichier et à quelle heure, mais il permet également d'avoir accès à toutes les versions précédentes de tous les fichiers qui y ont été déposés. Cela fait en sorte que s'il y a une mauvaise

modification d'un code qui est faite, par exemple, il est possible de savoir qui l'a fait, mais en plus la bonne version n'est pas perdue. Ce logiciel est optimisé pour les fichiers texte (fichier .c, .asm, .txt, etc.), car il montre ligne par ligne ce qui a été modifié et est capable de merger 2 modifications qui ont été faites en même temps. Par contre, il fait également très bien pour les fichiers de gestion, par exemple, pour toutes les raisons qui ont été citées plus haut.

Le logiciel Lean Testing, quant à lui, est une plateforme web qui permet le tracking de bug, la description de tous les plans de test et la réalisation de ces tests avec les résultats obtenus. Il est important d'en parler ici, car ce logiciel est très bon au niveau de la traçabilité. En effet, on peut y voir qui a écrit quel plan de test, qui a été désigné pour faire quel test et qui est désigné pour régler quel bug. On peut également y ajouter des captures d'écran, des vidéos ou des descriptions détaillées pour chaque bug, qui resteront en mémoire sur le site, avec la date.

En bref, tous ces logiciels ont grandement aidé l'équipe à comprendre l'importance de la traçabilité dans un projet, que ce soit pour ne pas perdre de version de fichiers importants, pour savoir à qui s'adresser pour le travail sur un module en particulier ou pour garder une trace de l'avancement de projet à chaque date.

7.3 RÉTROSPECTION ET CONCLUSIONS

La gestion de projet de la session S5 s'est en général très bien déroulée, surtout pour la planification des tâches pour arriver prêt à chaque revue et démonstration, en plus que chaque dépôt soit fait dans les temps sans avoir à courir après quelqu'un. Les logiciels qui ont été utilisés ont été bénéfiques et nous ont appris l'importance d'une bonne gestion du temps et de la traçabilité des fichiers et travaux effectués. L'importance d'une bonne courbe en S a été comprise et cela pourra donc être encore mieux fait aux prochaines sessions. Il serait bien pour l'avenir d'améliorer la séparation des tâches, c'est-à-dire de faire en sorte que tout le monde travaille de façon assez égale et que personne ne soit indispensable à une certaine tâche, dans le sens où si une personne est malade ou

absente, qu'au moins une autre personne soit au courant de son travail et puisse le continuer.

8 GESTION DE RISQUES

Pour la première revue, un fichier Excel détaillant les risques a été égayé. Ce fichier a ensuite été pas mal oublié pour tout le reste du projet. Il se trouve néanmoins que les risques du projet ont été bien gérés. Le résultat est un produit fini fonctionnel, sans devoir compromettre les nuits de sommeil des membres de l'équipe.

Le premier facteur que je considère réducteur de risques est la vaillance et l'autonomie des membres. Tout le monde a contribué de façon plutôt égale et à l'avance. Ce facteur ajoute aussi un bon climat dans le groupe, gage de motivation.

Le bon travail des gestionnaires de l'équipe a assuré l'atteinte des objectifs pour chaque revue ou démo. L'outil électronique Trello a aussi franchement aidé cette gestion, en permettant notamment d'afficher les courbes de progrès, d'entrer les heures de travail et de garder un suivi sur les tâches.

D'autres outils électroniques ont permis de réduire des risques. GitHub a permis de garder le code en lieu sûr et a amélioré le partage de code entre les membres. Le site LeanTesting quant à lui a permis d'offrir une bonne interface pour garder une trace des bogues et pour enregistrer les résultats de tests. Ce site, associé avec l'assurance-qualité a permis d'assurer que le code soit 100% fonctionnel et testé. C'est donc le dernier réducteur de risques.

Au final, même si le fichier Excel n'a pas été utilisé, c'est l'exercice de celui-ci qui a amené l'utilisation des outils électroniques et les gestionnaires qui eux ont permis de mitiger les risquer pour mener à terme le projet.

9 CONCLUSION

Finalement, après plusieurs itérations de code et sessions de déverminages, les spécifications visées pour le prototype ont pu être atteintes. C'est-à-dire :

- Afficher les notes jouées en mode asynchrone ;
- Afficher les accords joués en mode asynchrone, et
- afficher des partitions en temps réel selon un tempo prédéfini.

Sommes toutes, les objectifs initiaux sont restés très semblable au développement qui a été fait durant toute la session.

10 BIBLIOGRAPHIE

[1] [En ligne]. Available: https://aurga.wordpress.com/2012/08/11/methode-des-courbes-en-s-sigmoide-ou-gestion-de-la-valeur-acquise-earned-value-management/.

[2] [En ligne]. Available: http://blog.timeperformance.com/les-secrets-du-diagramme-de-la-valeur-acquise-en-4-images/.

[3] [En ligne]. Available: www.gel.usherbrooke.ca/s5elec/h17.