UNIVERSITÉ DE SHERBROOKE

Faculté de génie

Département de génie électrique et génie informatique

Rapport Final

S5 Projet

Conception d’un système embarqué

GEN500

Présenté à

Jean-Baptiste Michaud

Présenté par

Philippe Bourassa – boup2426

Michel Tulane – tulm2101

Frédéric Fafard – faff2302

François Brunet – bruf1902

Frédéric Perron – perf2007

Nicolas Cloutier – clon1502

Samuel Ouellette – oues3008

Stephane Lajoie – lajs2004

David Gaudreault – gaud1910

Sherbrooke – 18 avril 2017

# Table des matières

[Table des matières I](#_Toc480447777)

[1 Mise en contexte 1](#_Toc480447778)

[1.1 Mise en valeur de la personnalisation 1](#_Toc480447779)

[1.2 Description des fonctions personnalisées 1](#_Toc480447780)

[2 Gestion 2](#_Toc480447781)

[2.1 Planification temporelle et courbe en S 2](#_Toc480447782)

[2.2 Compte-rendu Traçabilité 3](#_Toc480447783)

[2.3 Gestion des risques 4](#_Toc480447784)

[2.4 Autres outils de gestion 6](#_Toc480447785)

[2.5 Rétrospection et conclusions 7](#_Toc480447786)

[3 Assurance qualité 8](#_Toc480447787)

[3.1 Mesure des objectifs 8](#_Toc480447788)

[3.2 Résumé de la planification 8](#_Toc480447789)

[3.3 Résumé de la mise en œuvre 9](#_Toc480447790)

[3.4 Résultats de l'exécution 11](#_Toc480447791)

[3.5 Rétrospection et conclusions 11](#_Toc480447792)

[4 Aspects techniques 12](#_Toc480447793)

[4.1 Architecture haut niveau et schéma du prototype 12](#_Toc480447794)

[4.2 Cahier des charges 13](#_Toc480447795)

[4.3 Processus de conception 13](#_Toc480447796)

[4.3.1 Tests préliminaires 13](#_Toc480447797)

[4.3.2 Conception du projet. 14](#_Toc480447798)

[4.4 Résumé de la réalisation 14](#_Toc480447799)

[4.5 Structure des interactions logicielles/matérielles 15](#_Toc480447800)

[4.6 Architecture UML des programmes 15](#_Toc480447801)

[4.7 Utilisation adéquate d'interruptions 15](#_Toc480447802)

[4.8 Architecture du traitement de signal 16](#_Toc480447803)

[5 Conclusion 18](#_Toc480447804)

# Mise en contexte

## Mise en valeur de la personnalisation

Les applications possibles avec l’utilisation des ondes sonores sont nombreuses et variées. Avec le bon matériel, on peut même utiliser le son en imagerie médicale, ou pour de la géolocalisation par les animaux, par exemple. Notre produit se sert, comme ces autres applications, de la propagation du son. Cependant, il se démarque par la transmission moins conventionnelle des ondes dans un solide rigide et très élastique. Grâce à ces propriétés, le projet permet de transformer ce type d’objet en une interface de contrôle efficace, discret et très peu coûteux. Pour atteindre ces résultats, on envoie un son inaudible par l’humain à travers l’objet et on le mesure périodiquement à l’aide d’une paire piézoélectrique collée sur la surface. À l’aide d’une télécommande, on enregistre différentes réponses fréquentielles correspondant à des déformations mécaniques appliquées par un humain. On compare ensuite les nouvelles réponses fréquentielles mesurées avec ces préalablement enregistrées pour déterminer si une même déformation est appliquée. Le résultat de ces corrélations peut être utilisé pour exécuter n'importe quelle action comme envoyer un message ou activer d’autres circuits électroniques. Dans le cadre de notre prototype, les corrélations activent des relais qui alimentent un ventilateur, une pompe à eau, une ampoule et un haut-parleur.

## Description des fonctions personnalisées

Comme il vient d’être expliqué un peu plus haut, notre prototype active des relais qui alimentent un ventilateur, une pompe à eau, une ampoule ainsi qu’un haut-parleur. Puisque le but de notre prototype était d’utiliser un nain de jardin en céramique comme objet, nous avons choisi 4 fonctions utiles à l'entretien d’une fleur, mais nous aurions aussi pu utiliser ces relais pour une multitude d’autres fonctions plus variées les unes que les autres. La plus grande limitation est la forme de l’objet dans lequel, le signal est transmis. Avec notre prototype, où l’objet a finalement été une saucière à cause de limitation de budget, nous avons choisi d’implémenter 4 fonctions, mais avec un objet de forme plus optimale, nous aurions pu facilement en implémenter beaucoup plus puisque la télécommande et le logiciel traitant le signal le permettent. Il faut simplement être capable de générer autant de déformations différentes sur ce même objet que de fonctions voulues.

# Gestion

## Planification temporelle et courbe en S

Au niveau de la planification temporelle, nous avons décidé d’utiliser le site web de gestion gratuit « Trello »[[1]](#footnote-1). Avec ce dernier, nous avions différentes listes de tâches à faire ainsi que les listes des tâches terminées associées. Il y avait, entre autres, les listes de tâches techniques, tâches livrables, tâches autres, etc. Ensuite, avec l’extension chrome « Plus for Trello »[[2]](#footnote-2), nous pouvions entrer les heures travaillées dans les tâches faites au préalable et créer des rapports qui étaient ensuite traités par un membre de l’équipe qui avait fait un document Excel pour créer et gérer la courbe en S. Ce document gérait donc le temps budgété, les acquis ainsi que le coût réel en temps. Finalement, nous utilisions aussi l’extension Chrome « *Elegantt* »[[3]](#footnote-3) pour créer un gant qui était synchronisé avec les tâches Trello au niveau des échéances.

Donc, au final, tout était au même endroit pour les membres de l’équipe et la courbe en S était mise à jour une fois par semaine par le Directeur Trello. Des screenshots de notre organisation ainsi que le document Excel est dans le répertoire « 2.1 Planification Temporelle » du dossier de remise.

La figure qui suit est la courbe en S finale obtenue au cours du projet. Il avait été prévu qu’un travail plutôt constant serait fait durant la session, avec une pause durant la semaine de relâche et un sprint en fin de session. On voit que, jusqu’à la semaine 9, nous avons assez bien suivi ce qui était prévu. Ces tâches correspondantes étaient surtout des tâches de gestions pour le début du projet. Ensuite, il a été plus difficile de suivre ce qui avait été prévu dû à cause de la charge de travail des APPs et dû au fait que nous avons appris la matière nécessaire au projet plus tard que ce qu’on pensait. On peut aussi voir entre les semaines 13 et 14 que beaucoup de temps a été mis, mais qu’il n’y a presque pas eu d’acquis : ceci du temps de débogage. Au final, les prédictions ont été assez bonnes.

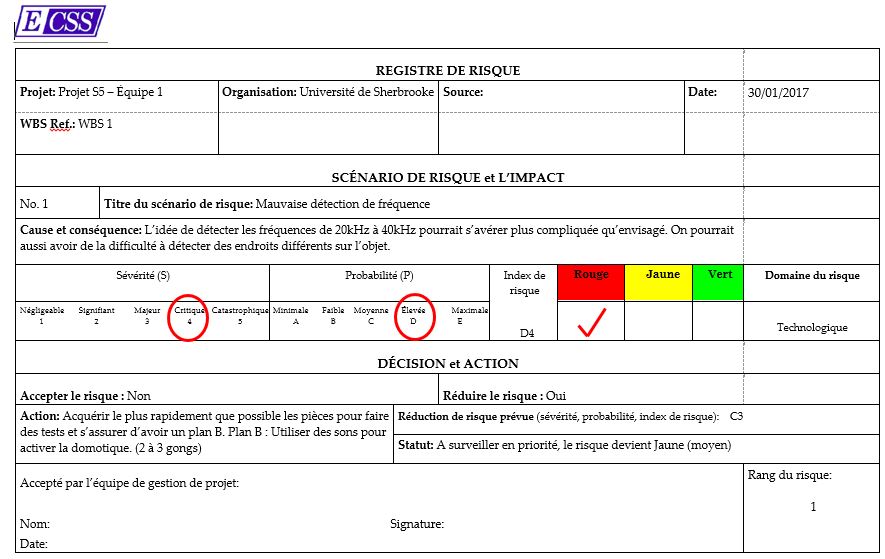
|  |
| --- |
|  |

## Compte-rendu Traçabilité

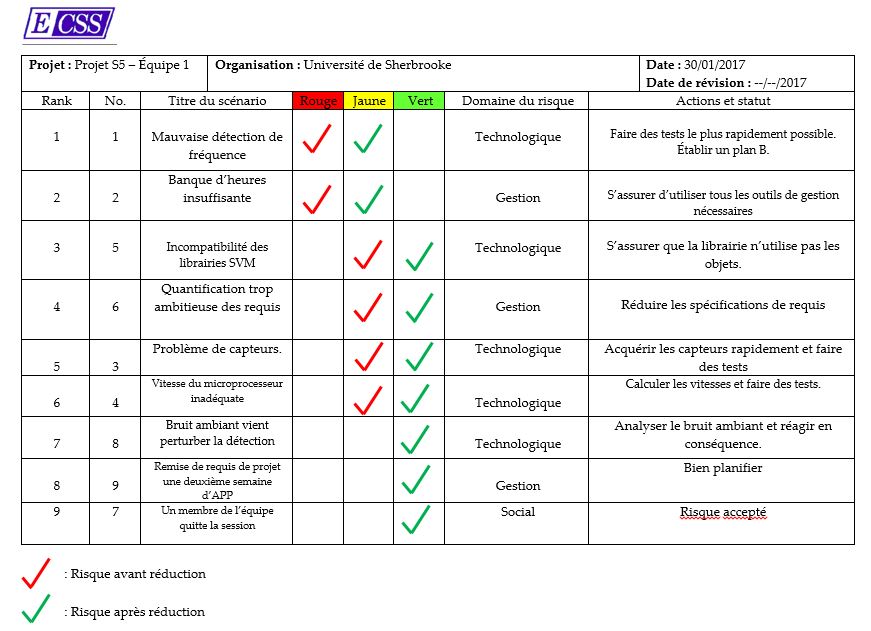
La traçabilité a été assurée à la fois par « Trello »1 et « GitHub »[[4]](#footnote-4). Premièrement, chacune des tâches dans trello était attribuée à une ou plusieurs personnes. Cet outil nous a donc permis de garder un historique du nombre d’heures passées sur chacune des tâches par chacun des membres et à quel moment. On pouvait aussi regarder le développement du projet en regardant la courbe en S qui était construite à l’aide des extensions de Trello parlé plus tôt. De plus, puisque nous utilisons « GitHub »4 pour gérer la documentation, il est possible de garder une trace des différents fichiers mis à jour par les différents membres de l’équipe, les modifications apportées, et à quel moment cette mise à jour a été effectuée. Il y avait aussi tout ce qui a été construit au niveau de l’Assurance qualité qui a permis un beau suivi des tests fait et des résultats obtenus. On pouvait donc s’assurer de ne pas faire aucun travail en double et d’avancer à un bon rythme. Bref, les outils de gestion utilisés ont permis une excellente traçabilité.

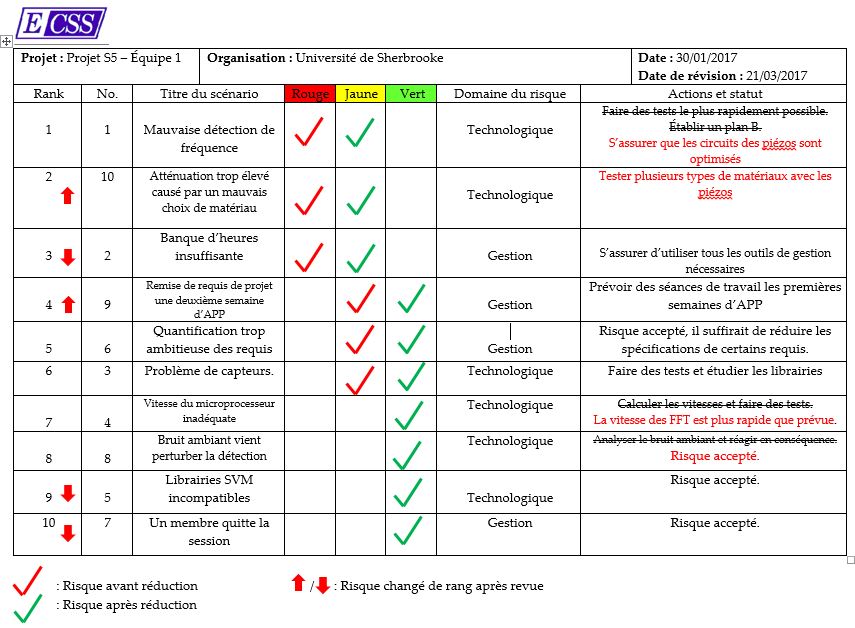
## Gestion des risques

La gestion des risques a été adoptée avec la méthode de l’ECSS étudiée lors de la S4. L’équipe a donc établi une réunion pour discuter des différents risques que contient le projet. En utilisant la documentation ECSS, on obtient des fiches descriptives complètes pour chacun des risques établis. On obtient de l’information sur la sévérité et la probabilité du risque, son impact, si ledit risque est accepté et quelle action a été mise en place pour s’assurer de réduire l’impact et/ou la probabilité du risque. Voici un exemple d’une fiche descriptive de risques selon les standards de l’ECSS:



Dans l’exemple ci-dessus, on voit que le risque est grave (rouge), c’est pourquoi une grande partie des efforts de réduction de risques a été mise sur ce risque en particulier. On a donc établi un plan B au cas où ce risque venait bloquer la fonctionnalité principale du projet, qui serait de détecter des bruits au lieu de détecter une différence de réfraction de fréquences au travers d’un objet. L’effort mis pour que ce risque soit réduit au maximum a été très utile, puisque le projet possède une bonne détection de fréquence. On voit sur la figure suivante la liste des risques établie lors du début du projet:

On obtient alors une ligne directrice qui aide à visualiser où la majorité des efforts devra être placée. Évidemment, tout au long du projet, on a dû mettre à jour cette liste. On voit donc clairement comment la gestion des risques a été modifiée sur la figure suivante :



Après plusieurs tests, on a pu vérifier que l’incompatibilité des librairies SVM qui créait de l’inquiétude ne posait plus vraiment de problèmes. Ensuite, rendu à la fin de session, le risque qu’un membre quitte la session a énormément diminué. La remise de requis de projet en deuxième semaine d’APP à ensuite augmenté en risque, puisque toutes les remises lors de la session ont été la deuxième semaine d’APP. De plus, avec le travail demandé par chaque APP, la banque d’heures insuffisante pour mener le projet à terme a augmenté, puisque le temps était compté.

## Autres outils de gestion

* Nous avions un groupe « Facebook »[[5]](#footnote-5) pour aider à la communication dans l’équipe.
* Pour la gestion de la documentation, nous utilisions « GitHub »4. Cela permettait de donner accès facilement à la documentation à toute l’équipe et d’avoir plusieurs sauvegardes.
* Pour l’écriture de ce présent rapport, « Google Docx »[[6]](#footnote-6) a été utilisé pour faciliter le travail simultané des membres de l’équipe.
* Les présentations étaient construites avec « Keynote »[[7]](#footnote-7) sur iCloud. Ceci permettait le travail simultané des membres de l’équipe.

## Rétrospection et conclusions

Au cours de ce projet, nous avons développé de bonnes techniques de gestion et ça a paru tout au long de la session. Les tâches étaient claires, la gestion du temps a été facile, les tests avec l’assurance qualité étaient clairs. Les principaux points qu’on devra améliorer dans le futur sont la mise en marche de réunions hebdomadaires de suivi, la mise en marche d’un système standardisé pour ce qui est de la gestion de la documentation et le partage des façons de faire. Tous ces points auraient permis une meilleure communication et une meilleure collaboration entre les membres de l’équipe.

Premièrement, la mise en marche de réunions hebdomadaires n’aurait pas grugé beaucoup de temps pour chacun des membres et ceci aurait permis de s’assurer du travail effectué de chacun lors de la dernière semaine et du travail qui sera effectué dans la semaine en cours. On aurait premièrement eu une idée plus claire du travail de chacun, ce qui aurait permis de mieux attribuer les tâches et ce qui aurait aussi permis de faire une meilleure évaluation des pairs à la fin du projet. En effet, il très difficile de donner des notes aux autres puisqu’on ne travaille directement qu’avec une fraction de l’équipe. Finalement, ces réunions auraient pu régler les deux problèmes qui suivront dans cet rétrospection.

Deuxièmement, on aurait dû se créer un standard d’archivage de nos documents sur le « GitHub » puisque tout était sauvegardé sur des noms, oui intuitifs, mais sans référence ou nomenclature logique. Des dossiers étaient créés ici et là et, dans un projet de plus grande ampleur, cela aurait pu créer des problèmes tels que des pertes d’informations. Ceci aurait pu être abordé justement dans une réunion hebdomadaire.

Finalement, il arrivait souvent au gestionnaire Trello de se faire poser des questions sur les livrables, les tâches, les revues ou les démonstrations alors que toute l’information était présente à des endroits précis. En effet, toutes les listes et les cartes Trellos avaient des descriptions et des listes de contrôle où cette information-là se trouvait. L’erreur a été de faire un standard de gestion qui a été mal transmis aux autres membres de l’équipe. Il aurait donc été souhaitable de faire une réunion expliquant le fonctionnement de tout ça.

# Assurance qualité

L’objectif premier de l’assurance qualité était de respecter le cahier de charge fonctionnel. L’assurance qualité englobait tous les aspects du projet, que ce soit du point de vue de la gestion des risques, des plans de tests, de la gestion et de la rédaction de code. Une personne responsable de l’assurance qualité avait comme tâche de superviser chaque étape du processus de développement du projet. Cette personne avait comme tâche de faire des suivis avec les développeurs pour s’assurer que les plans de tests soient faits au fur et à mesure qu’une partie du cahier des charges est terminée.

## Mesure des objectifs

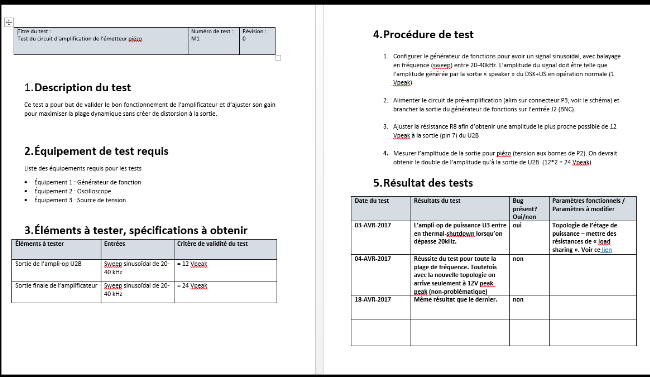
La façon dont chacun des requis a été quantifié était très arbitraire au début de la conception. Au fil du développement du projet, les objectifs sont devenus de plus en plus clairs et une mesure plus précise a pris place. On visait donc une précision de détection d’environ 80%, ce qui donne une réussite de 4 détections sur 5, ce qui était satisfaisant. Évidemment, il y a eu des changements vers la fin, mais la quantification faite restait tout de même fidèle à ce à quoi on s’attendait. On désirait obtenir une détection précise, mais pas trop, donc d’environ 1-2 cm aux environs. C’est-à-dire que si on appuie à un endroit pour une action et qu’on l’enregistre, on s’attend à ce que la détection se fasse si on appuie après à 1 cm de l’endroit précis préenregistré.

## Résumé de la planification

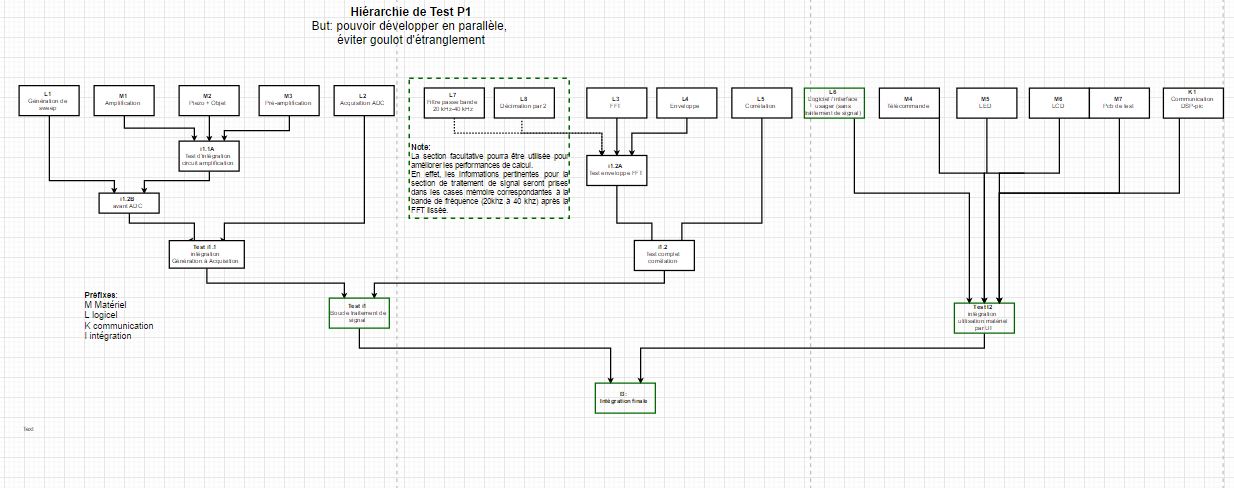
La planification tout le long du projet était relativement simple. On assignait des membres à des tâches sur Trello et chaque personne savait donc quoi faire et quand le faire. Avec l’ajout des risques et des plans de tests, on avait une bonne idée des possibles problèmes qui auraient pu survenir et du temps à réserver pour exécuter les tests. Une personne gérait le Trello et une personne gérait l’assurance qualité, de cette façon, on s’assurait que rien ne soit oublié.

## Résumé de la mise en œuvre

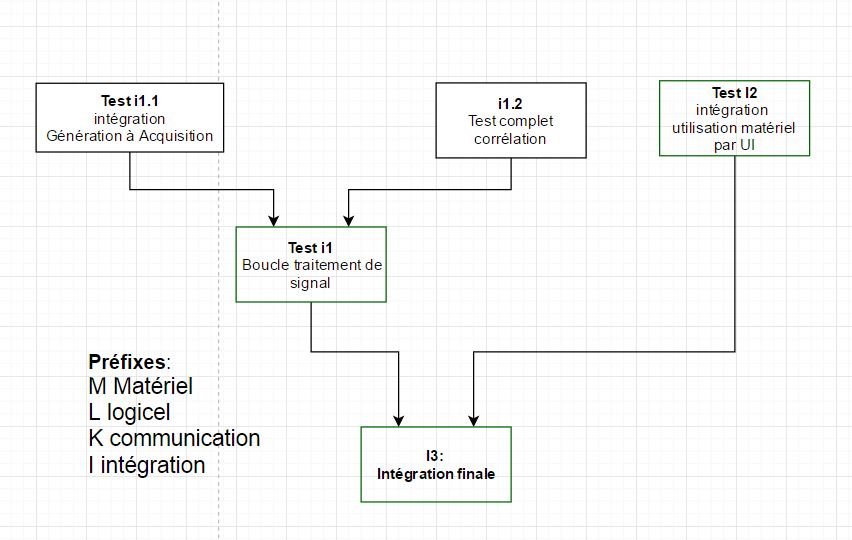
Le plan de test a été implémenté de façon hiérarchique, de façon à minimiser les goulots d’étranglement. Au début (au dessus), des tests unitaires peuvent être effectués indépendamment sur chaque section, puis il est possible de faire des tests d’intégration au fur et à mesure du développement. Un rapport doit être rempli pour chaque test. (à voir dans le dépôt) Les sections incluent les points suivants : Description, équipements requis, éléments à tester, procédure et résultats incluant les bugs à corriger.



Chaque test unitaire possède un préfixe selon sa catégorie (matériel, logiciel, communication) et un numéro pour être facilement retrouvable. La légende des préfixes est présentée dans la hiérarchie de test. Voici donc cette hiérarchie:



Voici une vue rapprochée des tests d’intégration et finale, qui comprend les tests d'intégration de la génération du sweep fréquentiel, des tests de corrélation ainsi que des tests sur le pic.



Une fois les tests d'intégration terminés, le test d’intégration final peut être fait sur le prototype. Il inclut un bon nombre de tests pour valider le bon fonctionnement de l’application en tout point.

## Résultats de l'exécution

L’exécution des tests a permis de valider l’ensemble du projet. La grande majorité des tests ont été réussis. En fait le fonctionnement est testé de manière non formelle tout au long du développement. C’est pourquoi lorsque la fonction est terminée et testée de manière formelle, il ne reste que rarement des problèmes. De tous les tests effectués seulement trois ont été échoués. L’un d’eux a été accepté puisque le résultat n’avait aucun impact sur les fonctionnalités finales. Les autres ont été corrigés, puis les tests ont été réalisés à nouveau et réussis.

## Rétrospection et conclusions

Finalement, toute l’équipe a pu comprendre pourquoi, sur le marché du travail, les compagnies qui ont un groupe d’assurance qualité dominent leur part du marché par rapport aux compagnies qui n’en possèdent pas, respectivement à leur part du marché. En effet, en testant et en mettant en doute constamment chaque partie implémentée lors du développement du projet, on s’est aperçu que l’investissement en temps en valait la chandelle. Il est impossible de s’assurer qu’un algorithme est sans faille si on ne fait pas tout en notre possible pour trouver une façon de trouver une faille. Dès que l’équipe avait un doute sur une façon dont le projet pouvait tourner mal, le responsable de l’assurance qualité faisait tout en son pouvoir pour prouver que ladite façon était vraie ou non, ce qui rendait chacune de nos fonctions plus robustes.

En conclusion, monter et gérer une assurance qualité sur des projets de moyen à long terme est quelque chose de très valorisant pour le projet ou le produit. Il se peut, toutefois, que l’assurance qualité soit une perte de valeur pour des petits projets à budget moindre. Par contre, en ingénierie, il est très probable que l’on ait à participer à des projets de grande envergure, et c’est à ce moment que l’on voit l’utilité, voir l’indispensabilité d’une assurance qualité pour qu’un projet soit rentable à son maximum.

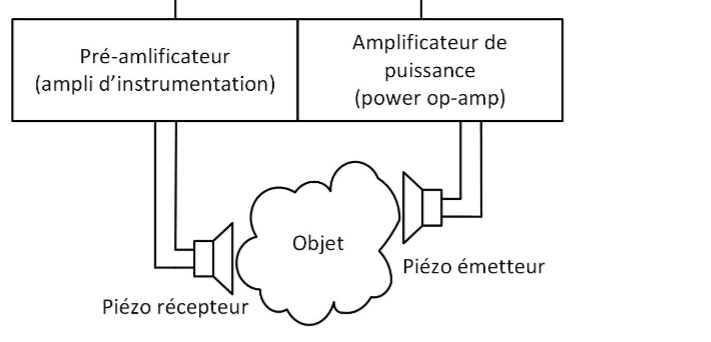
# Aspects techniques

## Architecture haut niveau et schéma du prototype

**Schéma de branchements**

L’architecture matérielle du projet a été conçue au fil de la conception selon nos besoins et les requis du prototype. Au départ, certains éléments étaient fixés dans l’architecture comme le DSK et le PIC qui nous permettaient de développer la partie logicielle et l’architecture de traitement de signal du prototype. Puis se sont ajoutés les éléments propres au fonctionnement de notre système, c’est-à-dire l’objet résonnant en céramique, l’émetteur et le capteur piézoélectrique et le circuit d’amplification et de préamplification faisant l'interface avec le DSK. Finalement, s’est ajoutée à l’architecture matérielle une carte de relais nous permettant de contrôler des éléments physiques (120V) à l’aide d’entrées sorties tout usage (GPIO) du DSK. Cette partie s’est ajouté à la toute fin lors de la préparation à la démonstration finale puisqu’elle n’était pas nécessaire pour développer tout le reste. Le schéma en question est disponible dans le répertoire *“4.1Architecture haut niveau et schéma du prototype/Architecture haut niveau”* de l’archive de dépôt (en fichier Visio .vsdx ainsi qu’en image .jpg).

**Schéma des amplificateurs**



Ce circuit, tel que mentionné plus haut, fait l’interface électrique entre le DSK+US ainsi que nos transducteurs piézoélectriques. La partie “amplificateur de puissance” se charge d’alimenter le piézo à une tension AC d’environ 10 Vpeak à partir du signal fourni par la sortie “haut-parleur” du DSK. Le signal amplifié, dans notre cas, est une fonction sinusoïdale dont la fréquence varie en “sweep” entre 20 kHz et 40 kHz. La topologie de l’étage de puissance de l’amplificateur a été revue après quelques tests, car le circuit intégré (ampli-op de puissance) n’arrivait pas à alimenter le piézo avec une seule de ses sorties (le slew rate était visible lorsqu’on augmentait la fréquence ou le gain). Au final, les deux amplificateurs contenus dans le circuit intégré (TCA0372) ont été mis en parallèle avec des résistances de partage de charge (*load sharing*) pour combiner les deux sorties, de façon à améliorer les caractéristiques (gain et slew rate disponibles) de l’étage de puissance.

La partie “préamplificateur” n’a subi que très peu de modifications par rapport à la conception de départ. En effet, un étage de gain configurable selon nos besoins (nos besoins d’amplification dépendaient du signal que nous arrivions à extraire de l’objet en résonance par le capteur piézoélectrique) avait été prévu pour ajuster la plage dynamique d’entrée du préamplificateur (voir le schéma électrique complet). Puisque la sortie du préamplificateur possédait une protection en tension pour l’ADC, nous pouvions procéder aux ajustements des gains lors de nos essais sans risques d’endommager l’ADC du DSK. Le schéma électrique complet des circuits d’amplification et de préamplification est disponible dans le répertoire *“4.1Architecture haut niveau et schéma du prototype/Schéma électrique”* de l’archive de dépôt (en format Altium .schDoc ainsi qu’en .pdf).

## Cahier des charges

Deux piliers de notre système d’assurance qualité nous permettaient de garder le cap sur la rencontre des spécifications du prototype : le cahier des charges fonctionnel et le plan de la hiérarchie des tests.

Les spécifications que devait rencontrer notre prototype quant à la fiabilité, la robustesse et la performance étaient listées dès le départ dans le cahier des charges. Chacun de ces requis que contenait le cahier des charges était validé dans un rapport de test ; l’association entre les requis et les rapports de tests se faisait dans le cahier des charges fonctionnel. Le cahier des charges se trouve dans le répertoire *“4.1 Cahier des charges”* de l’archive de dépôt (en format Excel .xlsx).

## Processus de conception

### Tests préliminaires

Puisque le projet impliquait de nombreuses incertitudes, il a fallu confirmer suivre un processus rigoureux pour valider la faisabilité tout au long de la session. En effet, au départ, nous ne savions pas à quel point le spectre fréquentiel serait modifié en touchant un objet. De plus, nous ne savions pas si le DSP permettait une vitesse de traitement suffisamment élevée pour que la détection soit faite en temps réel.

Pour éliminer les incertitudes par rapport à la faisabilité du projet, nous avons effectué différents tests préliminaires. Tout d’abord, nous avons testé la déformation du spectre fréquentielle d’un signal traversant un objet à l’aide d’un générateur de signal et d’un oscilloscope. Les résultats obtenus ont ensuite été analysés avec Matlab. Après ces essais, nous étions en mesure de confirmer que le signal serait modifié de manière à ce qu’il soit possible de détecter à quel endroit l’objet est touché.

Ensuite, la vitesse de traitement du DSP a été testée. À ce moment de la session, nos connaissances par rapport au DSP étaient plutôt limitées. Après avoir testé le temps nécessaire à l’exécution d’une fft pour la taille des tableaux désirés, nous étions en mesure de confirmer que le DSP était assez performant pour notre application.

### Conception du projet.

Le reste de la conception a été fait de manière à valider chacune des étapes à mesure qu’elles sont terminées. Tout d’abord, la génération du signal avec le DSK et l’acquisition du signal a été développée. Il a ensuite été possible d’acquérir des signaux très représentatifs de ceux qui seront acquis lorsque le projet sera complété.

Les signaux enregistrés ont ensuite été analysés à l’aide de Matlab. De cette manière il a été possible de développer l’algorithme de détection avant d’essayer de l’implémenter. Nous avons été en mesure de développer un algorithme à la fois robuste qui demande un minimum de traitement. Une fois l’algorithme terminé, il ne restait qu’à l’implémenter sur le DSK. Bien entendu, chaque partie du projet a été testée selon les plans de tests.

## Résumé de la réalisation

La réalisation du projet s’est effectuée selon la planification. C'est-à-dire que chaque partie du projet a été réalisée et testée tel que prévu. Toutes les parties du projet ont aussi été effectuées dans l’ordre, par exemple, l’implémentation du traitement de signal a été réalisée après l’implémentation de l’acquisition du signal.

Lorsque possibles, les tâches ont été divisées pour que le développement s’effectue en parallèle. Par exemple, la programmation du pic a pu être faite en même temps que le montage et que la programmation du DSP.

## Structure des interactions logicielles/matérielles

La programmation des positions sur l’objet est fait avec l’aide de la télécommande. Pour programmer une position, il suffit de toucher l’objet à l’endroit désiré puis, sans lâcher l’objet, appuyez sur la touche «1», «2», «3» ou «4» de la télécommande. Chaque touche de la télécommande est associée à une fonction différente (1=Lumière, 2=Eau 3=Ventilateur, 4=Musique). Toucher l’objet à l’endroit où une position à été programmée va basculer sa fonction (allumer/éteindre). Si une touche de programmation de la télécommande est appuyée alors que l’objet n’est pas touché ou est touché à une position déjà programmée, une erreur est générée. La touche ­­­«C» permet de réinitialiser le système. L’écran LCD affiche simplement la dernière interaction (initialisation, programmation d’une position, détection d’une position, etc.). L’écran défile les 4 dernières interactions (une par ligne).

## Architecture UML des programmes

Pour les diagrammes UML complets de l’architecture des programmes, consulter la section 4.6 du dépôt.

## Utilisation adéquate d'interruptions

Notre programme a eu besoin de deux vecteurs d’interruptions. Les deux vecteurs utilisés sont l’interruption 4 et l’interruption 11. Chacune de ces interruptions ne contenait que le strict minimum d’opération, pour que cette partie du code ne devienne pas bloquante.

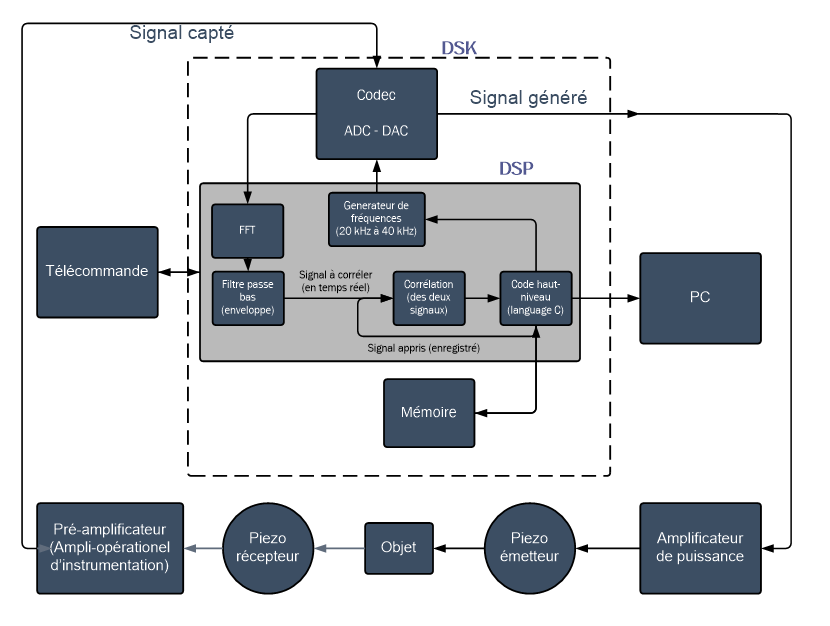
L’interruption 4 lève un *flag* signifiant que le codec a fini de convertir une valeur. La valeur n’est pas immédiatement lue, il faudra attendre que le processus soit rendu à cette étape dans le *main*. Le *flag* n’est remis à zéro qu’après la lecture de la valeur convertie, pour pouvoir attendre la prochaine valeur.

L’interruption 11, activée à une fréquence de 96 kHz, en fait un peu plus. D’abord, il lève aussi un *flag* qui servira à déclencher un processus dans le *main*, puis il envoie une valeur du tableau précalculé vers la sortie audio du DSK. Il est nécessaire de sortir cette valeur dans l’interruption pour ne pas générer des irrégularités dans la fréquence de sortie voulue. Le *flag* est remis à zéro après avoir incrémenté le compteur servant à sortir la bonne valeur dans la prochaine interruption 11.

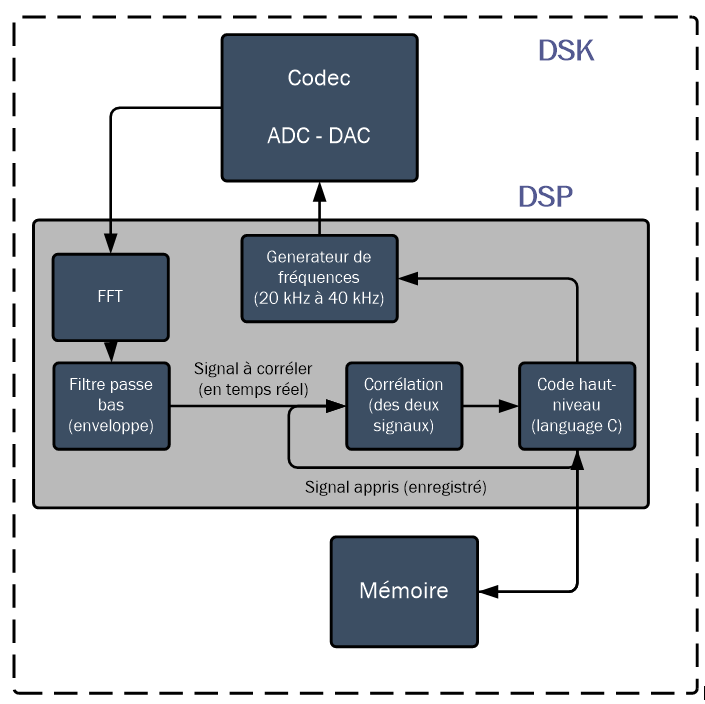
## Architecture du traitement de signal

Pour ce qui est du traitement de signal, un sweep sonore de 20 kHz à 40 kHz échantillonné à 96kHz d’une durée de 20 ms est généré à partir d’une table précalculée sur MATLAB. Ce sweep est par la suite amplifié pour être émis avec assez de puissance par un premier piézoélectrique. Un second piézoélectrique capte ensuite le signal qui est préamplifié pour aller vers l’ADC (niveau 0 à +1 V) du dsk.

Voici le schéma bloc de la boucle de traitement de signal utilisé:



Pour la suite du traitement, on utilise 4 séquences de 2048 valeurs lues. On applique une transformée de Fourier sur chacune de ces 4 trames, puis on en fait la moyenne. Cette moyenne permet d’enlever le bruit, puisque la moyenne du bruit est de 0. On applique ensuite un filtre passe-bas FIR à coefficients égaux d’ordre 4, pour décimer par 4 et utiliser moins de mémoire. C’est sur ces tableaux de 256 valeurs qu’on applique les corrélations. Lorsque la corrélation surpasse un certain seuil, une sortie correspondante du DSP est activée. Voici le schéma bloc correspondant à ce traitement:



# Conclusion

Globalement, les résultats du projet sont assez satisfaisants. Il est assez facile de différencier quelques touchers sur certains objets en céramique. C’est aussi flexible entre les utilisateurs. Les mêmes touches peuvent donc être utilisées par n’importe qui. Cependant, nous admettons que l’implémentation réalisée à des limites.

D’abord, il est difficile de trouver plus de 3 positions de toucher dont les réponses en fréquence seront assez différentes pour ne pas générer de fausses détections. Pour rendre le programme plus robuste et plus utile, il faudrait implémenter un meilleur traitement des signaux temporels reçus. Par exemple, on pourrait tirer avantage du fait que seulement quelques rares parties de la réponse en fréquence se modifient en touchant les objets. Il suffirait de créer un algorithme cherchant les plus grosses différences dans les réponses en fréquence. On pourrait, grâce à cela, faire des corrélations uniquement sur les endroits permettant de mieux différencier les signaux, plutôt que sur toute la plage complète.

De plus, notre matériel émettant les fréquences entre 20kHz et 40kHz est plus bruyant que nous l’espérions. Cela est dû au fait que nous interrompons l'émission du signal pendant le traitement des données précédemment reçues. Même en segmentant notre code au maximum pour permettre d’aller émettre le plus souvent possible, il est impossible d’être assez rapide. Certaines tâches, comme la FFT, sont trop longues. Pour régler ce problème, il serait nécessaire de réussir à générer la sortie sonore en parallèle au traitement, plutôt que séquentiellement.

1. https://trello.com/ [↑](#footnote-ref-1)
2. http://www.plusfortrello.com/p/about.html [↑](#footnote-ref-2)
3. https://elegantt.com/ [↑](#footnote-ref-3)
4. https://github.com/ [↑](#footnote-ref-4)
5. https://www.facebook.com/ [↑](#footnote-ref-5)
6. https://docs.google.com/document/u/0/?hl=fr [↑](#footnote-ref-6)
7. https://www.apple.com/ca/fr/keynote/ [↑](#footnote-ref-7)