Projet Tutoré Semestre 3
Cahier des charges

Table des matières 1/2

DURIEUX Romain – JEUDY Louis – ABBASI Elyas – BAREAU Alexandre

Table des matières 2/2

I. Rappel de la demande

Un client possède un sampler simple à piles mais ce dernier n'est pas pratique pour lui et son équipe. Il nous demande donc de réaliser un sampler portable. Notre client et son groupe musical, aimeraient piloter un système de déclencheur de samples (sons) à partir d'un AKAI MPD 218 sur une batterie avec une autonomie minimale de trois heures. Le MPD 218 ne sort que du signal midi sur 3 banques qui contiennent chacune 3 pages de 16 pads. Ils souhaitent une programmation pouvant remplir les fonctions de sampler sur la base d'un Raspberry. Il est possible de proposer un autre système hardware mais restant dans des tarifs assez bas pour pouvoir cloner des systèmes de secours.

II. Vision produit

a) Pour quoi ?

De manière générale, l'objectif principal comme défini précédemment pour ce projet est de réaliser un sampler portable. Pour réaliser celui-ci nous mettrons à profit nos connaissances, nos compétences et notre expérience emmagasinées au cours des semestres passés. Cependant, ce projet sera formateur puisqu'il traite d'un sujet méconnu de tous tant dans le matériel à utiliser que dans la conception, des choix techniques à faire ainsi que son côté professionnel.

b) Pour qui ?

Cette année, ce projet se base sur une mise en situation professionnelle et pour le réaliser nous devons répondre à une demande d'un client. Cette présence induit une obligation d'adaptation, que ce soit dans le sujet, les technologies à utiliser ou encore la communication. Pour notre projet, notre client se nomme Brieuc Bestel. Faisant parti d'une troupe musicale nommée « Jazz Combo Box », il nous propose ainsi un projet embarquant deux principaux domaines : la musique et l'électronique pour une utilisation lors de concerts dès 2021.

c) Comment ?

Afin de réaliser et de répondre correctement à la demande, plusieurs éléments vont être indispensables à rédiger, suivre et analyser.

- Livrables

Tout d'abord les phases de conception et d'analyse seront importantes et fondamentales afin de maîtriser excellemment notre sujet.

En premier lieu, la rédaction de ce cahier des charges passera par plusieurs étapes. Nous y retrouvons le rappel de la demande, le sujet choisi, la vision produit, une compréhension approfondie du sujet à travers la définition de chaque terme professionnel employé mais aussi une documentation sur le matériel imposé.

En second lieu, nous retrouverons dans ce document détaillé, le périmètre du projet (son fonctionnement global, toutes les fonctionnalités envisagées et/ou souhaitées avec leur propre scénario d'usage). A l'issue de la définition de ce périmètre, l'objectif sera de découper intelligemment les fonctionnalités en fonction de leur difficulté de réalisation. Ce découpage entrainera une première définition des niveaux d'ambition caractérisée par un bref résumé de chaque version du produit. Ainsi, nous pourrons par la suite détailler chaque version. Chacune d'elle contiendra le détail de ses fonctionnalités envisagées, des prototypes (maquette, simulation, logiciels), des scénarios et des éléments d'analyse (diagrammes). Une fois cette phase achevée, nous considérerons les choix techniques à faire pour la réalisation. Ces choix concernent chaque élément matériel, logiciel, mais aussi le coût global du projet.

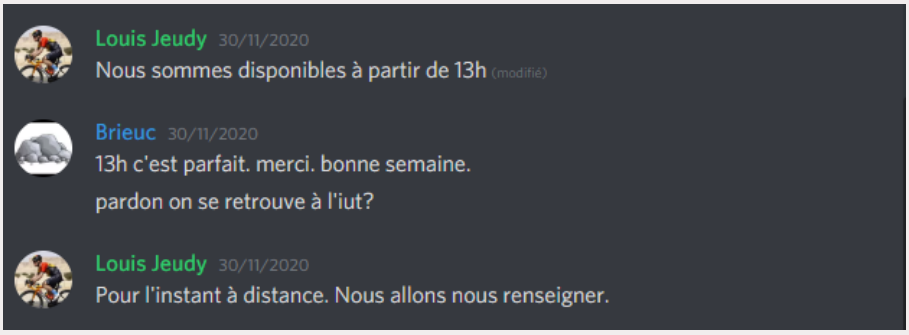
C'est ainsi que sur la base de cette analyse complète, nous entamerons la réalisation de prototypes qui seront réalisés à travers des études de faisabilités, d'explications propres à une partie d'une fonctionnalité. Cette tâche permettra de démontrer la face « professionnelle » du projet, c'est-à-dire des portions de code, d'essais…

Enfin, dès qu'un prototype (version) sera validé, il pourra être mis en production. Il sera documenté, c'est-à-dire que ses éléments de conception et d'utilisation seront écrits.

- Outils de pilotage et de suivi de projet

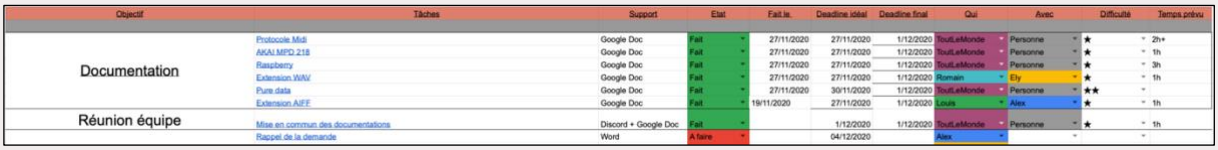
Afin que l'équipe puisse travailler avec une cohésion maximale nous avons mis en place plusieurs moyens technologiques concernant la communication, le suivi du projet et la gestion du code.

En ce qui concerne la communication, un serveur Discord a été mis en place. Chaque personne peut exposer ses idées, ses interrogations et suggestions. Il s'agit aussi d'un espace pour organiser nos rendez-vous vocaux. De plus, notre client à un espace sur ce serveur pour interagir avec nous. Ceci facilite la communication avec ce dernier si nous avons des questions à lui poser.

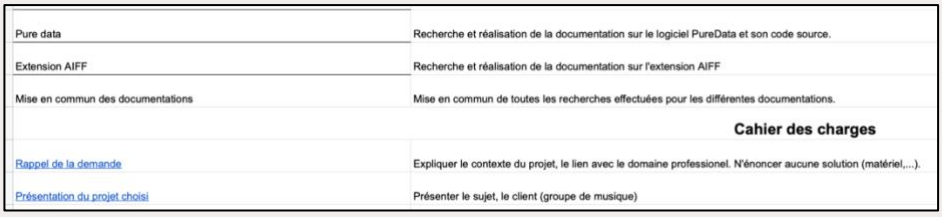


La gestion de tous les documents de synthèse, de prise de notes, sont stockés sur la plateforme collaborative Google Drive. Elle nous permet de réaliser chaque document de façon simultanée et direct.

Pour le côté gestion et planification du projet, c'est-à-dire l'organisation, la répartition des tâches nous tenons à jour un tableau de type « Gantt ». Chaque tâche est inscrite. Nous retrouvons pour une tâche, la personne travaillant dessus, la date de livraison à respecter, le temps requis à l'étudier, sa difficulté ainsi que son état d'avancement : « A faire », « En cours », « Fait ». A chaque tâche, une description détaillée de celle-ci est disponible dans un second tableau. En voici un extrait :



Objectif	Tâches	Support	Etat	Fait le	Deadline idéal	Deadline final	Qui	Avec	Difficulté	Temps prévu
Documentation	Protocole Midi	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	TousLeMonde	Personne	★	2h+
	AKAI MPD 218	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	TousLeMonde	Personne	★	1h
	Raspberry	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	TousLeMonde	Personne	★	3h
	Extension WAV	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	Romain	Ely	★	1h
	Pure data	Google Doc	Fait	27/11/2020	30/11/2020	1/12/2020	TousLeMonde	Personne	★★	
	Extension AIFF	Google Doc	Fait	19/11/2020	27/11/2020	1/12/2020	Louis	Alex	★	1h
Réunion équipe	Mise en commun des documentations	Discord + Google Doc	Fait		1/12/2020	1/12/2020	TousLeMonde	Personne	★	1h
	Rappel de la demande	Word	A faire		04/12/2020		Alex			



Pure data	Recherche et réalisation de la documentation sur le logiciel PureData et son code source.
Extension AIFF	Recherche et réalisation de la documentation sur l'extension AIFF
Mise en commun des documentations	Mise en commun de toutes les recherches effectuées pour les différentes documentations.
	Cahier des charges
Rappel de la demande	Expliquer le contexte du projet, le lien avec le domaine professionel. N'énoncer aucune solution (matériel,...).
Présentation du projet choisi	Présenter le sujet, le client (groupe de musique)

Enfin, l'IUT met à disposition une plateforme de versionning de codes sources. Elle se nomme « forge IUT La Rochelle ». Elle nous sera utile lors du développement du produit afin de programmer de façon collaborative et organisée.

- Outils de développement

En ce qui concerne les outils de développement, nous en avons déjà choisi certains au vu de notre expérience mais aussi grâce aux technologies indiquées dans le sujet.

Pour concevoir les différents diagrammes et autres éléments de conception nous utiliserons l'AGL Visual Paradigm.

Ensuite, comme indiqué dans le sujet de ce projet, nous développerons et travaillerons sur un nano ordinateur de la marque Raspberry. Celui-ci sera accompagné d'un contrôleur nommé Akai MPD 218.

En ce qui concerne les logiciels et les langages de programmation, nous pourrons les définir une fois que l'étude des choix techniques sera réalisée.

III. Prise de connaissances des éléments technologiques

Dès à présent, considérons une prise de connaissances à propos des technologies proposées par le client et des éléments connexes à celles-ci.

Définissons une documentation informative pour les éléments suivants : sampler, Raspberry, AKAI MPD 218, protocole MIDI, les extensions WAV, AIFF et le logiciel Pure Data.

• Principe d'un sampler

Pour rappel un sampler est une "boîte à musique" avec un clavier spécial et un ordinateur pour le traitement des sons. Lorsque l'on tape sur différentes touches de cette boîte, celle-ci va émettre des sons et l'utilisateur pourra en faire une musique.

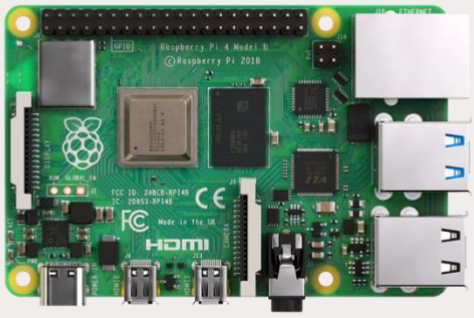
• Qu'est-ce qu'un Raspberry ?

Le Raspberry est un microcontrôleur qui s'apparente à un vrai ordinateur que l'on utilise tous les jours. Celui-ci est moins puissant mais est plus simple à transporter du fait qu'il est très petit, de l'ordre de 10x8cm². Ce microcontrôleur est programmable et nous pouvons y brancher divers composants afin de les programmer et les faire fonctionner de manière personnalisée.

Le Raspberry à l'image d'un ordinateur lambda possède une mémoire vive ainsi qu'une mémoire en dur caractérisée par une carte SD. Il comporte aussi un processeur, une carte Ethernet ainsi qu'une sortie audio Jack.

Les langages de programmation à adopter pour programmer ce genre de machine sont le Python, le C, C++…

Ce micro-ordinateur fonctionne sur le système d'exploitation Raspbian issue d'une distribution Linux.



- Qu’est-ce qu’un AKAI MPD 218 ?

Le AKAI MPD 218 est un clavier (contrôleur musical) qui fonctionne comme celui d’un ordinateur (mise à part que les touches sont différentes). Nous trouvons sur celui-ci des potentiomètres (résistances qui varient en les tournant), des pads (touches). Ce sampler en détient 48 réparties équitablement sur 3 banques et 18 potentiomètres répartis aussi sur 3 autres banques. Une banque correspond donc à une configuration de 16 pads. Les banques s’apparentent à des emplacements de stockage. Un fichier son est relié à un pad. Ainsi, pour un unique pad, 3 fichiers son lui peuvent être liés et sélectionnables selon le pad. Chaque pad détient une vélocité allant de 0 à 127. La vélocité est la pression appliquée sur un pad. D'ailleurs il existe une touche afin d’activer une vélocité maximale à chaque appui d’un pad. Ce sampler est le plus souvent utilisé pour la musique.

L’alimentation se fait par USB et la transmission des données quant à elle s’opère par le protocole MIDI.

Voici une signification de chaque bouton présent sur le contrôleur :

- Pads : Ces pads permettent de déclencher des sonorités de batterie ou d’autres échantillons du logiciel ou du module de son MIDI externe. Les pads sont sensibles à dynamique et à la pression et donc très réactifs et intuitifs.

- Pads Bank : Cette touche permet de sélectionner l’une des trois banques indépendantes de pads. Cela vous donne accès jusqu’à 48 pads différents (16 pads de 3 banques de pads différentes).

- Full Level : Lorsque le mode Full Level est activé, les pads jouent toujours à la vélocité maximale (127), peu importe la force à laquelle ils sont frappés.

- Sélecteur de programme (Prog Select) : Maintenez cette touche enfoncée tout en appuyant sur un des pads pour sélectionner le programme avec le même numéro que le pad. Un programme est un mappage des pads, qui peut être utile lors de situations particulières comme lorsque vous utilisez un ensemble de batterie General MIDI ou une échelle mélodique spécifique. (Si cette touche est maintenue enfoncée, aucun pad ne déclenchera de sons).

- Qu’est-ce que le protocole MIDI ?

Musical Instrument Digital Interface (MIDI) est un protocole de communication entre les instruments (électroniques, contrôleurs, séquenceurs) et les ordinateurs. Ce protocole a pour objectif de réaliser une uniformisation entre la partie software et hardware (langage commun). Ce protocole lorsqu’il est utilisé, donne naissance à un fichier d’extension .midi. Toutes les notes du solfège sont disponibles (tablature de guitare, batterie).

Dans un premier temps, détaillons les caractéristiques techniques de l’interface MIDI. Ce protocole ne transmet pas de signal audio mais des messages de commandes (appelés « signal MIDI »). Un message contient un identifiant de canal. Il existe ainsi plusieurs types de messages. Le premier se nomme « Status » afin de qualifier la nature des informations (par exemple, indique le type de message envoyé par le système électronique). Un second message peut être de type « Data » servant à quantifier les données avec des numéros, des noms de notes ou encore des valeurs de vélocités (valeur systématiquement envoyé). Le message de type « canaux » quant à lui permet de contrôler plusieurs instruments en même temps (16 canaux séparés, jouer en même temps sans que les flux de données soient mélangés). Enfin le message « Système » permet de gérer les messages en temps réel (synchronisation). Il fait abstraction des canaux MIDI pour se concentrer sur l’ensemble des machines/instruments connectés. Ce protocole peut circuler de deux manières différentes selon deux types de connexions. Une connexion appelée DIN à travers 5 broches ou une connexion USB de type A que l’on rencontre sur la plupart des ordinateurs. Ce dernier mode offre un échange bidirectionnel (envoi/réception MIDI). Pour accepter ce protocole, il faut impérativement que les machines comportent le Standard General MIDI (SGM). Cette norme permet d’associer une banque ou un type de son à un numéro unique et identique pour chaque machine. Une machine possède le SGM si et seulement si elle supporte :

- La Polyphonie (machine peut jouer plusieurs notes simultanément (16 notes minimum)
- La Multitimbralité (permet de jouer plusieurs sons différents en même temps (15 son minimum))
- Posséder des familles d’instruments en commun
- Le canal MIDI n°10 doit être consacré au kit de percussion)
- Paramètres compatibles (Instruments compatibles avec les messages de type modulation, volume, balance).

Ce protocole nous permet de garder la main sur chaque réglage que nous voulons fournir.

Dès à présent étudions le fichier MIDI. Celui-ci est conçu pour contenir les sauvegardes d’informations qui pourront être relues ultérieurement. Il s’agit donc d’un fichier qui enregistre les informations de numérotation de sons, de notes enregistrées et les réglages des paramètres (la panoramique, le volume, le vibrato, la modulation). Il est ainsi compatible avec tout logiciel de lecture appelé « Player ». Nous distinguons trois formats de ce fichier. Le format « 0 » ne contient qu’une une seule piste d’instrument (peut prendre en compte des messages jusqu’à 16 canaux MIDI). Le format « 1 » peut contenir une ou plusieurs pistes d’instruments.
Enfin le format « 3 » contient plusieurs séquences (issues de pattern).

Ce fichier est composé et structuré d'une façon extrêmement précise. Un premier bloc correspond au « header » c'est-à-dire à l'entête du fichier, donnant des informations générales sur son organisation : longueur, nombre de pistes, valeurs de synchronisation. Le deuxième bloc se nommant « Track chunk » est présent pour préciser le temps écoulé de l'évènement et les messages des canaux MIDI. Et enfin le troisième bloc « Variable length values » permet de gérer les octets que peut contenir une valeur. Ce fichier a donc un avantage d'être modifiable à souhait au niveau des notes, des paramètres… Par exemple il est possible d'accélérer le tempo sans engendrer de destruction sonore. Ceci est possible car seule la gestion des messages est réalisée, au contraire de l'audio.

- ## Que sont les extensions WAV et AIFF ?

Durant notre rendez-vous avec le client, ce dernier nous a indiqué sur son utilisation de fichiers sons ciblés. En voici une explication :

Les extensions de fichiers .aiff et .wav sont les extensions de fichier sonore que nous allons utiliser pour lire des différents sons. La différence entre les deux formats est que l'extension .aiff ou .aif est utilisée pour les ordinateurs MAC. Les fichiers wav ou wave peuvent être lus par tous les systèmes d'exploitation. L'avantage des extensions aiff et wav est que le son sera de meilleure qualité sans aucune perte de données mais sera plus volumineux car il ne sera pas compressé.

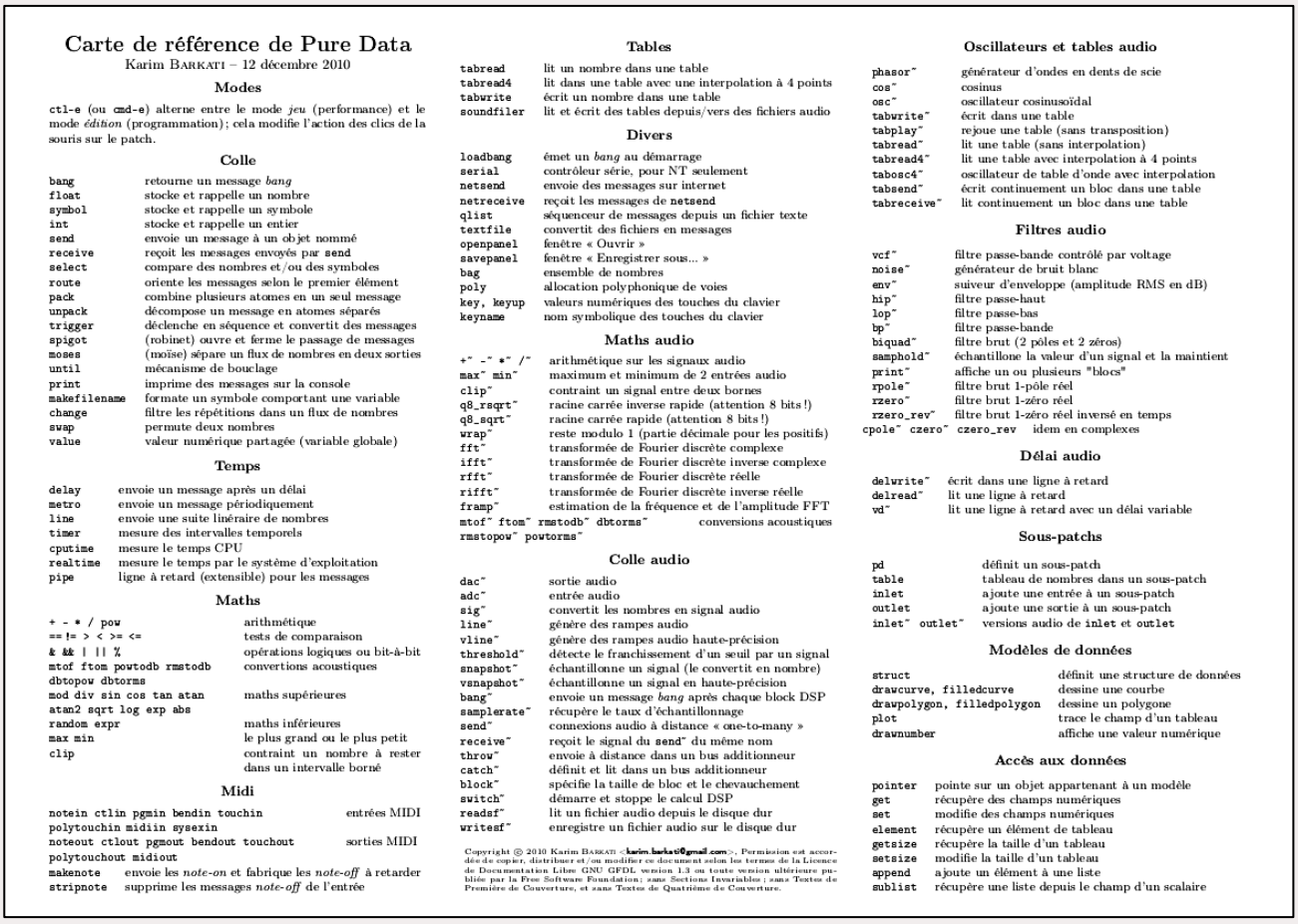
- ## Qu'est-ce que Pure Data ?

Tout d'abord Pure Data est le logiciel utilisé par notre client afin de configurer son clavier. A travers celui-ci, il assigne des sons à chaque pad.

Il s'agit d'un logiciel de programmation graphique orienté pour la création musicale. Grâce à celui-ci la modélisation d'instruments électroniques est possible (synthétiseur, contrôleur…). Il peut aussi être appliqué à d'autres domaines comme l'électronique, les sciences, les mathématiques. Il permet donc d'interfacer des capteurs (caméra, capteurs de présence) et d'autres systèmes afin de commander des robots, interagir avec des sites internet, visualiser des données.

PureData est un logiciel open source et disponible sur les OS Linux, Windows et macOS. Pour le moment, seul la version anglaise du logiciel existe. Il permet d'inclure de nombreuses librairies réalisées par des utilisateurs développeurs. De cette façon, le logiciel fait grandir sa capacité d'actions. Le code source étant disponible sur la plateforme GitHub (https://github.com/pure-data/pure-data) est proposé en langage C. De même pour ses librairies, elles sont écrites dans le même langage mais nous pouvons en trouver certaines développées en langage C++.

Du côté technique, dans cette programmation graphique, l'élément de base (objet) est une boîte rectangulaire. Un programme est appelé « patch » et il constitue un ensemble de boîtes connectées par des fils. Il s'agit d'un environnement de travail en temps réel rendant visible tout changement provoqué en cours de programmation. Il va permettre de gérer les signaux entrant dans l'ordinateur mais aussi ceux sortant de ce dernier. Chaque objet présent dans PureData représente une fonction. En voici une liste :



Carte de référence de Pure Data

Karim Barkati – 12 décembre 2010

Modes

`ctl-e` (ou `cmd-e`) alterne entre le mode *jeu* (performance) et le mode *édition* (programmation) ; cela modifie l'action des clics de la souris sur le patch.

Colle

`bang`	retourne un message *bang*
`float`	stocke et rappelle un nombre
`symbol`	stocke et rappelle un symbole
`int`	stocke et rappelle un entier
`send`	envoie un message à un objet nommé
`receive`	reçoit les messages envoyés par `send`
`select`	compare des nombres et/ou des symboles
`route`	oriente les messages selon le premier élément
`pack`	combine plusieurs atomes en un seul message
`unpack`	décompose un message en atomes séparés
`trigger`	déclenche en séquence et convertit des messages
`spigot`	(robinet) ouvre et ferme le passage de messages
`moses`	(moïse) sépare un flux de nombres en deux sorties
`until`	mécanisme de bouclage
`print`	imprime des messages sur la console
`makefilename`	formate un symbole comportant une variable
`change`	filtre les répétitions dans un flux de nombres
`swap`	permute deux nombres
`value`	valeur numérique partagée (variable globale)

Temps

`delay`	envoie un message après un délai
`metro`	envoie un message périodiquement
`line`	envoie une suite linéraire de nombres
`timer`	mesure des intervalles temporels
`cputime`	mesure le temps CPU
`realtime`	mesure le temps par le système d'exploitation
`pipe`	ligne à retard (extensible) pour les messages

Maths

`+ - * / pow`	arithmétique
`== != > < >= <=`	tests de comparaison
`& && | || %`	opérations logiques ou bit-à-bit
`mtof ftom powtodb rmstodb dbtopow dbtorms`	convertions acoustiques
`mod div sin cos tan atan atan2 sqrt log exp abs`	maths supérieures
`random expr`	maths inférieures
`max min`	le plus grand ou le plus petit
`clip`	contraint un nombre à rester dans un intervalle borné

Midi

`notein ctlin pgmin bendin touchin polytouchin midiin sysexin`	entrées MIDI
`noteout ctlout pgmout bendout touchout polytouchout midiout`	sorties MIDI
`makenote`	envoie les *note-on* et fabrique les *note-off* à retarder
`stripnote`	supprime les messages *note-off* de l'entrée

Tables

`tabread`	lit un nombre dans une table
`tabread4`	lit dans une table avec une interpolation à 4 points
`tabwrite`	écrit un nombre dans une table
`soundfiler`	lit et écrit des tables depuis/vers des fichiers audio

Divers

`loadbang`	émet un *bang* au démarrage
`serial`	contrôleur série, pour NT seulement
`netsend`	envoie des messages sur internet
`netreceive`	reçoit les messages de `netsend`
`qlist`	séquenceur de messages depuis un fichier texte
`textfile`	convertit des fichiers en messages
`openpanel`	fenêtre « Ouvrir »
`savepanel`	fenêtre « Enregistrer sous... »
`bag`	ensemble de nombres
`poly`	allocation polyphonique de voies
`key, keyup`	valeurs numériques des touches du clavier
`keyname`	nom symbolique des touches du clavier

Maths audio

`+~ -~ *~ /~`	arithmétique sur les signaux audio
`max~ min~`	maximum et minimum de 2 entrées audio
`clip~`	contraint un signal entre deux bornes
`q8_rsqrt~`	racine carrée inverse rapide (attention 8 bits !)
`q8_sqrt~`	racine carrée rapide (attention 8 bits !)
`wrap~`	reste modulo 1 (partie décimale pour les positifs)
`fft~`	transformée de Fourier discrète complexe
`ifft~`	transformée de Fourier discrète inverse complexe
`rfft~`	transformée de Fourier discrète réelle
`rifft~`	transformée de Fourier discrète inverse réelle
`framp~`	estimation de la fréquence et de l'amplitude FFT
`mtof~ ftom~ rmstodb~ dbtorms~ rmstopow~ powtorms~`	conversions acoustiques

Colle audio

`dac~`	sortie audio
`adc~`	entrée audio
`sig~`	convertit les nombres en signal audio
`line~`	génère des rampes audio
`vline~`	génère des rampes audio haute-précision
`threshold~`	détecte le franchissement d'un seuil par un signal
`snapshot~`	échantillonne un signal (le convertit en nombre)
`vsnapshot~`	échantillonne un signal en haute-précision
`bang~`	envoie un message *bang* après chaque block DSP
`samplerate~`	récupère le taux d'échantillonnage
`send~`	connexions audio à distance « one-to-many »
`receive~`	reçoit le signal du `send~` du même nom
`throw~`	envoie à distance dans un bus additionneur
`catch~`	définit et lit dans un bus additionneur
`block~`	spécifie la taille de bloc et le chevauchement
`switch~`	démarre et stoppe le calcul DSP
`readsf~`	lit un fichier audio depuis le disque dur
`writesf~`	enregistre un fichier audio sur le disque dur

Oscillateurs et tables audio

`phasor~`	générateur d'ondes en dents de scie
`cos~`	cosinus
`osc~`	oscillateur cosinusoïdal
`tabwrite~`	écrit dans une table
`tabplay~`	rejoue une table (sans transposition)
`tabread~`	lit une table (sans interpolation)
`tabread4~`	lit une table avec interpolation à 4 points
`tabosc4~`	oscillateur de table d'onde avec interpolation
`tabsend~`	écrit continuement un bloc dans une table
`tabreceive~`	lit continuement un bloc dans une table

Filtres audio

`vcf~`	filtre passe-bande contrôlé par voltage
`noise~`	générateur de bruit blanc
`env~`	suiveur d'enveloppe (amplitude RMS en dB)
`hip~`	filtre passe-haut
`lop~`	filtre passe-bas
`bp~`	filtre passe-bande
`biquad~`	filtre brut (2 pôles et 2 zéros)
`samphold~`	échantillone la valeur d'un signal et la maintient
`print~`	affiche un ou plusieurs "blocs"
`rpole~`	filtre brut 1-pôle réel
`rzero~`	filtre brut 1-zéro réel
`rzero_rev~`	filtre brut 1-zéro réel inversé en temps
`cpole~ czero~ czero_rev`	idem en complexes

Délai audio

`delwrite~`	écrit dans une ligne à retard
`delread~`	lit une ligne à retard
`vd~`	lit une ligne à retard avec un délai variable

Sous-patchs

`pd`	définit un sous-patch
`table`	tableau de nombres dans un sous-patch
`inlet`	ajoute une entrée à un sous-patch
`outlet`	ajoute une sortie à un sous-patch
`inlet~ outlet~`	versions audio de `inlet` et `outlet`

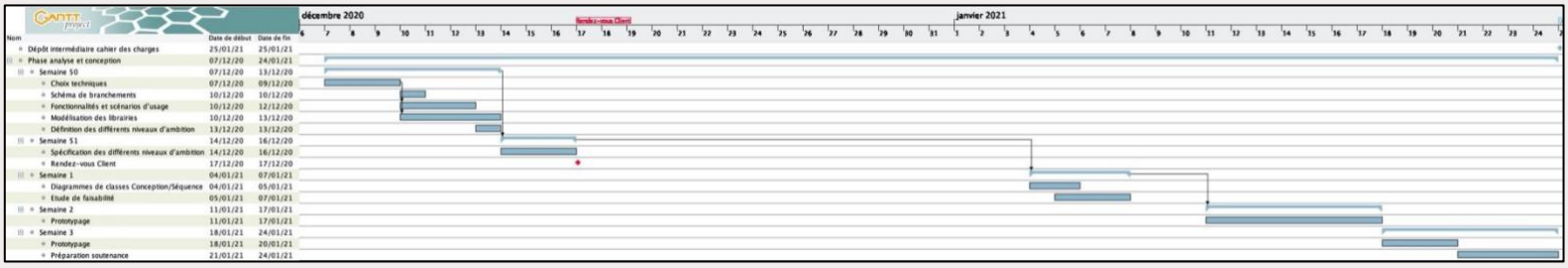
Modèles de données

`struct`	définit une structure de données
`drawcurve, filledcurve`	dessine une courbe
`drawpolygon, filledpolygon`	dessine un polygone
`plot`	trace le champ d'un tableau
`drawnumber`	affiche une valeur numérique

Accès aux données

`pointer`	pointe sur un objet appartenant à un modèle
`get`	récupère des champs numériques
`set`	modifie des champs numériques
`element`	récupère un élément de tableau
`getsize`	récupère la taille d'un tableau
`setsize`	modifie la taille d'un tableau
`append`	ajoute un élément à une liste
`sublist`	récupère une liste depuis le champ d'un scalaire

IV. Planning prévisionnel



Date déf obj	Objectif	Tâches	Support	Etat	Fait le	Deadline idéal	Deadline final	Qui	Avec
19/11/2020	Documentation	Protocole Midi	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	ToutLeMonde	Personne
		AKAI MPD 218	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	ToutLeMonde	Personne
		Raspberry	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	ToutLeMonde	Personne
		Extension WAV	Google Doc	Fait	27/11/2020	27/11/2020	1/12/2020	Romain	Ely
		Pure data	Google Doc	Fait	27/11/2020	30/11/2020	1/12/2020	ToutLeMonde	Personne
		Extension AIFF	Google Doc	Fait	19/11/2020	27/11/2020	1/12/2020	Louis	Alex
19/11/2020	Réunion équipe	Mise en commun des documentations	Discord + Google Doc	Fait		1/12/2020	1/12/2020	ToutLeMonde	Personne
20/11/2020		Rappel de la demande	Word	Fait	04/12/2020	04/12/2020		Alex	Personne
		Présentation du projet choisi	Word	Fait	03/12/2020	04/12/2020		Ely	Personne
		Vision Produit	Word	Fait	02/12/2020	04/12/2020	02/12/2020	Louis	Personne
04/11/2020		Mise en commun des différentes recherches sur les technologies	Word	Fait	03/12/2020	04/12/2020		Romain	Personne
		Planning previsionnel	Word	Fait	07/12/2020	04/12/2020	07/12/2020	Louis	Romain
7/12/2020 (S50)		Périmètre du projet => Choix technique => raspberry	Word	Fait		11/12/2020		Romain	Personne
		Périmètre du projet => Choix technique => Dimensionnement de la batterie	Word	En cours		11/12/2020		Alex	Personne
		Périmètre du projet => Choix technique => langage	Word	Fait	11/12/2020	11/12/2020	11/12/2020	Louis	Personne
		Périmètre du projet => Choix technique => pureData	Word	Fait	10/12/2020	11/12/2020	10/12/2020	Louis	Personne
		Périmètre du projet => Choix technique => OSC	Word	A faire		11/12/2020		Ely	Romain
		Périmètre du projet => Choix technique => librairies Midi	Word	A faire		11/12/2020		ToutLeMonde	Personne
		Périmètre du projet => Choix technique => librairies Son	Word	A faire				ToutLeMonde	Personne
		Périmètre du projet => schémas de branchement	Word	Fait	11/12/2020	11/12/2020	11/12/2020	Ely	Personne
		Périmètre du projet => Fonctionnalitée et scénarios d'usages avec contraintes	Word	Fait	11/12/2020	11/12/2020	11/12/2020	Louis	Personne
		Périmètre du projet => Algorigramme point de vue utilisateur	Word	Fait	12/12/2020	12/12/2020	12/12/2020	Louis	Personne
		Périmètre du projet => Modélisation librairies Midi	Word	En cours		11/12/2020		Louis	Personne
		Périmètre du projet => Modélisation librairies Son		A faire				ToutLeMonde	Personne
		Définition du prix global du projet	Word	En cours		13/12/2020		Louis	Personne
		Définition des différents niveaux d'ambition	Word	Fait	12/12/2020	13/12/2020	12/12/2020	Louis	Personne
7/12/2020 (S51)		Spécification des différents niveaux d'ambition=> Version 1	Word	En cours		16/12/2020		Louis	Personne
		Spécification des différents niveaux d'ambition=> Version 2	Word	A faire		16/12/2020		ToutLeMonde	
		Spécification des différents niveaux d'ambition=> Version 3	Word	A faire		16/12/2020		ToutLeMonde	
		Spécification des différents niveaux d'ambition=> Version 4	Word	A faire		16/12/2020		ToutLeMonde	
		Spécification des différents niveaux d'ambition=> Version 5	Word	A faire		16/12/2020		ToutLeMonde	
		Spécification des différents niveaux d'ambition=> Version 6	Word	A faire		16/12/2020			
		RDV CLIENT	IUT			17/12/2020		ToutLeMonde	
7/12/2020 (S1)		Diagrammes de classes Conception/Séquence	Word						
		Etude de faisabilité	Word						
7/12/2020 (S2/S3)		Prototypage	Word						
	Cahier des charges	Préparation soutenance	Word						
07/12/2020 (S4)		Soutenance							

Au vu de notre diagramme de Gantt, la période des vacances de noël sera destinée à se réunir au moins une fois par semaine et à se focaliser sur le prototypage. Des tests de programmation pourront alors être effectués.

V. Périmètre du projet

Afin de délimiter au mieux ce projet, considérons notre objectif à réaliser pour répondre à la demande de notre client.

Au moyen des différentes discussions et démonstrations réalisées avec M. Bestel mais aussi à travers l'énumération de sa demande, nous avons essayé de définir le périmètre le plus cohérent.

D'un point de vue global, nous avons donc pour objectif de développer une application permettant de gérer le déclencheur de samples à l'aide de différents composants matériels et logiciels. Ainsi, nous allons définir l'ensemble des choix techniques afin de pouvoir ensuite énumérer en détail le projet. Dans cette partie, il sera spécifié en détail le niveau d'ambition de réalisation le plus haut afin de comprendre entièrement ce projet.

a) Fonctionnalités et contraintes

A travers cette partie, nous allons définir toutes les fonctionnalités présentes au sein de notre future conception ainsi que des scénarios d'usage.

Pour l'instant, il s'agit d'une énumération générale des fonctionnalités concernant le niveau d'ambition le plus extrême. Elles seront segmentées lors du détail des différents niveaux d'ambition (versions du projet).

- #### Fonctionnalités

De manière générale, lorsqu'un pad est appuyé, la LED placée en dessous de celui-ci reste allumée jusqu'à la fin de lecture du son.

1) L'utilisateur peut lancer en boucle un son.

2) La lecture d'un son peut être arrêtée au moyen d'un deuxième appui sur le pad à partir duquel il a été lancé.

3) Lorsque l'utilisateur émet une pression sur un pad, un son est lancé directement. Ceci est appelé un « trigger ».

4) Gérer la possibilité d'avoir deux sons se jouant en même temps. Il s'agit d'une boucle et d'un « trigger ».

5) Mise en place d'un système pouvant appliquer la réverbération sur le son joué à l'aide des potentiomètres du contrôleur AKAI MPD 218.

6) Une interface graphique permet à l'utilisateur d'assigner des sons aux différents pads en le branchant au microcontrôleur.

7) Le produit donnera la possibilité de gérer la vélocité d'un pad.

- Contraintes

1) L'utilisateur doit pouvoir allumer l'appareil et mettre en route le dispositif en un minimum de temps.

2) Le temps d'utilisation sur batterie doit être au minimum de trois heures.

3) La batterie doit alimenter le microcontrôleur et le clavier AKAI MPD 218.

4) Les fichiers son portent l'extension WAV ou AIFF.

5) Le stockage de ceux-ci doit se faire sur une carte micro SD.

6) Un pad correspond à un son

7) La sortie du son doit se faire par la prise Jack.

8) Le moyen de stockage recevant les sons doit avoir une capacité supérieure à 2 GO. Ce dernier doit être sur le microcontrôleur.

9) Une latence minimale de 12 m/s est requise entre le moment où l'utilisateur appuie sur le pad et le moment où le son se lance.

10) Une utilisation du protocole MIDI est requise.

11) Rester dans des tarifs assez bas pour pouvoir cloner des systèmes de secours.

12) La batterie doit être légère et posséder un câblage déjà fait.

13) La mise sous tension s'opère par une sortie USB-C.

b) Choix techniques

- Microcontrôleur

Pour notre PTS nous avons eu besoin de trouver un type de micro-ordinateur. Après quelques recherches nous avons trouvé 3 types de micro-ordinateur. Voici donc les différences de chacun d'entre eux.

	Arduino ATmega 2560	**Raspberry Pi 4 model B**	**Raspberry Pi 3 model B**	**Raspberry Pi Zero**
Tension d'alimentation (d'ouverture)	**5V**	5V	**5V**	**5V**
Mémoire RAM	8ko (statique)	**8Go (dynamique)**	1Go (dynamique)	512 Mo (dynamique)
Mémoire ROM min-max	256Kb (Flash)	**8Go-128 Go**	**8Go-128 Go**	0Go-32 Go
Vitesse de l'horloge	16MHz			
Vitesse processeur		**1.5GHz**	1.2GHz	1 GHz
Port USB A	Aucun	**2 * 3.0** 2 * 2.0	4 * 2.0	Aucun (2 micro-USB type B)
Prise Jack	Non	**Oui**	**Oui**	Non
Prix (€)	40	78	37	**20**
Ventilateur	Non	**Oui**	**Oui**	Non
OS	Non	Raspbian	Raspbian	Raspbian

Au vu des caractéristiques présentées, le micro-ordinateur Arduino ne détenant pas de port USB, la solution de l'ajout d'un composant ne nous paraît pas judicieux à notre niveau. De même pour le Raspberry Pi Zero. De plus, l'utilisation d'adaptateur ne conserverais pas une grande vitesse de transmission des données. Nous ne pouvons pas les choisir du fait qu'on ne pourra pas leur connecter le AKAI. Nous voulons un micro-ordinateur qui soit le plus rapide possible afin d'avoir le temps de latence le plus petit possible pour respecter le cahier des charges. Ceci est caractérisé par la mémoire RAM et la vitesse du processeur qui se montre bien meilleur chez la Raspberry Pi 4 model B. Enfin, le fait que l'Arduino et le raspberry PI Zero ne possède pas de prise Jack élimine tout utilisation de ces technologies.

- Dimensionnement de la batterie

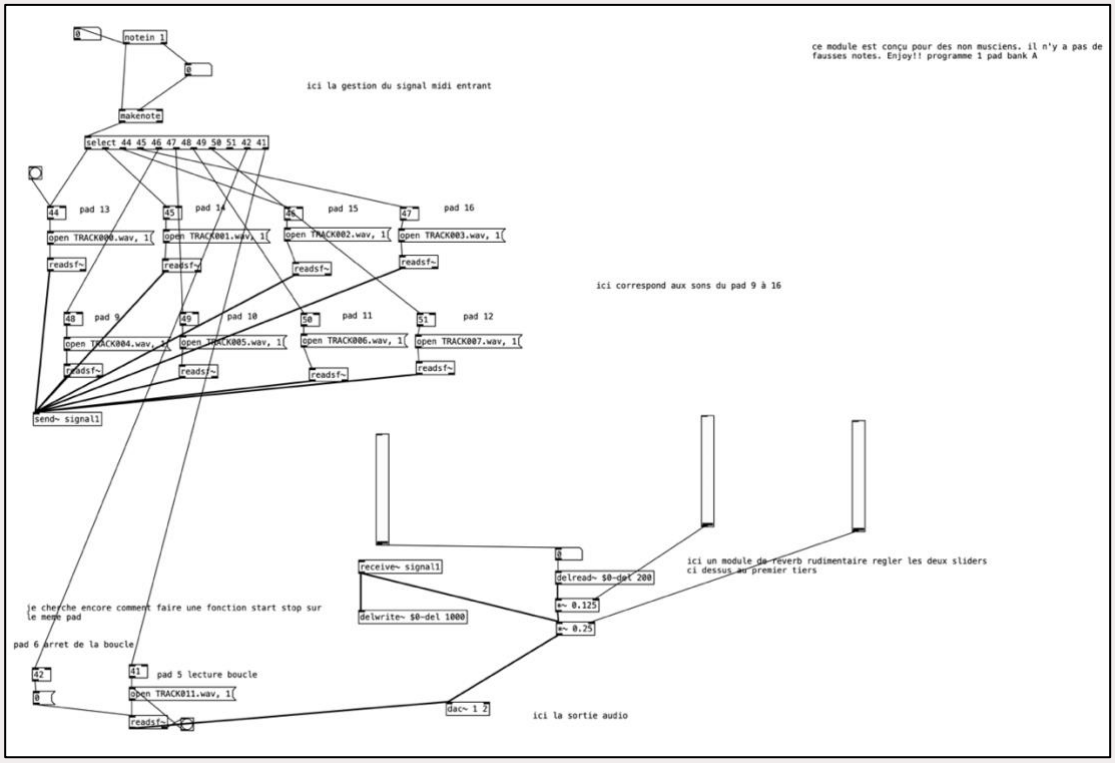
Nous savons que l'autonomie demandée est de 3 heures au minimum. Nous avons choisi une autonomie de 4h pour les éventuels imprévus et les temps de pause et d'attente.
L'alimentation du Raspberry Pi 4 est de 5V et 3A (pris en compte des périphériques). Ainsi, pour trouver la puissance nous faisons :
Puissance = Tension * Intensité (P=U*I) = 5*3 = 15W .
La puissance nominale est donc de 15 Watt.
Pour la batterie, nous devons calculer la valeur de sa capacité en Ah (ampère-heure) à partir d'une une tension d'alimentation correspondante au Raspberry de 5V. Pour la capacité, il suffit de multiplier l'intensité du Raspberry par le nombre d'heures d'autonomie. On obtient donc 12 Ah.
Nous avons donc la tension et la capacité de la batterie, mais pour la tension nous optons pour du 6V car les batteries de 5V n'existent pas. La batterie sera donc en 6V avec une capacité de 12 Ah. Nous devons aussi prendre en compte l'usure de la batterie, on décide donc de rajouter 20% de sa capacité, ce qui nous donne (12*20/100 = 2.4 | 12+2.4 = 14.4) une capacité de 14.4 Ah. On arrondi donc à une capacité de 15 Ah.

- Langages

La Raspberry PI 4 peut embarquer des programmes écrit en langage Python, Scratch, C, C++. Durant notre formation nous avons été formés principalement sur le C/C++. Nous pratiquons très bien ce langage. Au vu de notre projet et d'après nos recherches, beaucoup de librairies sont écrites en C/C++. Cependant, nous constatons des possibilités intéressantes avec le langage Python sur lesquelles nous pourrons travailler.

- Logiciels

Comme nous l'avons évoqué précédemment, PureData est le logiciel utilisé par notre client afin de paramétrer son contrôleur AKAI MPD 218. Nous allons donc utiliser ce logiciel et analyser son code source afin de pouvoir s'inspirer des fonctionnalités que propose cette interface. L'objectif ici, sera de comprendre quelles fonctionnalités seront utiles afin de les extraire et de pouvoir se les approprier. Suite à la démonstration, M. Bestel nous a donné un fichier programmé de façon graphique avec le logiciel afin de pouvoir comprendre comment fonctionne le paramétrage du contrôleur musical. Expliquons de manière générale ce que fait ce programme pour envoyer un son. Nous pourrons ainsi faire une analogie entre ces fonctionnalités et celles qui seront à reproduire.



Tout d'abord la fonction **notein** lit les notes MIDI reçues par PureData et envoi le numéro de la note reçue ainsi que la valeur de sa vélocité et son numéro de canal. Ensuite la fonction **makenote** va programmer une note retardataire à partir des trois données récupérées précédemment. La fonction **select** va alors comparer le numéro de pad reçu avec ceux qu'elle contient dans sa liste. Si le pad est présent, un bang est émis vers la sortie représentant le pad touché. A la suite de cette action, la fonction **open** suivi du nom de fichier permet d'ouvrir ce dernier et de le lire grâce à l'instruction **readsf~**. La fonction **Send~ signal1** permet de communiquer avec **receive~ signal1** afin de ne pas ajouter une multitude de câbles et de garder une interface lisible. Enfin nous constatons en bas de page pour les pads 5 et 6 qu'un module de réverbération leur est destiné mais n'est pas fonctionnel. La réverbération permet sur un son musical de donner un effet d'écho.

- Librairies

Pour concevoir notre application, nous faisons le choix d'utiliser des librairies. Nous avons donc analysé deux parties distinctes que le programme pourrait comporter. Tout d'abord une partie correspondra à un établissement de la connexion entre le contrôleur et l'ordinateur à travers le protocole MIDI. De plus, nous devrons récupérer le signal de chaque pad envoyé afin de les différencier. Pour ce faire nous avons sélectionné une librairie se nommant RtMidi. Celle-ci est disponible en C/C++ et une liaison avec cette dernière est disponible en Python pour les entrées et sorties d'appareil MIDI.

Ensuite, nous avons besoin de lancer un son. D'après notre utilisation, nous avons besoin d'ouvrir le fichier son, le charger, le jouer (une fois ou en boucle), l'arrêter. Lors de la recherche d'une bibliothèque afin de lancer les fichiers sons dans le langage C/C++ avec notre sampler, nous n'en avons pas trouvé. Nous avons donc la possibilité de créer notre propre fonction de lecture de son en C/C++ à partir des commandes terminales du système d'exploitation Raspbian qui fonctionne de la même manière que Linux. Cette façon s'appuie sur la bibliothèque « SoX » (Sound eXchange). « SoX » lit et écrit des fichiers audio dans les formats les plus courants et peut éventuellement leur appliquer des effets. Cette librairie peut nous permettre de combiner plusieurs sources d'entrées audio. Sur de nombreux systèmes, il peut agir comme un lecteur ou un enregistreur audio multipiste. Une autre possibilité est d'utiliser également des commandes systèmes depuis le langage Python ou bien d'utiliser le sous module (équivalent à une librairie dans le monde de ce langage) Mixer qui est inclus dans le module principal Pygame.

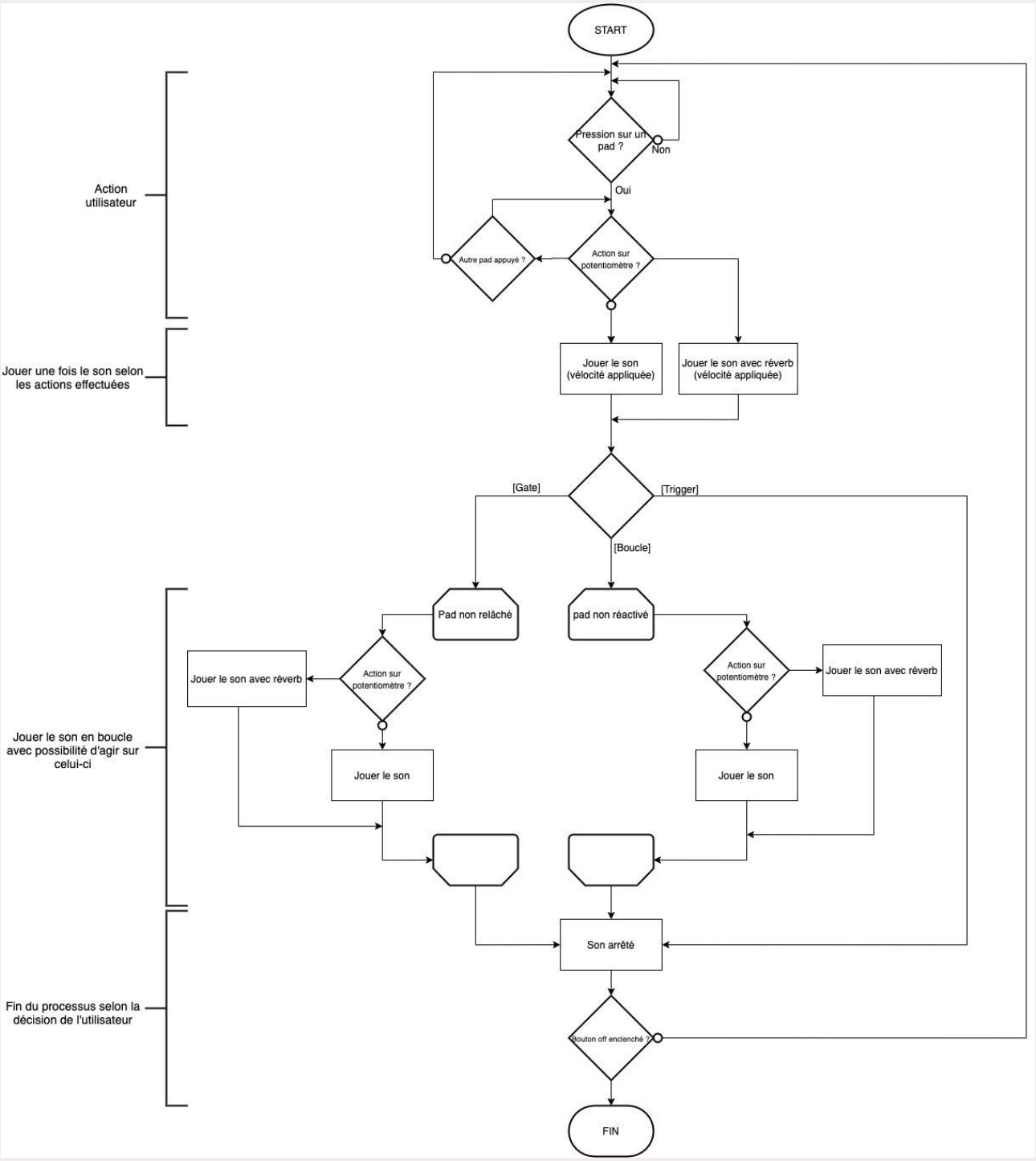
c) Scénario d'usage

Définissons le scénario nominal concernant l'utilisation complète du produit prévu.

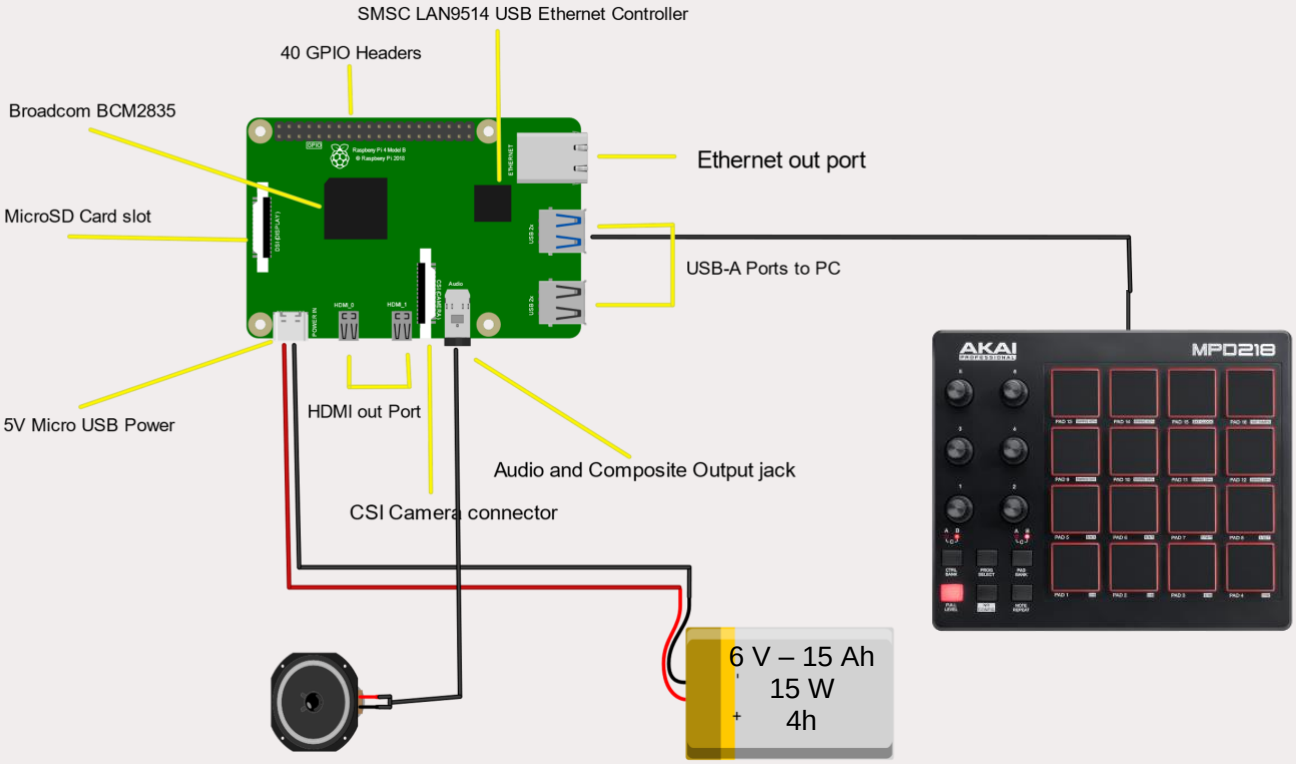
L'utilisateur met en route le microcontrôleur relié à un écran. Le AKAI est branché à ce dernier. Il se rend sur l'interface graphique développée. Il choisit pour les pads qu'il souhaite les sons à leur assigner. Une fois cette action faite, l'utilisateur peut réaliser les branchements et allumer le dispositif. Il peut ainsi lancer ses sons de plusieurs manières : d'un simple appui (« trigger »), en boucle, ou bien jouer un son en maintenant l'appui sur un pad (« gate »). Une fois ce dernier relâché, le son s'arrête. Lorsqu'une boucle est lancée, la possibilité d'arrêter le son lui sera donnée. Deux sons pourront être lancés simultanément, c'est-à-dire un trigger et une boucle. Selon son envie, il pourra appliquer une réverbération sur le son lancé ainsi que le jouer selon une certaine vélocité appliquée sur le pad. Une autre possibilité sera de jouer un cluster, c'est-à-dire de déclencher plusieurs triggers simultanément.

d) Logigramme « Client »

Afin d'expliquer les fonctionnalités que devra remplir le produit final, constatons ci-dessous une représentation graphique du processus entier. Ce logigramme a pour objectif de démontrer ces fonctionnalités de façon simple et compréhensible pour n'importe quel utilisateur. Pour celui-ci aucune partie ou morceau de code est modélisé.



e) Schéma de branchement



f) Coût du projet

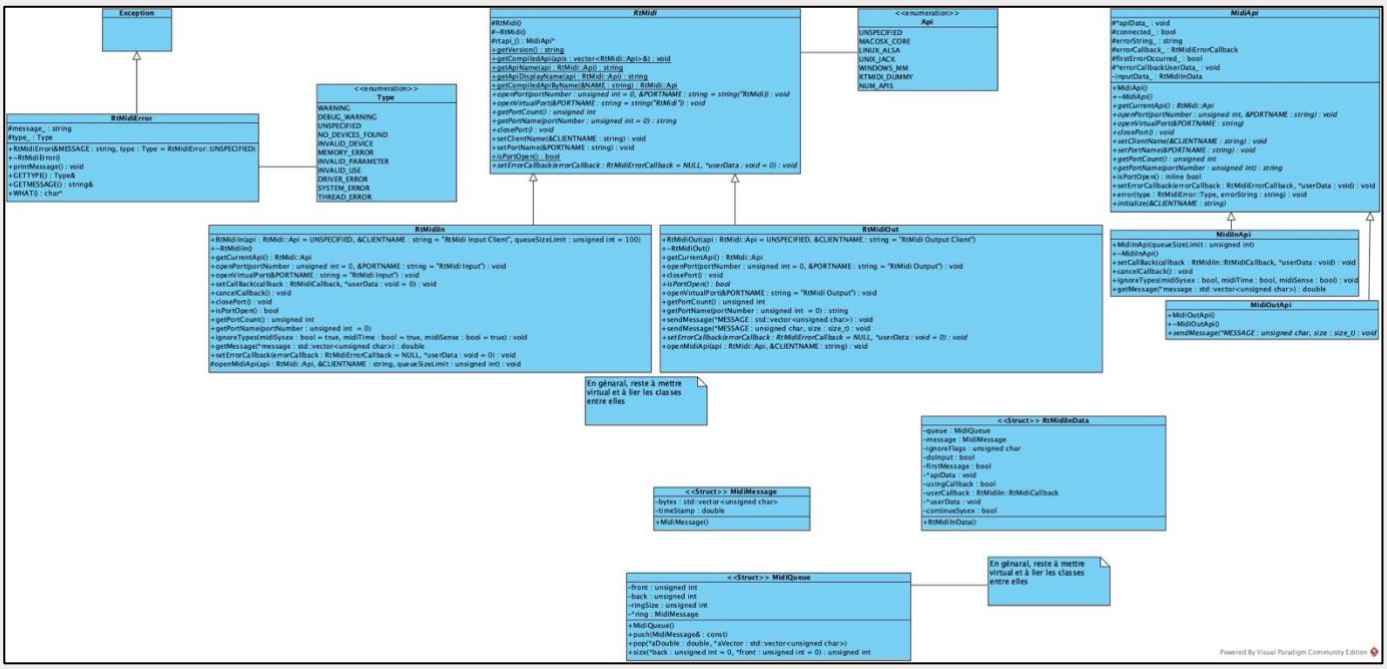
Matériel	Prix
Akai MPD	85€
Raspberry PI 4	78€
Boitier	10€
Ventilateur	6€
Dissipateur thermique	3€
Câble HDMI	5€
Carte SD de 32Go	13€
Batterie 6V 15 Ah	23€
	Total : 223 €

g) Modélisation des librairies

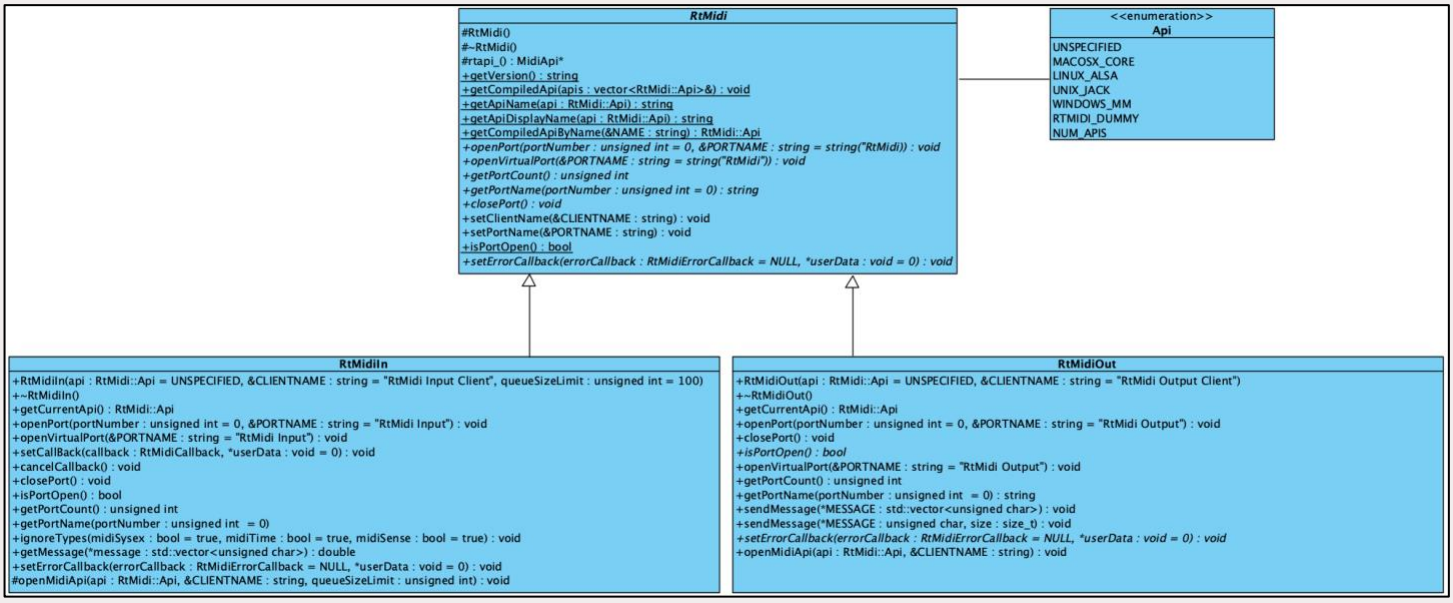
- Midi

Retrouvons ci-dessous, la modélisation complète de la librairie RtMidi dans le langage C++. D'après la documentation fournie et les explications des classes, celles qui nous intéressent sont RtMidiIn et RtMidiOut. Elles permettent de gérer les messages MIDI entrant et sortant.

C++ :

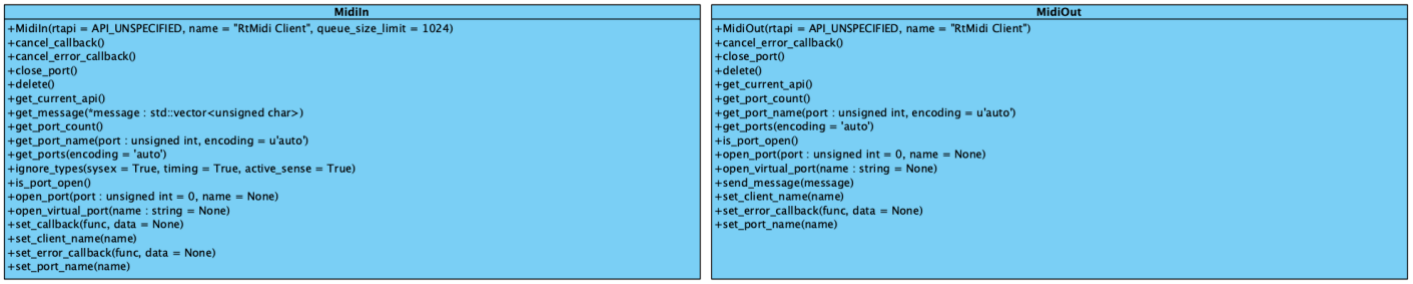


Précisément, voici les classes qui nous intéressent. Il s'agit des entrées et sorties de messages MIDI.



Liaison Python :

Les deux classes précédentes ont été réalisées en Python. Ceci se nomme une liaison. En voici une modélisation et une explication de chaque méthode, c'est-à-dire, ce qu'elle réalise. Ces explications seront similaires dans les deux langages.



Classe RtMidi.MidiIn :

cancel_callback()

Supprimer la fonction de rappel enregistrée pour l'entrée MIDI.

cancel_error_callback()

Supprimer la fonction de rappel enregistrée pour les erreurs.

close_port()

Fermer un ou des ports MIDI

delete()

Désallouer le pointeur vers l'instance C++.

get_current_api()

Retourner l'API MIDI pour l'instance actuelle de RtMidiIn.

get_message(*message : std ::vector<unsigned char>)

Récupérer un message MIDI envoyé depuis un appareil MIDI. Renvoie un tuple à deux éléments, le message MIDI et un temps delta. Le message MIDI est une liste d'entiers contenant les données du message (numéro de la note, identifiant du pad, vélocité). Le delta est un float représentant le temps en secondes écoulé depuis la réception de l'évènement MIDI.

get_port_count()

Retourne le nombre de port(s) MIDI (entrant et sortant) disponible.

get_port_name(port : unsigned int, encoding = u'auto')

Renvoie le nom du port d'entrée MIDI avec un numéro (exemple : port 0, port 1…)

get_ports(encoding = 'auto')

Retourne une liste de noms des ports d'entrée MIDI disponibles. L'index de liste de chaque nom correspond à son numéro de port.

ignore_types(sysex = True, timing = True, active_sense = True)

Activer ou désactiver le filtrage d'entrée de certains types d'évènements MIDI.

is_port_open()

Retourne True si un port a été ouvert et False sinon.

open_port(port : unsigned int = 0, name = None)

Ouvrir un port d'entrée MIDI avec son numéro correspondant en paramètr

open_virtual_port(name : string = None)

Ouvrir un port d'entrée MIDI virtuel.

set_callback(func, data=None)

Enregistrer une fonction de rappel pour l'entrée MIDI.

set_client_name(name)

Définir le nom du client MIDI.

set_error_callback(func, data=None)

Enregistrer une fonction de rappel pour les erreurs.

set_port_name(name)

Définir le nom du port actuellement ouvert.

Classe RtMidi.MidiOut :

cancel_error_callback()

Supprimer la fonction de rappel enregistrée pour les erreurs.

close_port()

Fermer un ou des ports MIDI.

delete()

Désallouer le pointeur vers l'instance C++.

get_current_api()

Retourner l'API MIDI pour l'instance actuelle de RtMidiIn.

get_port_count()

Retourne le nombre de port(s) MIDI (entrant et sortant) disponibles.

get_ports(encoding = 'auto')

Retourne une liste de noms des ports de sortie MIDI disponibles. L'index de liste de chaque nom correspond à son numéro de port.

is_port_open()

Retourne True si un port a été ouvert et False sinon.

open_port(unsigned int port = 0, name = None)

Ouvrir un port de sortie MIDI avec son numéro correspondant en paramètre.

open_virtual_port(name=None)

Ouvrir un port d'entrée ou de sortie MIDI virtuel.

send_message(message)

Envoyer un message MIDI vers le port de sortie.

set_client_name(name)

Définir le nom du client MIDI.

set_error_callback(func, data=None)

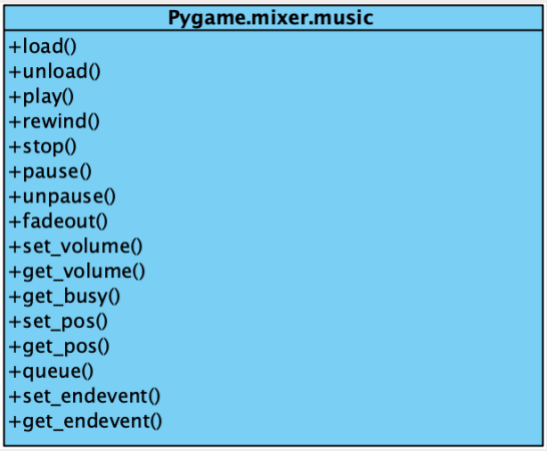
Enregistrer une fonction de rappel pour les erreurs.

set_port_name(name)

Définir le nom du port actuellement ouvert.

- Son

Pygame :



load()

Charger un fichier de musique pour le préparer à la lecture .

unload()

Décharger les fichiers musiques chargés pour libérer les ressources

play()

Démarrer la lecture de la musique.

rewind()

Redémarrer la musique (Réinitialise la lecture de la musique actuelle au début.)

stop()

Arrêter la lecture de la musique si elle est en cours de lecture. Attention, le fichier n’est pas déchargé.

pause()

Mettre en pause la musique.

unpause()

Reprendre la lecture d’une musique mise en pause.

fadeout

Arrêter la musique avec un effet de fondu. Attention, cette fonction se bloque jusqu'à ce que la musique disparaisse. Les appels à fadeout() et set_volume() n'auront aucun effet pendant ce temps. Si un événement a été défini à l'aide de set_endevent(), il sera appelé une fois que la musique aura été arrêtée.

set_volume()

Régler le volume de la musique. Le paramètre est un nombre de type float entre 0,0 et 1,0.

get_volume()

Obtenir le volume de la musique. La valeur renvoyée est un float compris entre 0,0 et 1,0.

get_busy()

Permet de savoir si une musique est cours de lecture. Cette fonction renvoie « True » si une musque est cours de lecture et « False » sinon.

set_pos()

Permet de définir la position à laquelle jouer le son.

get_pos()

Obtenir la position (temps de lecture) en temps réel de la musique. Ceci est renvoyé sous la forme d'un nombre de millisecondes pendant lequel la musique a été joué.

queue()

Mettre un ou des sons en file d'attente. Ceci permettra de jour les sons désignés à la fin de celui qui est en cours

set_endevent()

Définir un signal/évènement que la musque doit envoyer/émettre lorsqu'elle se termine.

get_endevent()

Obtenir le signal/évènement que la musique doit en envoyer lorsqu'elle s'arrête. S'il aucun signal est trouvé, la fonction renvoie « NOEVENT ».

Appels systèmes Sox :

Voici le fonctionnement de la bibliothèque « SoX » :

Les commandes de la bibliothèque SoX s'invoquent avec les mots clés play, sox et rec. Dans notre cas nous n'utiliserons pas les commandes rec.

Les commandes play :

Cette commande accepte tout type de fichier son.

$ play fichier

Augmenter le volume courant de n fois (n est un chiffre) :

$ play fichier vol n

Les commandes sox :

Cette ligne transforme un fichier son d'une extension à une autre.

$ sox fichier.extension fichier.autreExtention

Cette commande fusionne deux fichiers sons. Cela permet d'avoir deux sons l'un a côté de l'autre dans un seul fichier.

$ sox fichier fichier2 fichierFinal

La commande suivante permet de modifier un fichier afin de lui ajouter un filtre d'ajout de décibel. Le résultat sera placé dans un nouveau fichier.

$ sox entree.wav sortie.wav vol 10db

Cette ligne sert à changer la fréquence d'échantillonnage du fichier et placer le résultat dans un nouveau fichier.

$ sox entree.wav -r 22050 sortie.wav

Appliquer une réverbération à un fichier.

$ sox entree.wav sortie.wav lowp 1000.0

Appliquer un fichier son à deux canaux (deux commandes peuvent effectuer cette tâche).

$ sox entree.wav -c 2 sortie.wav split

$ sox left.wav right.wav -c 2 sortie.wav -M

Cette dernière commande par rapport à la précédente défini que parmi deux fichiers, l'un sera entendu à gauche et l'autre à droite par le biais d'un changement de channel. Cette combinaison sera appliquée dans un fichier résultat. Ce fichier sera à lire avec un système de son en stéréo.

Les commandes système que nous venons d'expliquer, « play » et « sox » viennent de la bibliothèque de commandes systèmes « SoX ».
Grâce à ces commandes nous pouvons jouer un son et interagir avec par le biais de filtre.

VI. Définition des différents niveaux d'ambition

Résumons dans cette partie les différents niveaux d'ambitions que nous souhaitons intégrer à notre produit final. Une version devra correspondre à une fonctionnalité.

a) Version 1

Tout d'abord cette première version concerne une utilisation minimale du produit à développer. Elle sera ainsi moins complète que les versions suivantes. L'objectif de celle-ci est de pouvoir permettre à l'utilisateur d'appuyer sur un pad afin de lancer un son depuis le contrôleur AKAI MPD 218. Il s'agit du déclenchement d'un trigger. Dans cette version, les sons à jouer seront utilisés à travers leur nom de fichier correspondant à un pad particulier.

b) Version 2

Cette version englobera la précédente et une nouvelle fonctionnalité sera ajoutée. L'utilisateur aura la capacité de lancer une lecture d'un son en boucle et celui-ci pourra être arrêté en appuyant à nouveau sur le pad correspondant.

c) Version 3

En plus des deux versions précédentes, celle-ci donnera la possibilité de pouvoir gérer le lancement de plusieurs sons en même temps. Pour une boucle, une seule à la fois est jouée. Cependant, pour les sons « trigger », le musicien peut réaliser un cluster en lançant jusqu'à 4 sons simultanément. Lorsqu'une boucle est en cours, il est possible de déclencher un trigger simultanément. Dans ce cas, si l'utilisateur appui sur un nouveau pad le nouveau son attend que l'autre soit terminé s'il s'agit d'une boucle.

d) Version 4

Cette version vise à permettre l'utilisation des potentiomètres du contrôleur. Une fois un son lancé au moyen d'un pad, l'utilisateur pourra faire varier la valeur de résistance du potentiomètre afin d'appliquer une réverbération au son joué. Ces derniers prennent des valeurs entre 0 et 255. Ce principe s'appuie sur la notion d'une réverbération wet/dry (mouillé/sec en français). Plus le potentiomètre tend vers 255, plus l'acoustique du son joué prend un effet mouillé et inversement. Il s'agit d'une réverbération en sortie du son et non sur toute sa durée.

e) Version 5

Cette version n'implémentera pas une nouvelle fonctionnalité pour le contrôleur en lui-même mais une interface graphique sur le microcontrôleur pour simplifier la configuration du clavier. Ce logiciel aura pour objectif d'assigner des sons aux pads.

f) Version 6

Enfin, cette version donnera la possibilité de jouer un son à un volume correspondant à la force donnée sur le pad. Il s'agit de la vélocité. Plus la vélocité est grande, plus le son sera joué à un volume important.

g) Version 7

Cette version prendra en compte l'ajout de la fonctionnalité appelée « gate ». Cette fonctionnalité réside dans le fait que l'utilisateur appuiera sur un pad en le maintenant pour jouer un son de la durée voulue. En parallèle d'un gate, un trigger, un cluster ou bien une boucle pourront être lancés.

VII. Spécification détaillée des niveaux d'ambition

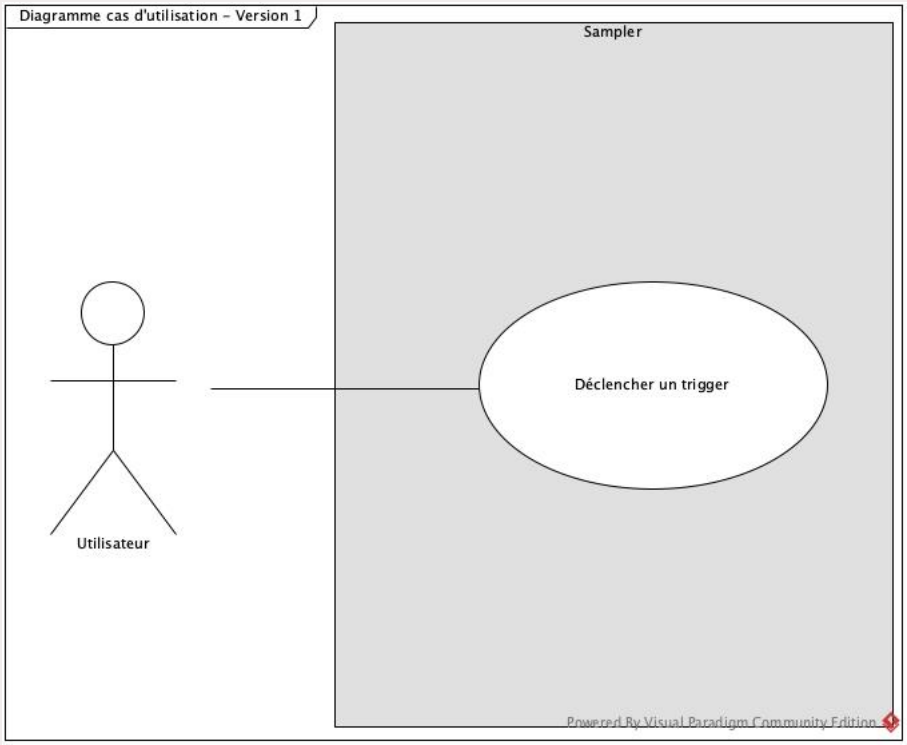
Analysons dès à présent chaque niveau d'ambition en détail. Dans chaque version, les contraintes énumérées précédemment seront à respecter obligatoirement.

a) Version 1

- Fonctionnalités associées

Lorsque l'utilisateur émet une pression sur un pad, un son est lancé directement.

- Scénario

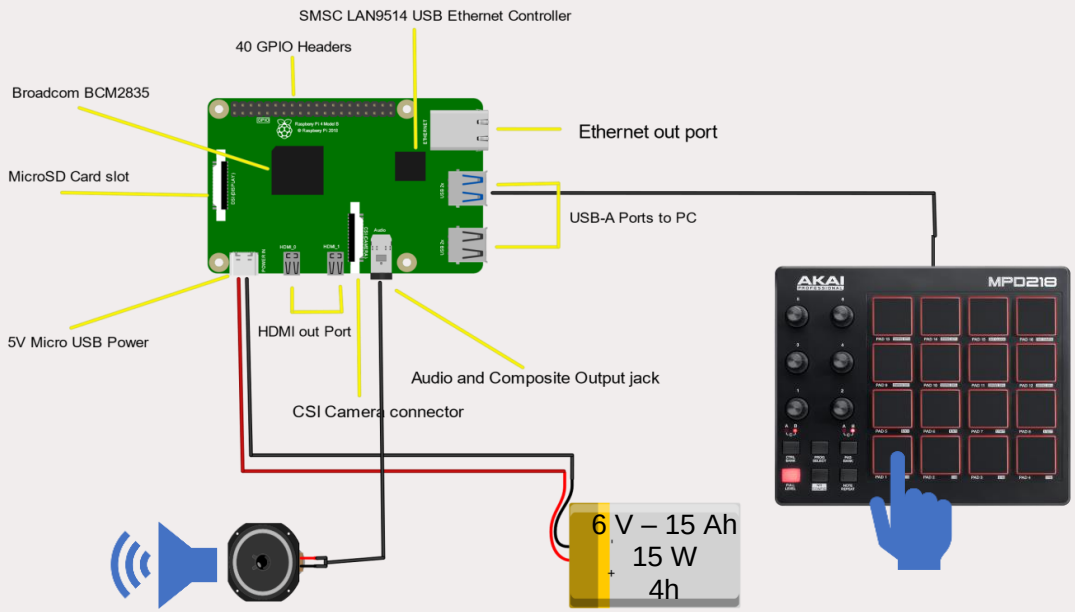


Ce diagramme étant réalisé pour la première version, il contient seulement un cas qui est « Lancer un trigger ». D'autres cas viendront se greffer à lui lors des spécifications des niveaux suivants. Constatons les scénarios le caractérisant :

Description détaillée du cas d'utilisation « Déclencher un trigger »

Cas d'utilisation - description simplifiée
PROJET : PTS3 : Conception d'un sampler portable
Cas d'utilisation : Déclencher un trigger

Référence : **Diagramme cas d'utilisation – Version 1**	Version : V1
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : **13/12/2020**
Résumé : Permet à l'utilisateur de lancer un son (nommé trigger)	
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**	
Références des documents associés : - Définition de la version 1 - Schéma d'utilisation - Diagramme d'analyse/conception (librairies utilisées)	
Description des enchaînements d'interactions du scénario nominal : % précondition : l'utilisateur a mis sous tension le dispositif % - L'utilisateur appui sur un pad. Ainsi, un son se joue une unique fois. Il peut lancer un son autant de fois qu'il le souhaite.	
Liste des enchaînements alternatifs ou d'exception	
- L'utilisateur décide de lancer un second son. Cependant, le premier n'est pas terminé. Ainsi, le premier son est remplacé par le nouveau. - L'utilisateur décide de lancer un second son. Cependant, le premier n'est pas terminé. Ainsi, le premier son est joué en simultané. - L'utilisateur décide de lancer un second son. Cependant, le premier n'est pas terminé. Ainsi, le second son est joué une fois le premier terminé. - L'utilisateur décide de lancer un second son. Cependant, le premier n'est pas terminé. Ainsi, le second son ne sera pas lancé.	



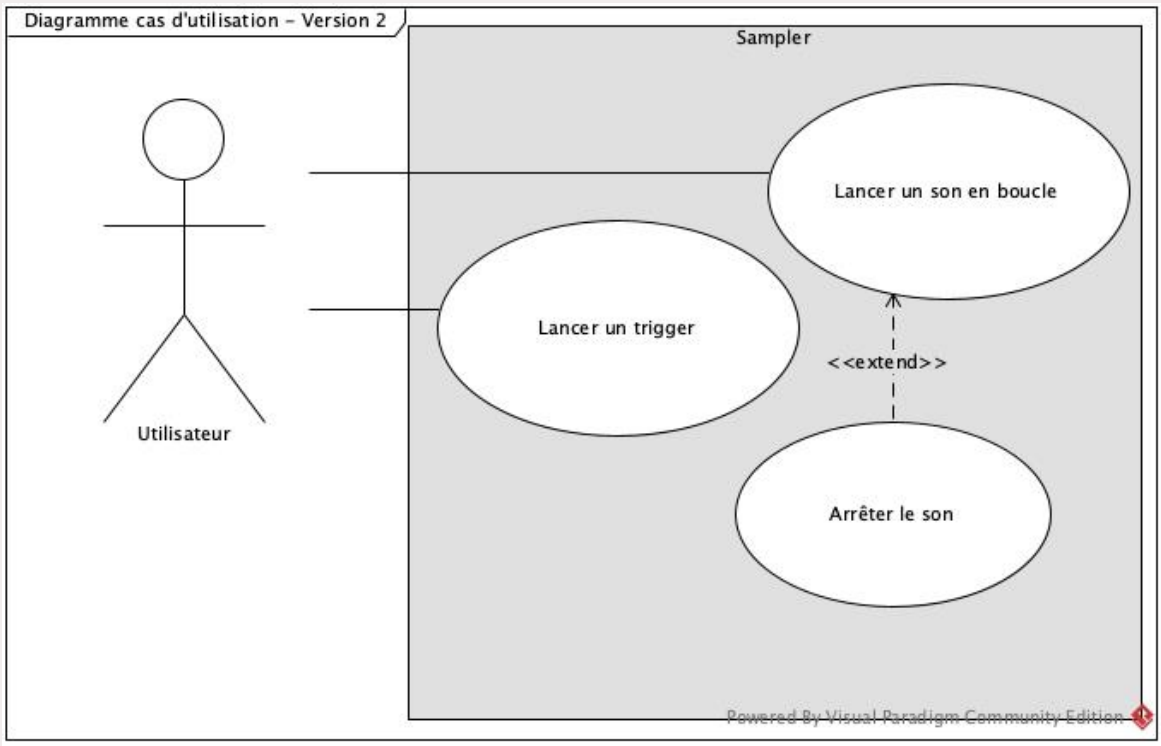
b) Version 2

- ### Fonctionnalités associées

La fonctionnalité précédente est conservée et une nouvelle fonctionnalité vois le jour. Dès que l'on appuie sur certains pads, le son est automatiquement lancé en boucle. Pour arrêter ce son, il suffira d'appuyer à nouveau sur le même pad.

- ### Scénario

Pour cette nouvelle version, le cas d'utilisation vu dans la précédente, a subi une mise à jour et se nomme désormais « Lancer un son en boucle ». Ce dernier inclus une autre fonctionnalité concernant l'arrêt du son lancé.



Description détaillée du cas d'utilisation « Lancer un son en boucle»

Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Lancer un son en boucle	
Référence : **Diagramme cas d'utilisation – Version 2**	Version : V2
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : **14/12/2020**

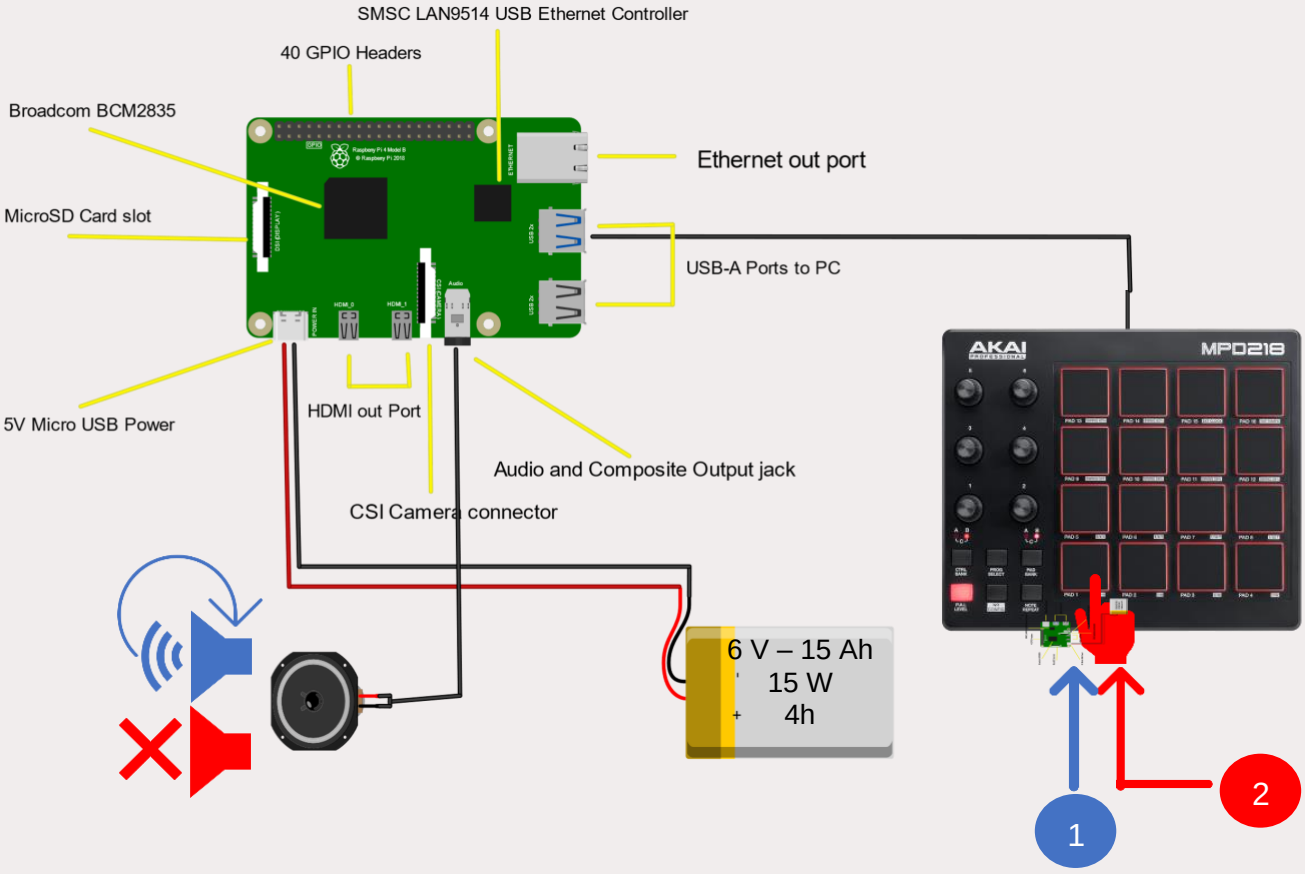
Résumé : Permet à l'utilisateur de lancer un son en boucle.
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**
Références des documents associés : - Définition de la version 2 - Schéma d'utilisation - Diagramme d'analyse/conception (librairies utilisées)
Description des enchaînements d'interactions du scénario nominal : % précondition : l'utilisateur a mis sous tension le dispositif % - L'utilisateur appui sur un pad. Ainsi, le son se joue en boucle, c'est-à-dire qu'une fois qu'il se termine, il se relance automatiquement sans aucune action utilisateur.
Liste des enchaînements alternatifs ou d'exception
- L'utilisateur décide de lancer un second son (boucle ou trigger) simultanément. Cependant, le premier n'est pas terminé et étant lancé en boucle, le nouveau son ne sera pas lancé. - L'utilisateur décide de lancer un second son (boucle ou trigger). Cependant, le premier n'est pas terminé. Ainsi, le premier son est remplacé par le nouveau. - L'utilisateur décide de lancer un second son (boucle ou trigger). Cependant, le premier n'est pas terminé. Ainsi, le premier son est joué en simultané. - L'utilisateur décide de lancer un second son (boucle ou trigger). Cependant, le premier n'est pas terminé. Ainsi, le second son est joué une fois le premier terminé.

Description détaillée du cas d'utilisation « Arrêter le son »

Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Arrêter le son	
Référence : **Diagramme cas d'utilisation – Version 2**	Version : V2
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : **14/12/2020**
Résumé : Permet à l'utilisateur d'arrêter un son en boucle.	
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**	

Références des documents associés : - Définition de la version 2 - Schéma d'utilisation - Diagramme d'analyse/conception (librairies utilisées)
Description des enchaînements d'interactions du scénario nominal : % précondition : l'utilisateur a mis sous tension le dispositif et a lancé un son en boucle % - Afin d'arrêter le son qui se lit en boucle, l'utilisateur appui une seconde fois sur le même pad pour mettre fin à la lecture.

- Schéma



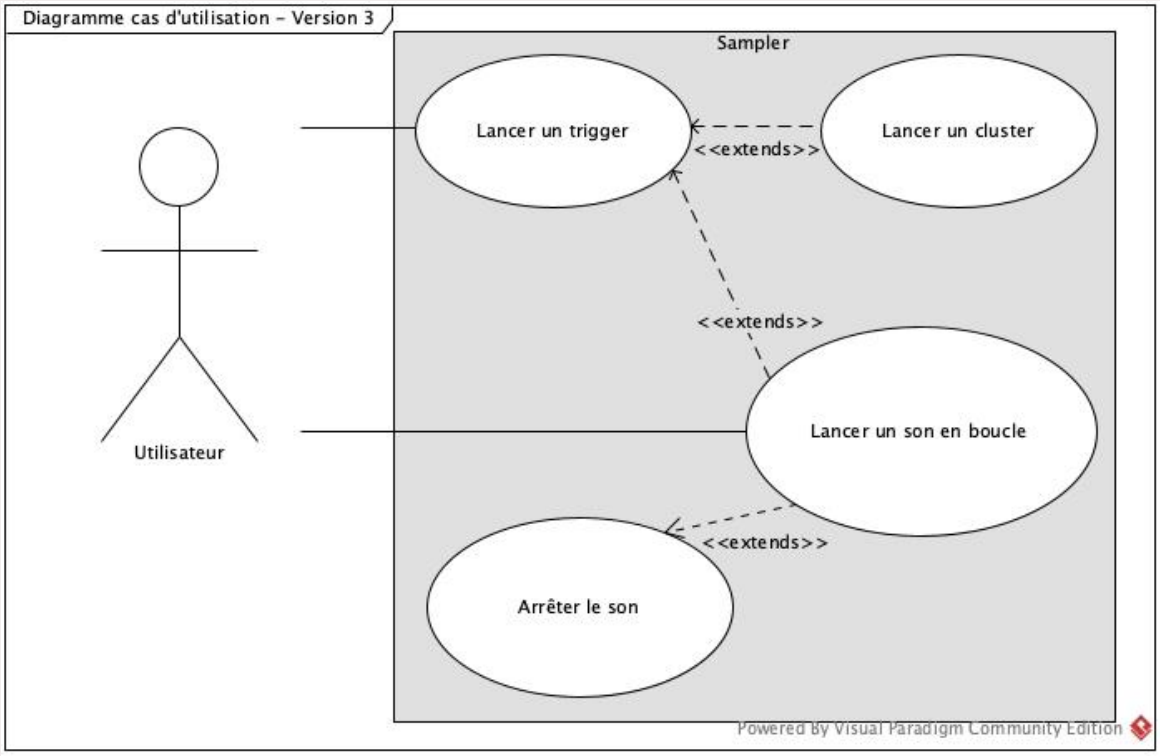
c) Version 3

- Fonctionnalités associées

Aux deux fonctionnalités précédentes, une troisième se rajoute. Elle réside dans le fait de donner la possibilité à l'utilisateur de jouer deux sons simultanément. Pour cela, après avoir appuyé sur un pad, il pourra en activer un second. Une seule boucle est jouée à la fois. Or pour les « triggers », le musicien peut réaliser un cluster en lançant des sons simultanément. Un trigger peut se jouer en même temps qu'une boucle. Cependant, si l'utilisateur lance une boucle et appui sur un nouveau pad pour en lancer une second, cette dernière attend que l'autre soit terminé pour se jouer.

- Scénario

Dans cette version, un troisième cas d'utilisation se rajoute. Il s'agit de « Lancer un second son en boucle ». Nous considérons qu'il s'agit d'un second son à condition qu'un autre ait déjà été lancé auparavant. Ainsi, nous détaillerons seulement ce cas car la description des précédents reste la même.



Description détaillée du cas d'utilisation « Lancer un cluster »

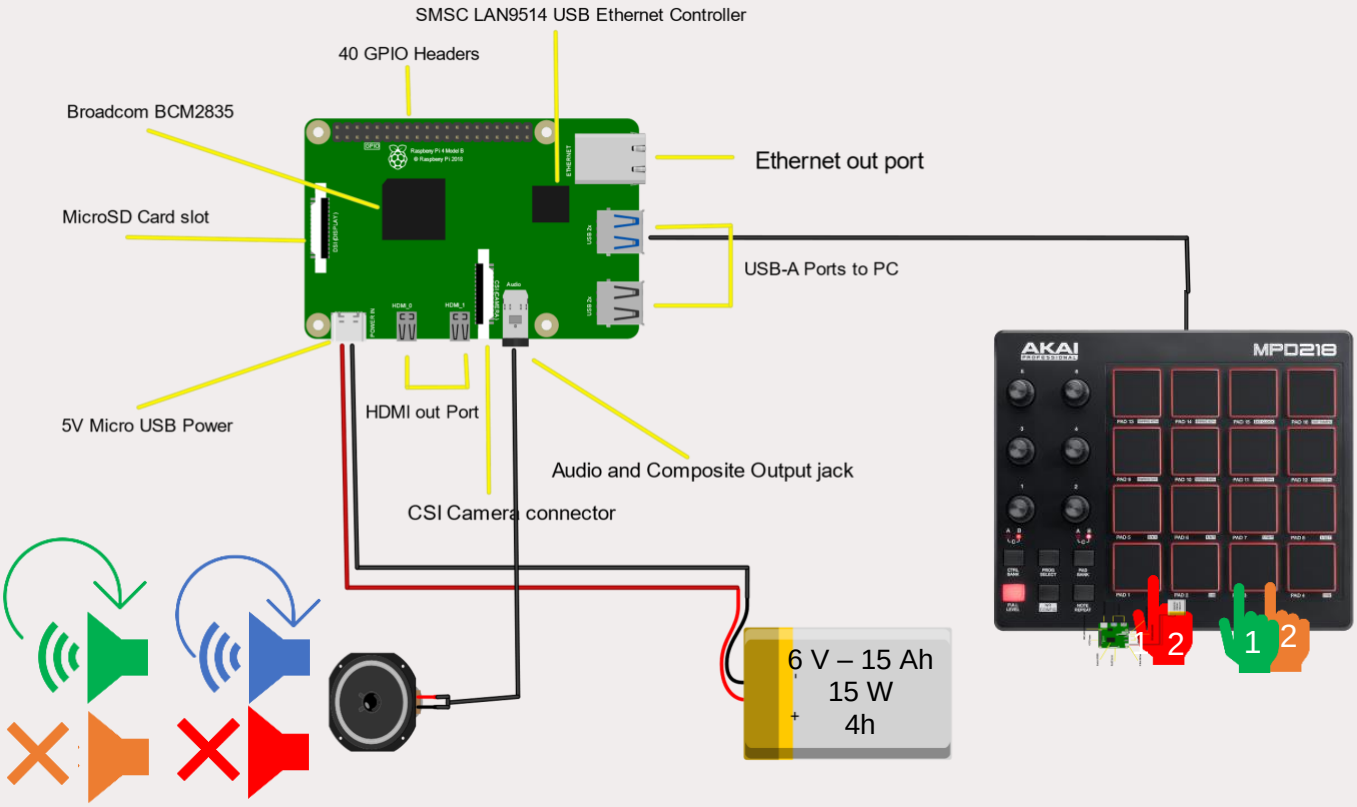
Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Lancer un cluster	
Référence : **Diagramme cas d'utilisation – Version 3**	Version : V3
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : 15/12/2020
Résumé : Permet à l'utilisateur de lancer plusieurs trigger simultanément	
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**	
Références des documents associés : - Définition de la version 3 - Schéma d'utilisation - Diagramme d'analyse/conception (librairies utilisées)	
Description des enchaînements d'interactions du scénario nominal : % précondition : l'utilisateur a mis sous tension le dispositif % - L'utilisateur appuie sur plusieurs pads en même temps. Ceci permet de jouer plusieurs triggers simultanément.	

Description détaillée du cas d'utilisation « Lancer un son en boucle»

Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Lancer un son en boucle	
Référence : **Diagramme cas d'utilisation – Version 3**	Version : V3
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : **14/12 /2020**
Résumé : Permet à l'utilisateur de lancer un son en boucle et un autre simultanément (boucle, trigger, cluster)	
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**	

Références des documents associés : - Définition de la version 3 - Schéma d'utilisation - Diagramme d'analyse/conception (librairies utilisées)
Description des enchaînements d'interactions du scénario nominal : % précondition : l'utilisateur a mis sous tension le dispositif et a lancé un son en boucle % - L'utilisateur appui sur un pad. Ainsi, le son se joue en boucle, c'est-à-dire qu'une fois qu'il se termine, il se relance automatiquement sans aucune action utilisateur.
Liste des enchaînements alternatifs ou d'exception
- L'utilisateur décide de lancer un second son simultanément. Il peut déclencher un trigger ou un cluster lorsque la boucle se joue. - L'utilisateur décide de lancer une seconde boucle. Cependant, la première n'est pas terminée. Ainsi, la seconde boucle est jouée une fois la première terminée.

- Schéma



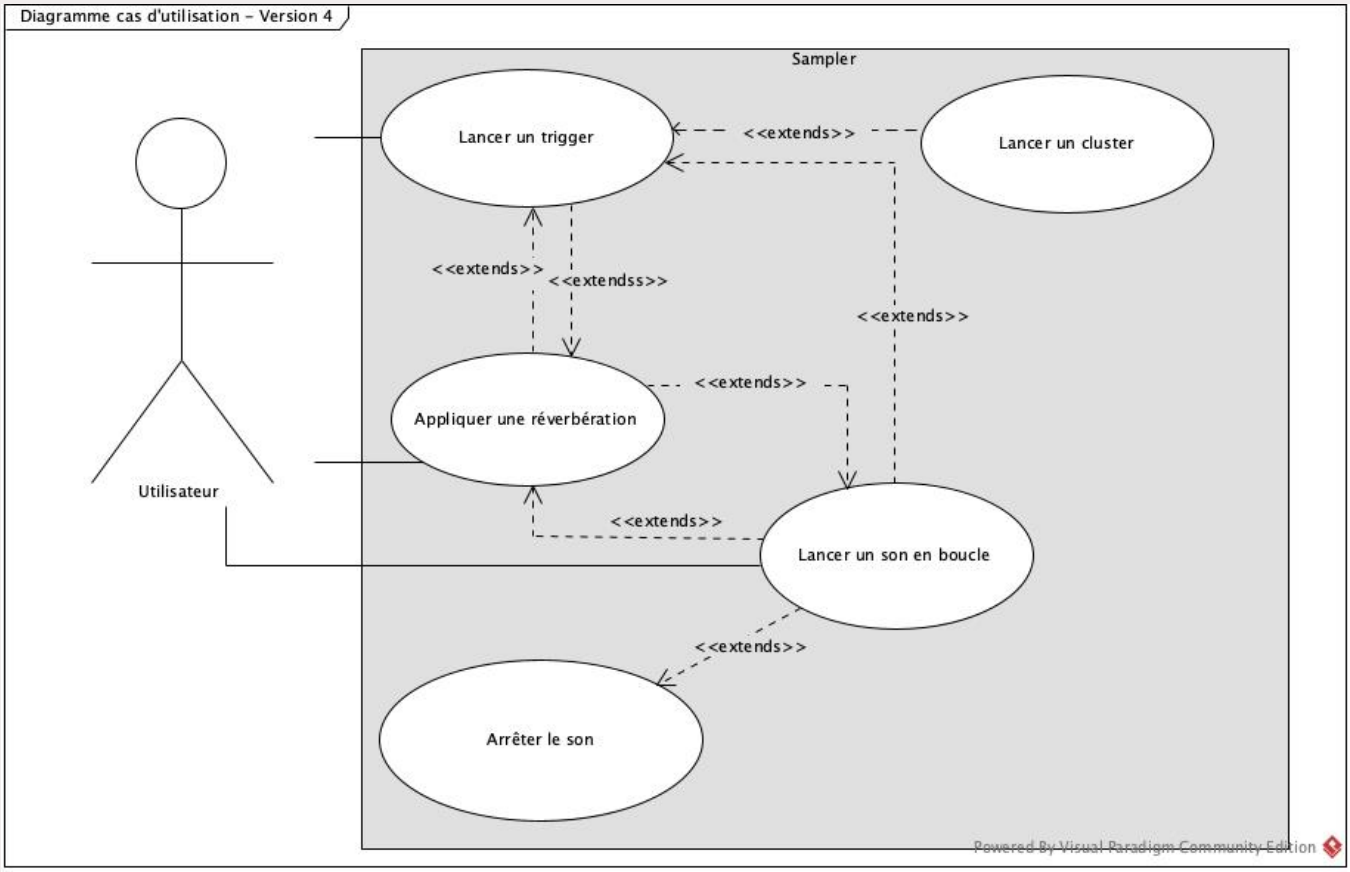
d) Version 4

- Fonctionnalités associées

Afin d'améliorer le produit, ajoutons une fonction permettant d'utiliser les potentiomètres. Ceux-ci vont permettre d'agir sur la partie acoustique du son. Une réverbération wet/dry (mouillé/sec en français) sera appliquée au son. Comme expliqué précédemment, elle sera appliquée selon des valeurs envoyées par les potentiomètres. Pour rappel, lorsque le potentiomètre tend vers 255, l'acoustique du son joué prend de plus en plus un effet mouillé et inversement. Cette réverbération se fait en sortie du son.

- Scénario

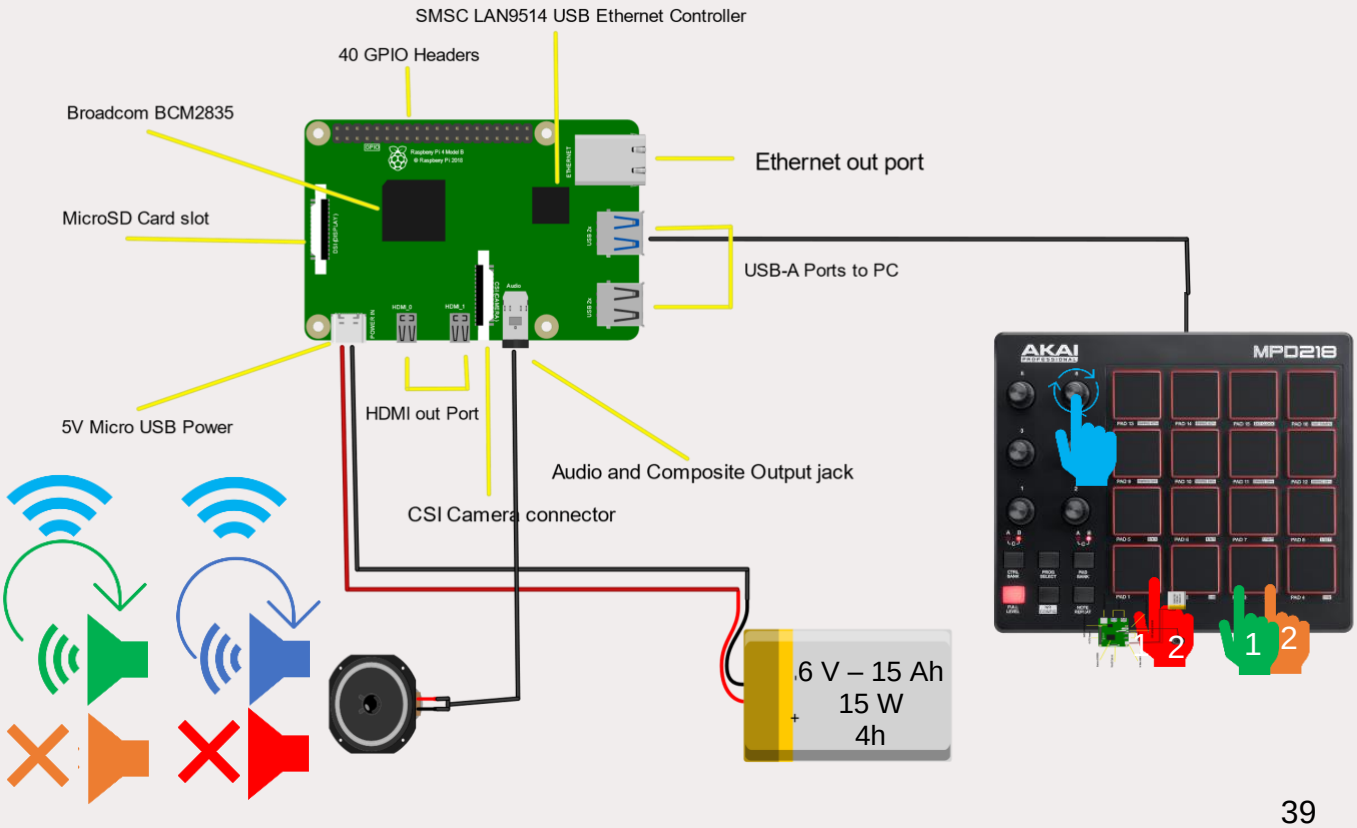
Pour cette nouvelle version, un quatrième cas d'utilisation prend place. Il s'agit d' «Appliquer la réverbération». Cette fonctionnalité peut être utilisée si et seulement si au moins un son est lancé.



Description détaillée du cas d'utilisation « Appliquer une réverbération »

Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Appliquer la réverbération	
Référence : **Diagramme cas d'utilisation – Version 4**	Version : V4
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : 15/12/2020
Résumé : Permet à l'utilisateur d'appliquer une réverbération à un son.	
Références des documents associés : - Définition de la version 4 - Schéma d'utilisation - Diagramme d'analyse/conception (librairies utilisées)	
Description des enchaînements d'interactions du scénario nominal : % pré condition : l'utilisateur a mis sous tension le dispositif % - L'utilisateur fait varier la résistance d'un potentiomètre afin d'appliquer la réverbération se traduisant par un phénomène de raisonnante en sortie. La réverbération est applicable après ou avant d'avoir lancer n'importe quel son.	

- Schéma



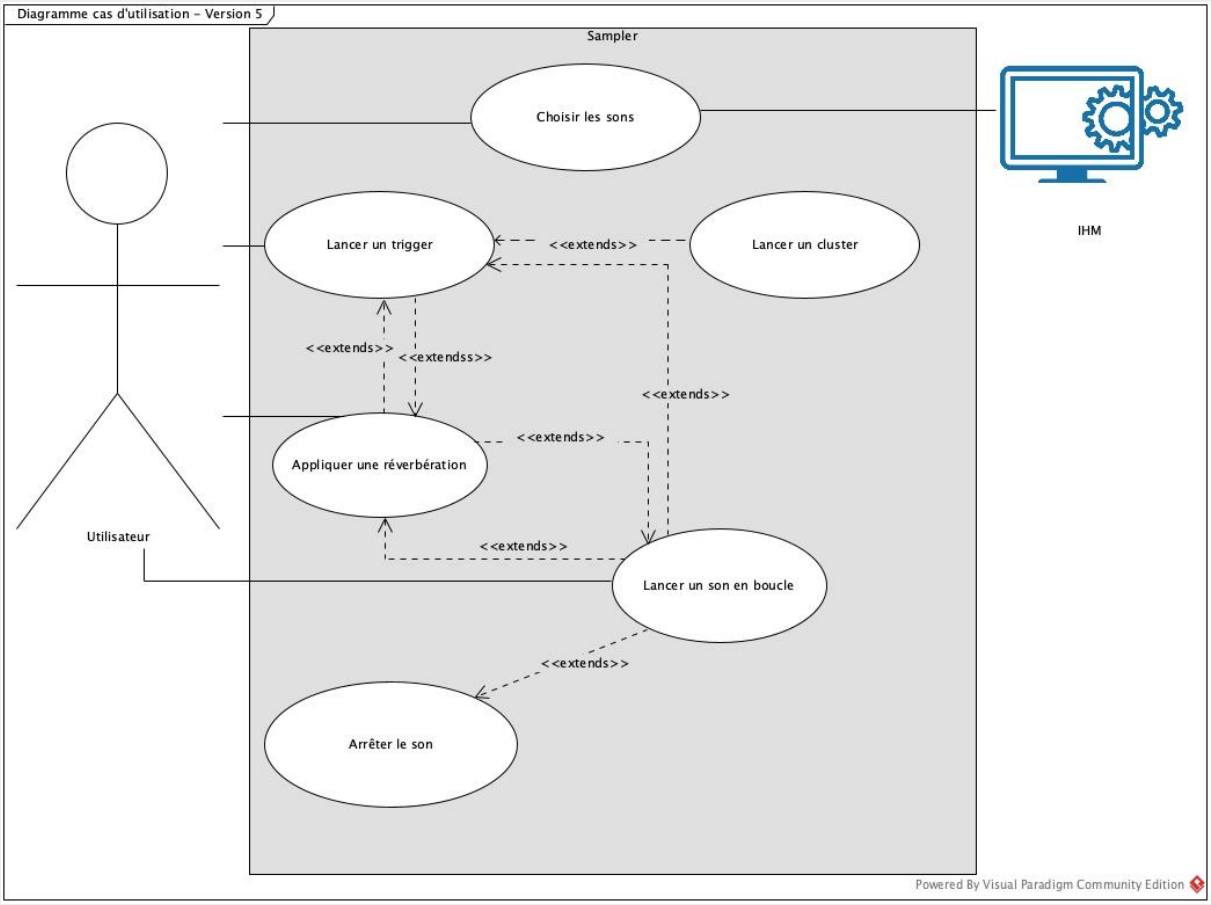
e) Version 5

- Fonctionnalités associées

Cette fonctionnalité concerne un deuxième livrable pour le produit. Il s'agit d'une interface logicielle afin de pouvoir configurer le contrôleur dans l'objectif d'assigner librement chaque pad à un son.

- Scénario

En ce qui concerne cette cinquième version, l'utilisateur aura la possibilité de configurer le contrôleur avec l'interface homme-machine prévue à cet effet.

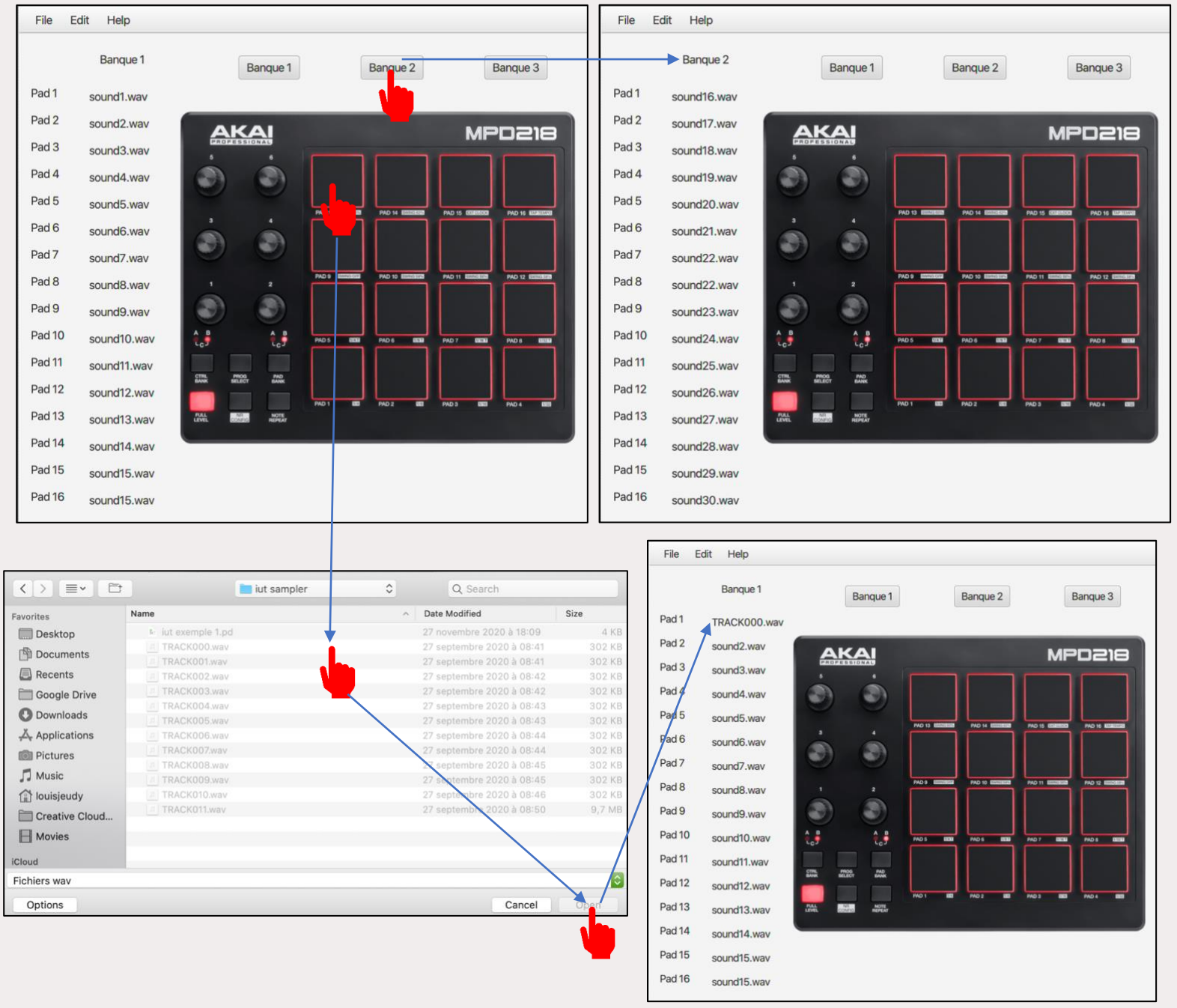


Description détaillée du cas d'utilisation « Choisir les sons »

Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Choisir les sons	
Référence : **Diagramme cas d'utilisation – Version 5**	Version : V5
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : 15/12/2020
Résumé : Permet à l'utilisateur de configurer chacun des pads selon la banque sélectionnée sur un logiciel.	
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**	
Références des documents associés : - Définition de la version 5 - IHM - Diagramme d'analyse/conception (librairies utilisées)	
Description des enchaînements d'interactions du scénario nominal : % précondition : l'utilisateur a mis sous tension le dispositif en le branchant à un écran externe. De plus il a lancé le logiciel % - Grâce à l'interface, l'utilisateur sélectionne la banque pour laquelle il veut configurer les pads. Pour chacun d'entre eux, lorsqu'il clique sur pad à configurer, un explorateur de fichier s'ouvre et le nouveau son à associer peut être sélectionné.	

- Maquettes IHM

L'IHM se décompose en deux parties utilisables. La première est le choix de la banque pour laquelle nous voulons configurer les pads. La seconde, permet d'assigner à chaque pad un son que l'on sélectionne et qui permet d'ouvrir un explorateur de fichier.

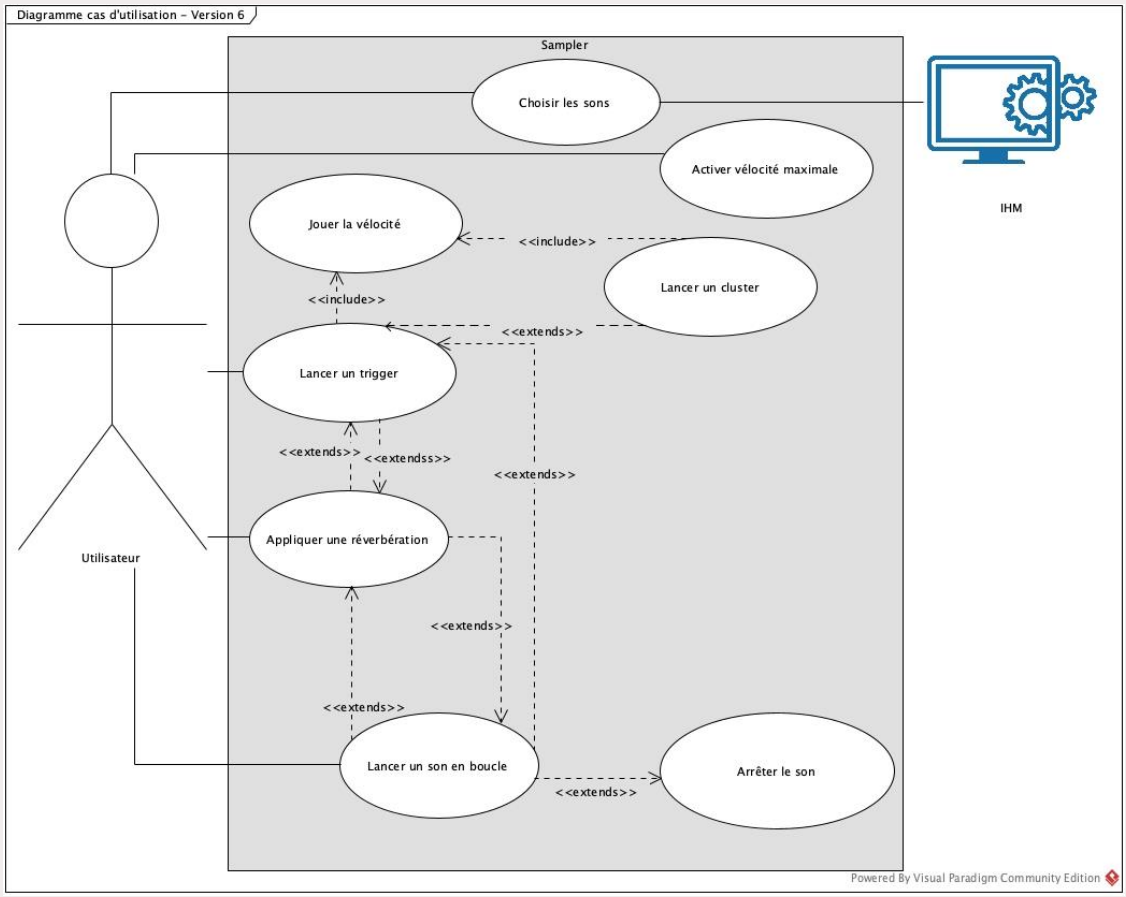


f) Version 6

- Fonctionnalités associées

Cette fonctionnalité a pour but d'offrir au client, la possibilité de jouer plus ou moins fort un trigger grâce à la vélocité (pression) donnée au pad. Pour rappel, la vélocité est caractérisée par une valeur allant de 0 à 127.

- Scénario



Description détaillée du cas d'utilisation « Jouer la vélocité »

Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Jouer la vélocité	
Référence : **Diagramme cas d'utilisation – Version 6**	Version : V6
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : 16/12/2020

Résumé : Permet à l'utilisateur de jouer le son avec un volume différent selon la pression émise sur le pad.
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**
Références des documents associés : Définition de la version 6 - Schéma - Diagramme d'analyse/conception (librairies utilisées)
Description des enchaînements d'interactions du scénario nominal : % pré condition : l'utilisateur a mis sous tension le dispositif % - L'utilisateur applique une pression lorsqu'il appuie sur un pad. En fonction de la force de celle-ci, le volume du son joué sera plus ou moins fort.
Description des scénarios alternatifs : - L'utilisateur décide de jouer la vélocité de manière maximale (Voir cas utilisation « Activer vélocité maximale ».

Description détaillée du cas d'utilisation « Activer vélocité maximale »

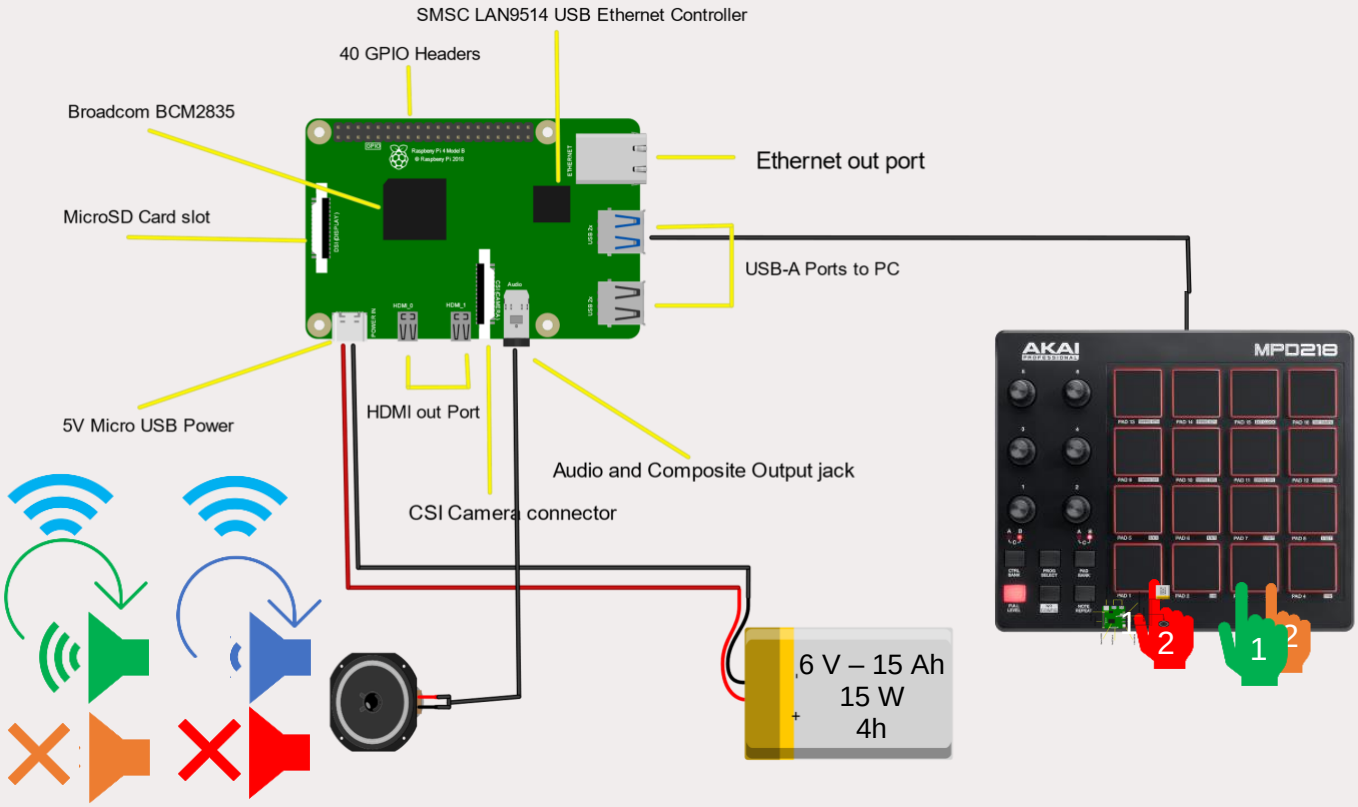
Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Activer vélocité maximale	
Référence : **Diagramme cas d'utilisation – Version 6**	Version : V6
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : 16/12/2020
Résumé : Pour n'importe quelle pression apportée au pad, le son sera joué au volume le plus fort.	
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**	
Références des documents associés : Définition de la version 6 - Schéma - Diagramme d'analyse/conception (librairies utilisées)	

Description des enchaînements d'interactions du scénario nominal :

% pré condition : l'utilisateur a mis sous tension le dispositif %

- L'utilisateur applique appui sur le bouton « Full level » qui permet par défaut de jouer les sons avec une vélocité maximale. Les sons seront donc jouer au volume le plus fort possible (127).

- Schéma

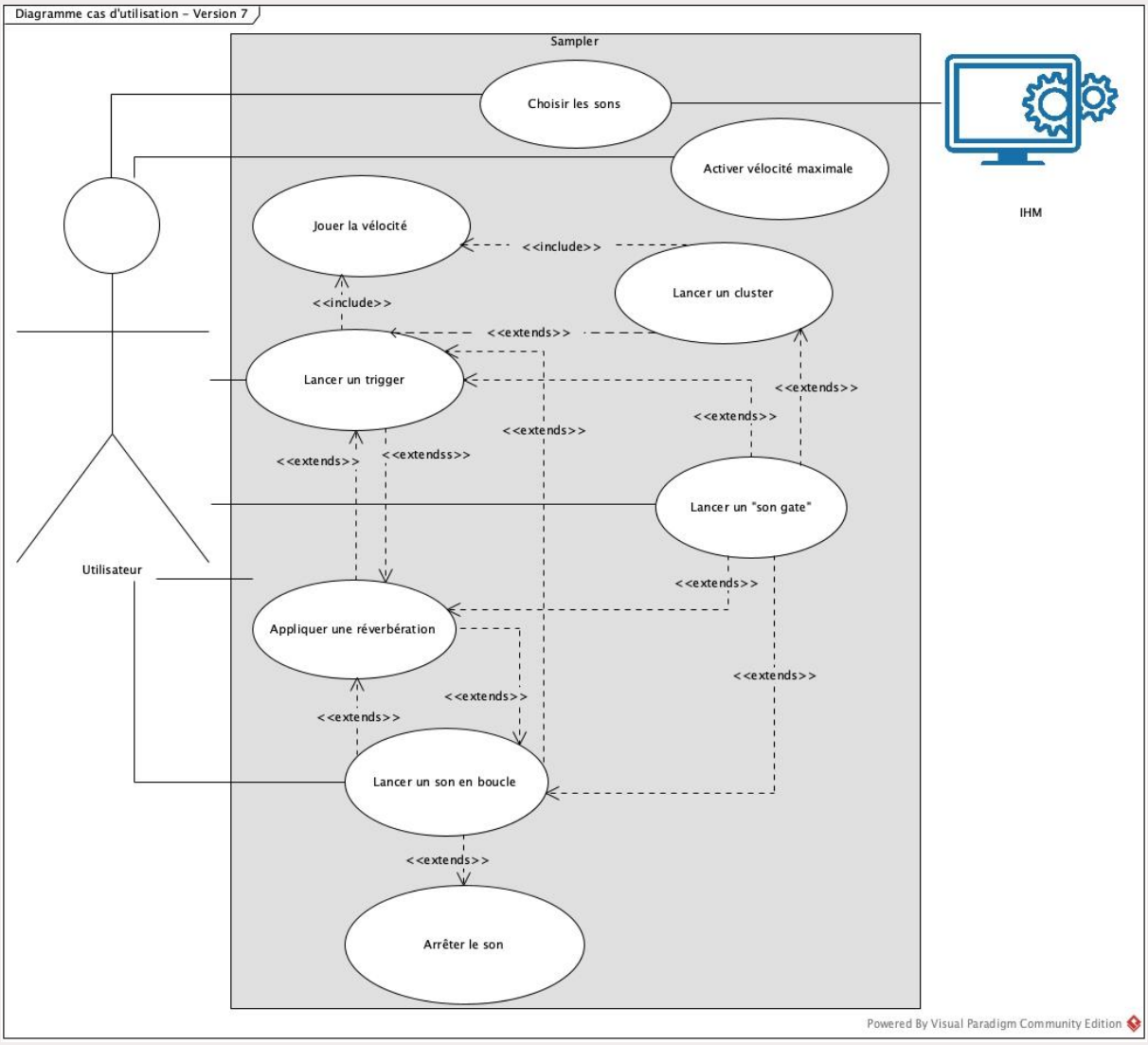


g) Version 7

- Fonctionnalités associées

Pour cette fonctionnalité, l'utilisateur pourra pour un son, le jouer pendant une durée voulue. Celle-ci correspond au temps durant lequel il reste appuyé sur un pad en le maintenant. Ce type d'action est appelé un « son gate ».

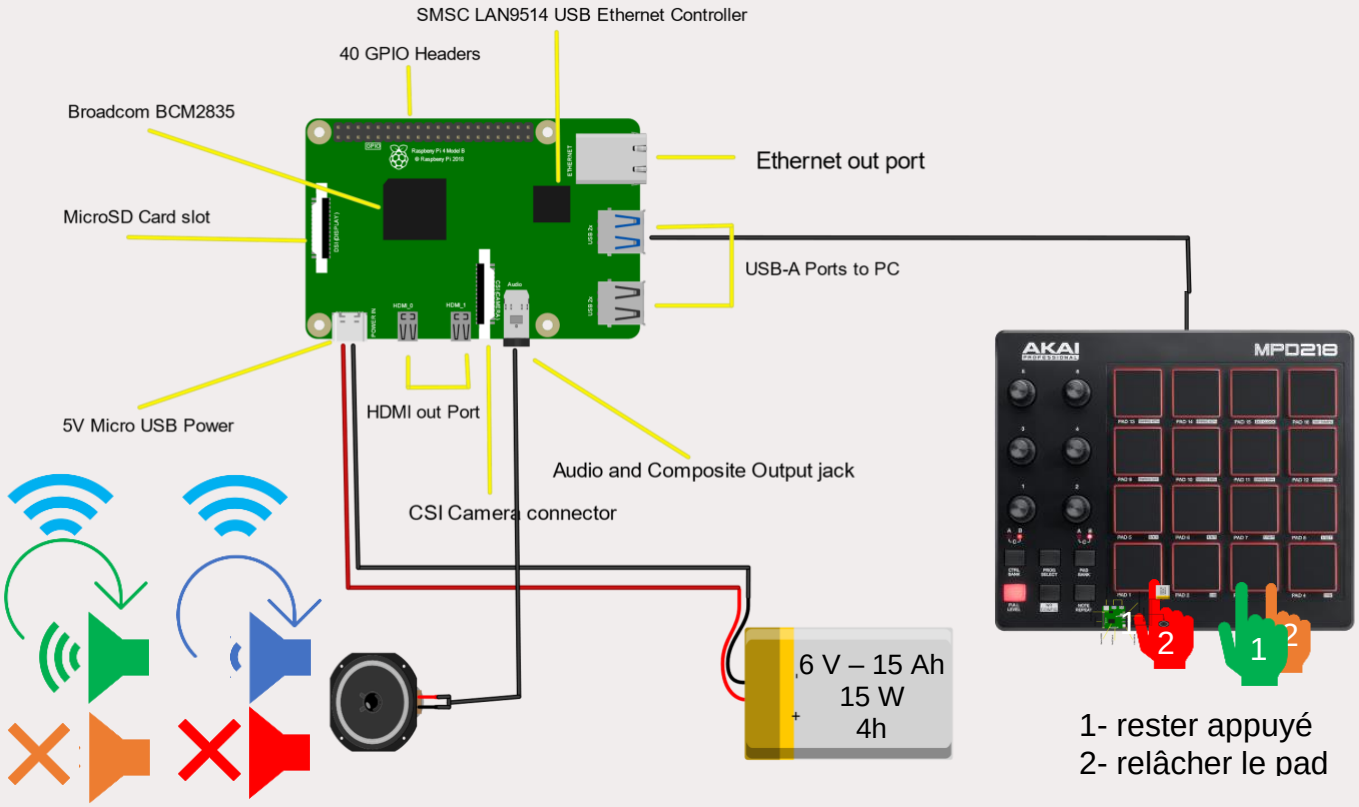
- Scénario



Description détaillée du cas d'utilisation « Lancer un son gate »

Cas d'utilisation - description simplifiée	
PROJET : PTS3 : Conception d'un sampler portable	
Cas d'utilisation : Lancer un « son gate »	
Référence : **Diagramme cas d'utilisation – Version 7**	Version : V7
Auteur : **Louis Jeudy, Romain Durieux, Elyas Abbasi, Alexandre Bareau**	Date : 15/01/2021
Résumé : Permet à l'utilisateur de jouer le son durant un temps voulu.	
- Acteur déclencheur : Utilisateur **- Acteur(s) participant(s) : Aucun**	
Références des documents associés : Définition de la version 7 - Schéma - Diagramme d'analyse/conception (librairies utilisées)	
Description des enchaînements d'interactions du scénario nominal : % pré condition : l'utilisateur a mis sous tension le dispositif % - L'utilisateur maintient une pression sur un pad pour lancer un son en continu. Dès qu'il ne maintient plus le pad, le son s'arrête.	
Description des enchaînements d'interactions du scénario nominal : - L'utilisateur peut lancer un son gate simultanément avec un trigger. - L'utilisateur lance un son gate pendant qu'une boucle se joue. - L'utilisateur décide de jouer un son gate combiné à un cluster. - L'utilisateur applique une réverbération avant ou pendant le lancement d'un son gate.	

- Schéma


SMSC LAN9514 USB Ethernet Controller
40 GPIO Headers
Broadcom BCM2835
Ethernet out port
MicroSD Card slot
USB-A Ports to PC
5V Micro USB Power
HDMI out Port
Audio and Composite Output jack
CSI Camera connector
6 V – 15 Ah
15 W
4h
AKAI
MPD218
1
2
1
2
1- rester appuyé
2- relâcher le pad

VIII. Planning prévisionnel semestre 4

Date	Tâches
Semaine 4 (25/01/2021 au 31/01/2021	Étude de faisabilité et prototypes
Semaine 5 (01/02/2021 au 07/02/2021	Prototypes et confirmation des choix techniques
Semaine 6 (08/02/2021 au 14/02/2021)	Version 1 réalisée et documentée
Semaine 7 (15/02/2021 au 21/02/2021)	Version 2 réalisée et documentée
Semaine 8 (22/02/2021 au 28/02/2021)	Version 3 réalisée et documentée
Semaine 9 (01/03/2021 au 07/01/2021)	Version 4 réalisée et documentée
Semaine 10 (08/02/2021 au 14/02/2021)	Version 5 réalisée et documentée
Semaine 11 (15/02/2021 au 21/02/2021)	Version 6 réalisée et documentée
Semaine 12 (22/02/2021 au 28/02/2021)	Version 7 réalisée et documentée
Semaine 13 (29/03/2021 au 04/03/2021)	Formation de l'utilisateur
Semaine 14 (05/04/2021 au 09/04/2021)	**Soutenance**

IX. Prototypage

a) Étude de faisabilité technique

Dès à présent, étudions le degré de faisabilité de notre projet. Celui-ci va nous permettre d'énumérer les points faisables ou non faisables selon nos capacités. Il va nous permettre aussi la confirmation des choix techniques parmi ceux énumérés précédemment. Nous allons ainsi pouvoir analyser les risques techniques dans la réalisation de certaines fonctionnalités. Cette étude va donc être réalisée sur la base de petit prototypes, c'est-à-dire prendre les technologies en main, réaliser des tests, et essayer de comprendre le fonctionnement.

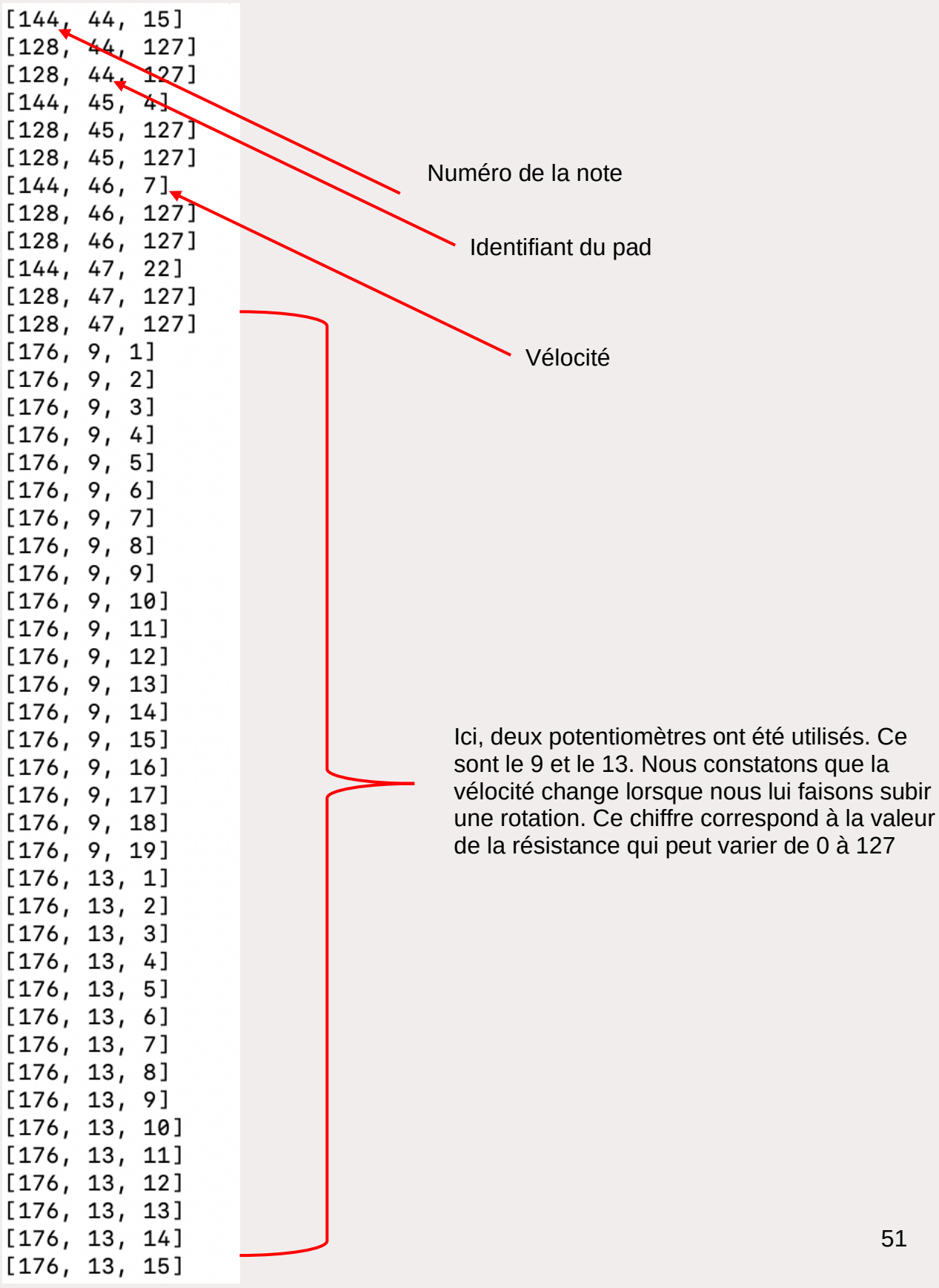
Essai n°1 : RtMidi (python) & Pygame

Considérons une première réalisation dont l'objectif a été de prendre en main les technologies, c'est-à-dire les librairies python à savoir RtMidi et Pygame.
Ce prototype a pour objectif de lancer un trigger à l'appui d'un pad, lancer un son en boucle, jouer un cluster. Il est aussi possible de jouer un trigger et un cluster lorsqu'une boucle est en cours.

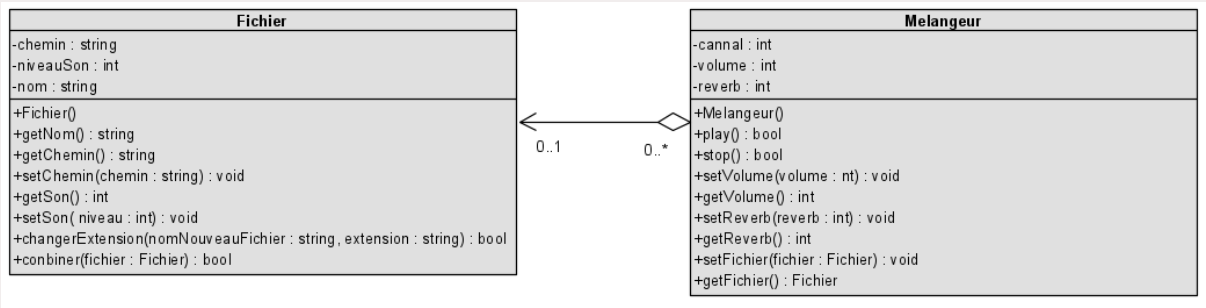
```
1   #!/usr/bin/python3.4
2
3   # Import et chargement des modules requis
4   import rtmidi_python as rtmidi
5   from pygame import mixer
6   import pygame
7
8   midi_in = rtmidi.MidiIn()
9   midi_in.open_port(0) # ouverture du port MIDI.
10
11  def pad44():
12      mixer.music.load('/Users/louisjeudy/Desktop/TRACK000.wav')
13      mixer.music.play(-1, 0.0, 0)
14
15  def pad45():
16      pygame.mixer.Channel(0).play(pygame.mixer.Sound('/Users/louisjeudy/Desktop/TRACK008.wav'))
17
18  def pad46():
19      pygame.mixer.Channel(1).play(pygame.mixer.Sound('/Users/louisjeudy/Desktop/TRACK011.wav'))
20
21  def pad47():
22      pygame.mixer.Channel(2).play(pygame.mixer.Sound('/Users/louisjeudy/Desktop/TRACK011.wav'))
23
24  while True:
25      mixer.init()
26      message, delta_time = midi_in.get_message()
27      if message:
28          print message
29          touche = message[1]
30          velocite = message[2]
31          if touche == 44 and velocite == 127:
32              pad44()
33              print message
34          if touche == 45 and velocite == 127:
35              pad45()
36              print message
37          if touche == 46 and velocite == 127:
38              pad46()
39              print message
40          if touche == 47 and velocite == 127:
41              pad47()
42              print message
```

Dans ce programme 4 fonctions sont définies. Elles représentent chacune un pad du Akai. Elle permettent de jouer un son chargé par le module Pygame à travers le chemin absolu désignant l'emplacement du fichier (ligne 12-16-19-22). Dans le cas présent, le son est joué à une vélocité maximale. Ici, chaque numéro de pad est récupéré par la fonction midi_in.get_message(). Celle-ci renvoyant un array de trois

données (vues précédemment), nous isolons le numéro du pad et la vélocité (29-30). Ce système est aussi valable pour les potentiomètres. Ainsi, des tests peuvent se faire sur ces derniers afin d'effectuer des actions voulues. A présent, du point de vue console (Terminal) analysons l'affichage qui est réalisé. Dans le terminal, lorsqu'un pad est appuyé, son message MIDI s'affiche. En voici un exemple :



Essai n°2 : SoX



Ce diagramme de classe modélise la vision possible que nous aurons de notre programme en utilisant l'idée de la bibliothèque "SoX". Nous aurons à jouer un fichier son qui peut être amené à changer de chemin ou de nom, plusieurs méthodes lui seront donc attribuées. Ainsi qu'une classe « mélangeur » qui permettra de changer le son de différentes façons.