

Projet S3-S4 pad d'arcade multi-console

Cahier des charges

13/01/2015

IUT Valence département Informatique

Fakri Tis, Charles Morel, Thomas Dubois, Hadrien Mittoux, Joachim Woerly-Moussier

Tuteur : Sébastien Jean

Table des matières

A. Recueil des besoins	4
1. Objectif	4
2. Architecture fonctionnelle obligatoire	5
2.1. Être compatible avec un maximum de machines	5
2.2. Être configurable de manière autonome ou non.....	5
2.3. Le pad devra être réalisable facilement par le client	5
3. Architecture fonctionnelle optionnelle	6
3.1. Être réversible droitier/gaucher.....	6
3.2. Avoir été travaillé sur un plan ergonomique.....	6
3.3. Gérer des profils d'association de touche.....	6
B. Cahier des Charges	7
1. Présentation du projet	7
1.1. Objectif du projet	7
1.2. Périmètre fonctionnel du projet	9
1.3. Cadre technique	11
2. Les fonctions.....	13
2.1. Architecture fonctionnelle	13
2.2. Description des principales fonctions	16
2.3. Documents supplémentaires	28
C. Analyse de différentes connectiques	29
1. La connexion à l'Arduino	29
1.1. La partie interne au contrôleur.....	29
1.2. La communication entre machine et contrôleur.....	30
2. Atari 2600/SMS/Mega Drive/CPC	31
2.1. Connecteur	31
2.2. Protocole	31
3. NES/SNES.....	32
D. Annexes.....	33
1. Vocabulaire.....	33
2. Table d'illustration.....	35

A. Recueil des besoins



Figure 1 - Pad Classique

1. Objectif

L'objectif de cette collaboration est de créer un contrôleur de jeu, type arcade, constitué d'un joystick, de 6 boutons d'actions et de 2 boutons d'options (cf. fig.1). Il doit permettre de jouer sur différents types de machines. Il faut pouvoir connecter le contrôleur à une machine et s'en servir de manette. Ce projet a pour but de limiter le nombre de manettes d'un utilisateur. En effet le nombre de manettes se multiplie avec le nombre de machines. Ce stick permet d'avoir une manette compatible avec plusieurs machines (cf. fig2). La difficulté étant principalement de traduire les signaux des manettes pour qu'ils soient compatibles avec toutes les machines. Mais aussi d'adapter la connectique.

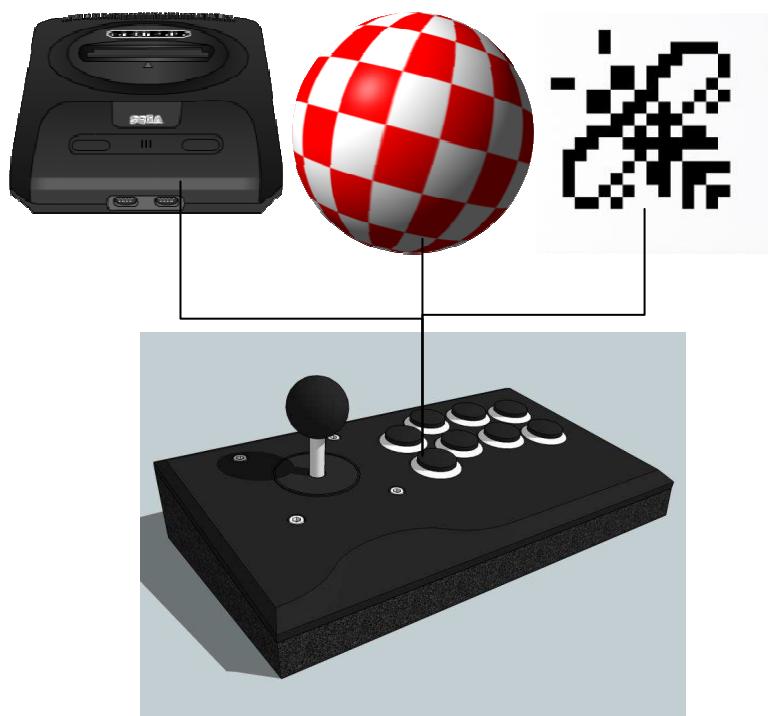


Figure 2 - Schéma explicatif de l'utilité de la manette

2. Architecture fonctionnelle obligatoire

Toutes les conditions qui suivent devront être remplies par le contrôleur.

2.1. Être compatible avec un maximum de machines

Les priorités étant aux machines suivantes : CPC, Master System, Megadrive, Atari 2600.

2.2. Être configurable de manière autonome ou non

La configuration définit le fait d'adapter les signaux envoyés par le joystick au protocole utilisé par les machines auxquelles il est branché. Par exemple, la position « haut » du joystick devra être interprétée comme si l'on appuyait sur la touche « haut » d'une manette classique. Cette configuration doit pouvoir se faire de manière automatique au branchement à la console ou bien de manière manuelle par un système de sélection de machine. Les associations d'un bouton d'une manette à un de ceux du contrôleur devront potentiellement être paramétrables via une application PC ou Smartphone. Cette configuration devra être persistante.

2.3. Le pad devra être réalisable facilement par le client

Il devra avoir un coût peu élevé. Les composants électroniques (boutons, sticks, processeurs, etc) sont disponibles dans des magasins spécialisés à faible coût. Le boîtier doit pouvoir être réalisable dans un fablab¹. Les logiciels devront être libres de droits.

¹Un fab lab (contraction de l'anglais fabrication laboratory, « laboratoire de fabrication ») est un lieu ouvert au public où il est mis à sa disposition toutes sortes d'outils, notamment des machines-outils pilotées par ordinateur, pour la conception et la réalisation d'objets.

3. Architecture fonctionnelle optionnelle

Toutes les conditions qui suivent sont des fonctions supplémentaires que le contrôleur pourra probablement remplir.

3.1. Être réversible droitier/gaucher

Le boîtier pourrait être utilisable quel que soit le sens dans lequel on l'utilise.



Figure 3 – Réversibilité Gaucher/Droitier

3.2. Avoir été travaillé sur un plan ergonomique

Un travail qui permettrait d'ajouter des fonctions ergonomiques tel qu'un système d'inclinaison pour un meilleur confort de jeu ou un emplacement de rangement des câbles pour éviter un surnombre de boîtes de rangement.

3.3. Gérer des profils d'association de touche

Le client pourrait paramétriser de différentes façons les touches sur l'application de configuration et enregistrer ses configurations dans des profils. Il pourrait choisir ensuite le profil directement sur le joystick. Le contrôleur pourrait aussi (via le paramétrage fait préalablement ou via un bouton supplémentaire) simuler un appui répété et rapide d'un bouton (auto-fire).

B. Cahier des Charges

1. Présentation du projet



Figure 4 - Pad Classique

1.1. Objectif du projet

L'objectif de cette collaboration est de créer un contrôleur de jeu (JoyPad), de type arcade, constitué d'un joystick, de 6 boutons d'actions et de 2 boutons d'option (cf. fig.1). Il doit permettre de jouer sur différents types de machines. Il faut pouvoir connecter le contrôleur à une machine et s'en servir de manette. Ce projet a pour but de limiter le nombre de manettes d'un utilisateur. En effet le nombre de manettes se multiplie avec le nombre de machines. Ce stick permet d'avoir une manette compatible avec plusieurs machines (cf. fig2). Les difficultés étant principalement de traduire les signaux des manettes pour qu'ils soient compatibles avec toutes les machines, mais aussi d'adapter la connectique du joystick au différentes console.

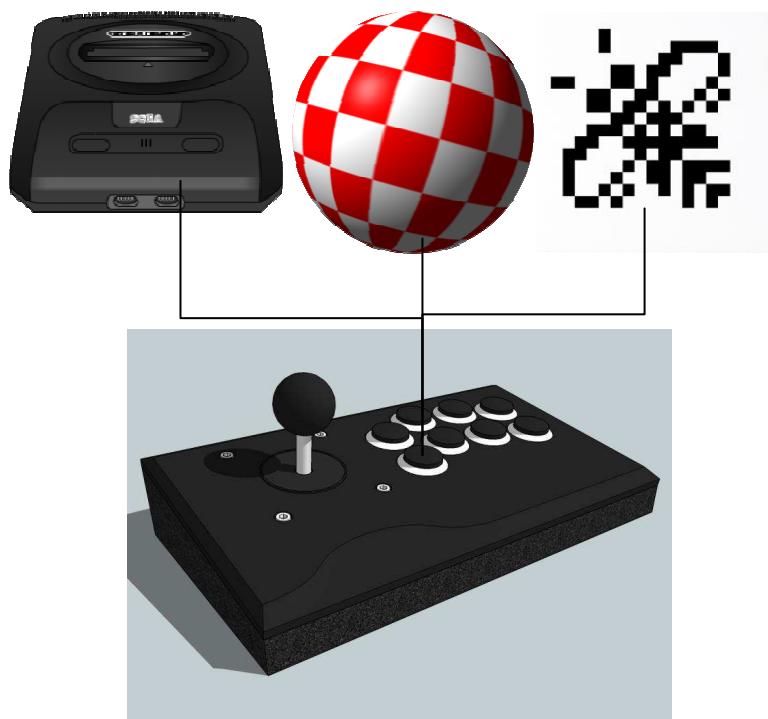


Figure 5 - Schéma explicatif de l'utilité de la manette

Le pad pourra être adaptable droitier-gaucher :



Figure 6 - Réversibilité Gaucher/Droitier

1.2. Périmètre fonctionnel du projet

Nous avons décidé de créer deux cas possibles : un cas de réalisation « réaliste » et un cas de réalisation « optimal ».

1.2.1. Diagramme des cas d'utilisation réaliste

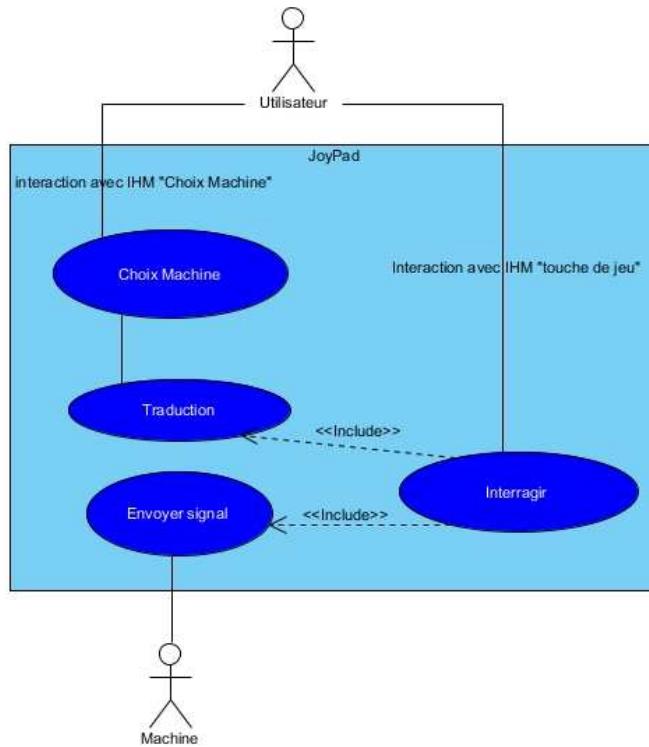


Figure 7 - Diagramme des cas d'utilisation réaliste

Dans le cas présent, le dispositif sera composé du seul JoyPad. La configuration sera totalement statique : aucun changement ne pourra être effectué sur le JoyPad sans devoir le reprogrammer.

Dans ce cas, deux acteurs seront disponibles :

- l'utilisateur du JoyPad, c'est-à-dire le joueur.
- la machine sur laquelle sera connecté le JoyPad.

L'utilisateur aura la possibilité de faire deux actions :

- appuyer sur un bouton correspondant au choix de la machine; cela aura pour effet de changer les signaux qui seront envoyés à cette dernière (et donc de s'adapter au protocole de la machine réceptrice).
- appuyer sur un bouton de jeu ; cela aura pour effet de déclencher une conversion de l'appui de la touche vers un signal qui sera ensuite envoyé à la machine.

La machine n'a aucun effet sur le JoyPad, elle ne fait que recevoir les signaux. Elle les traitera ensuite en toute autonomie, indépendamment du JoyPad.

1.2.2. Diagramme des Use Cases optimal

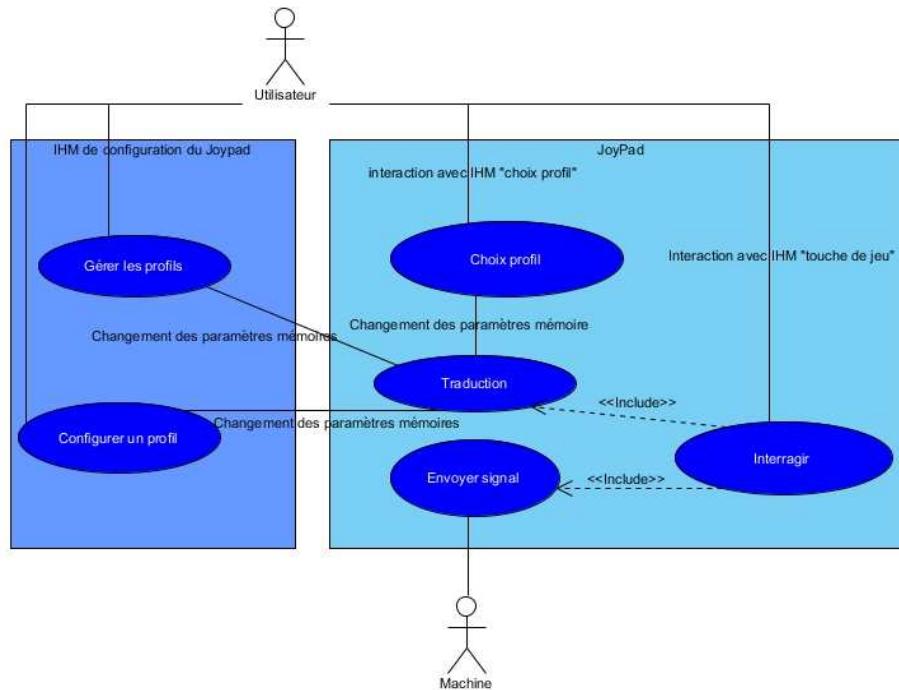


Figure 8 - Diagramme des cas d'utilisations optimal

Ce cas reprend les grandes lignes du cas réaliste. Cependant, au lieu d'être « figé » comme ce dernier, il permet de configurer le JoyPad autour de profils. On ajoute alors un nouveau dispositif qui est une IHM de configuration de profil.

Les acteurs sont identiques à ceux du cas précédent. Les principales différences découlent du système de profil, en effet cela implique deux actions supplémentaires liées à l'utilisateur :

- la gestion des profils, c'est à dire la création d'un nouveau profil, la copie d'un profil de la mémoire de l'IHM de configuration vers le JoyPad (ou inversement), le transfert d'un profil ou la suppression.
- La configuration d'un profil, c'est à dire éditer l'association de touches avec les signaux envoyés à la machine (ainsi que les gestions de macro, combo et auto-fire)

Pour la suite de ce cahier des charges, nous décrirons avant tout ce cas.

1.3. Cadre technique

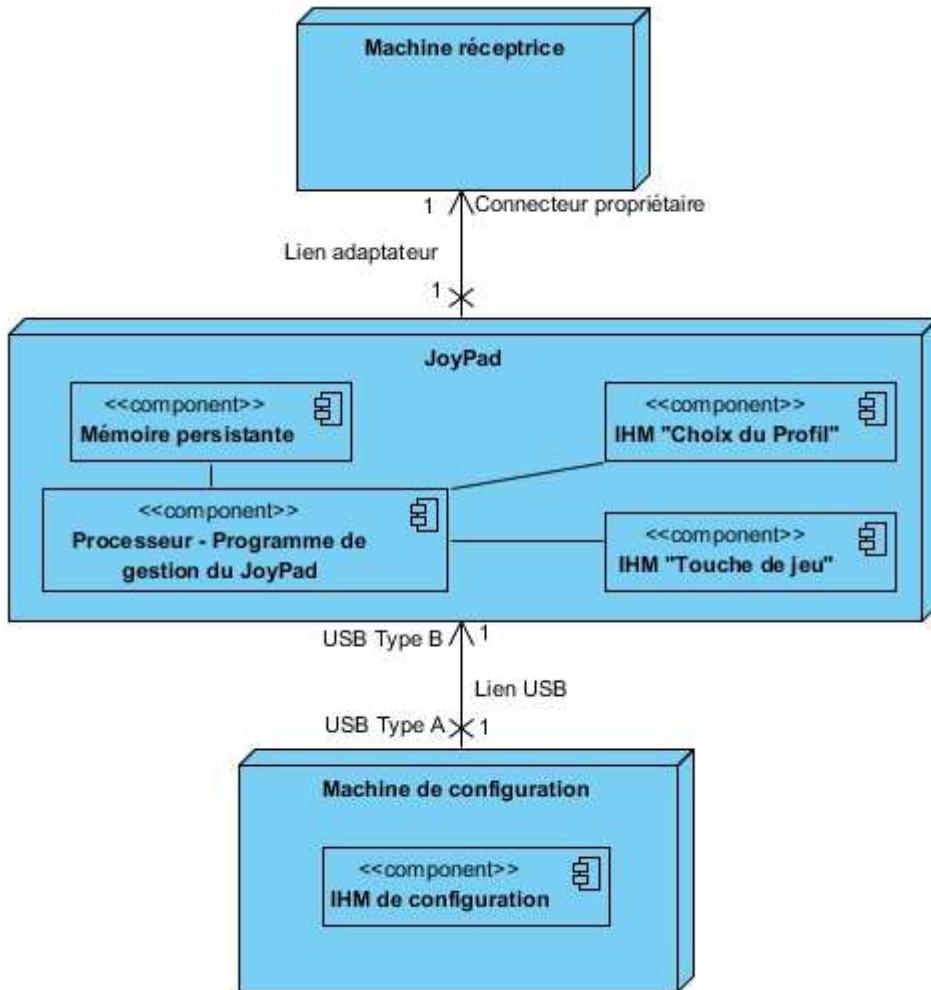


Figure 9 - Diagramme d'architecture

Le premier dispositif est la machine réceptrice. Elle n'est pas fixée à une machine particulière : à l'aide d'un adaptateur, le JoyPad sera compatible au connecteur propriétaire de la machine. De plus, le protocole d'envoi d'informations de la part du JoyPad sera adapté à la machine à l'aide d'un choix fait directement par l'utilisateur sur le JoyPad. Il sera impossible d'utiliser le JoyPad avec une machine lors de sa configuration.

Le second dispositif est le JoyPad. Il s'agit du cœur du système. Il est composé dans un premier temps d'une IHM « touches de jeu » de 8 touches de jeu (dont un START et un SELECT) avec un joystick. L'utilisation de cette IHM provoquera l'envoi de signaux à la machine. Elle sera générée à l'aide d'un programme de gestion du JoyPad reposant sur une carte Arduino ainsi que d'une mémoire persistante. Enfin, une IHM « Choix de profil », composée de deux boutons (±) permettra de choisir les types de signaux envoyés à la machine en fonction de la touche enfoncee.

Enfin, le dernier dispositif est une machine de configuration, certainement un PC. Il sera relié au JoyPad à l'aide d'un simple câble USB. Il permettra de configurer le JoyPad à l'aide d'une IHM permettant la gestion et la modification des profils.

Le système sera donc divisé en deux programmes distincts : le premier est destiné au fonctionnement du JoyPad et est donc intégré dans l'Arduino, le second est destiné au dispositif de configuration et servira à mettre au point les divers profils utilisé par le JoyPad.

2. Les fonctions

2.1. Architecture fonctionnelle

Comme évoqué précédemment, le dispositif est divisé en deux programmes dont les modules sont représentés sur le schéma ci-dessous :

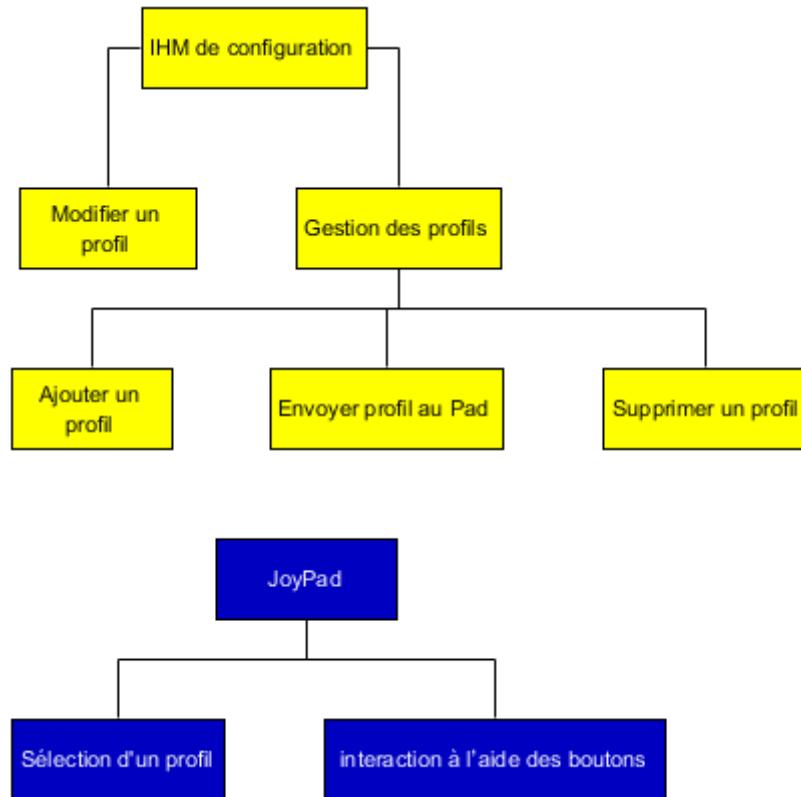


Figure 10 - répartition sur les sites

Les structures des données et des fonctions liées à chacun des programmes sont précisées dans cette sous-partie. Leur fonctionnement sera précisé dans la partie Scénarios (Cf.2.2.3)

2.1.1. Le programme du JoyPad

Ce programme sera créé en structure de programmation orientée objet. Voici le diagramme de classes qui le représente :

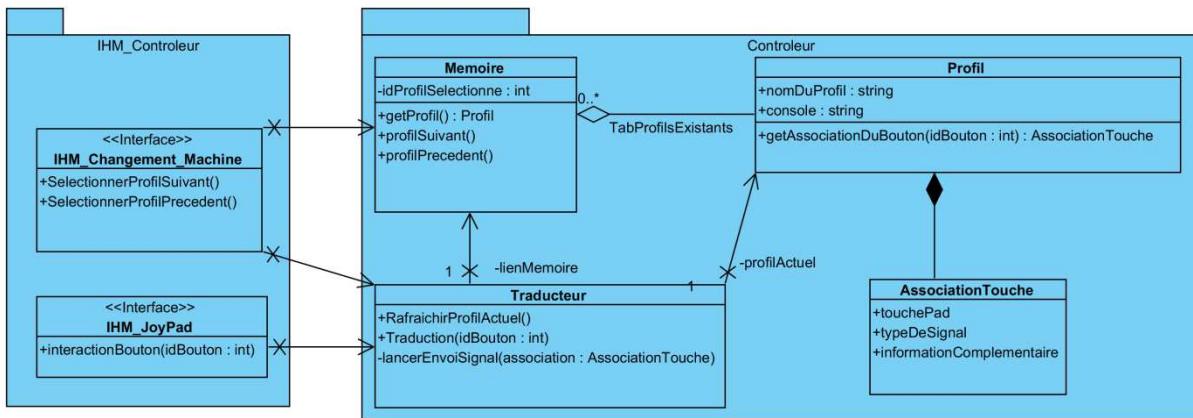


Figure 11 - Diagramme de classe du JoyPad

Il est constitué de deux paquets, l'un servant à caractériser les IHM nommés « IHM_contrôleur », le second servant à caractériser les autres fonctions clés du JoyPad nommé « Contrôleur ».

Dans le premier paquet, la classe IHM_Changement_Machine contient toutes les fonctions qui sont en lien direct avec le changement de Profil de Jeu, le second caractérise plutôt les fonctions liées aux touches de jeu. Les fonctions présentées ici sont appelées lors de l'interaction d'un utilisateur sur l'un des boutons du JoyPad.

Dans le second paquet, la classe Mémoire permet de symboliser le cœur informationnel du dispositif de jeu. Elle contient toutes les informations dont a besoin la classe Traducteur pour opérer. Ces informations sont les profils enregistrés dans le JoyPad, contenant des Associations entre touches et signaux à envoyer.

2.1.2. Le programme de configuration du JoyPad

De même que pour le programme précédent, il sera créé en structure de programmation orientée objet. Voici le diagramme de classes qui le représente :

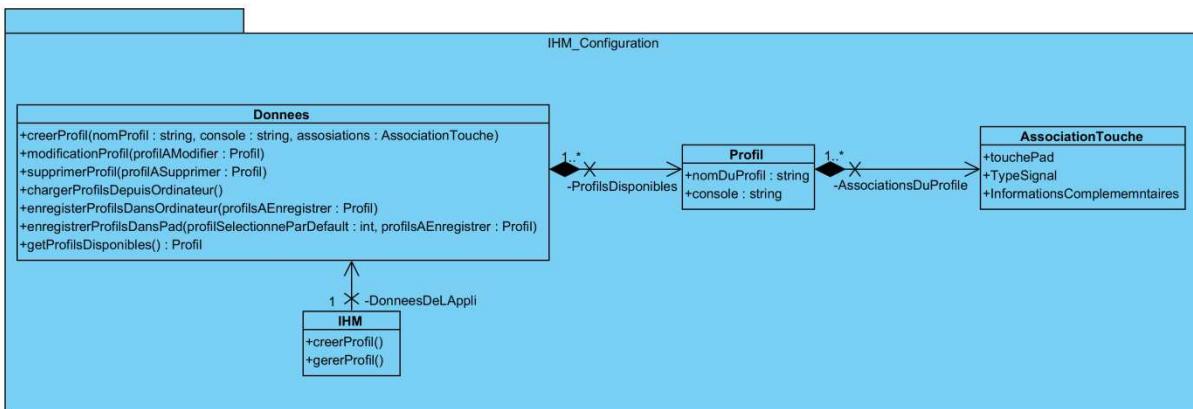


Figure 12 - Diagramme de classes de l'IHM de configuration

Le programme est ici représenté sous une forme assez simplifiée (notamment pour la Classe IHM). La classe Données contient toutes les données des profils créés sur l'ordinateur et permet par le biais de ses fonctions de les modifier. Les données sont identiques à celles du programme du JoyPad. Ici, l'interaction avec l'utilisateur se fera à l'aide de la classe IHM.

2.2. Description des principales fonctions

2.2.1. Objectif et description

Les grandes fonctions qui seront développées sur ce dispositif sont:

- Ajouter un profil : permet de créer un profil en précisant son nom et la machine associée. Une attribution des touches par défaut sera utilisée.
- Supprimer un profil : supprime un profil donné.
- Éditer un profil : permet de changer le nom, la machine associée et l'attribution des touches d'un profil.
- Sélectionner un profil : choisit le profil courant du pad.
- Interagir avec les boutons : permet de réaliser des actions sur le jeu.

2.2.2. Type d'utilisateur

L'utilisateur "joueur" est le seul utilisateur, il peut ajouter et supprimer des profils depuis l'IHM de configuration. De même, il peut renommer, sélectionner une machine et attribuer des touches. Depuis le pad, il peut sélectionner un profil à travers l'IHM de changement de profil et interagir avec les boutons à travers l'IHM JoyPad.

2.2.3. Scénarios (diagrammes de séquence ou d'activité)

2.2.3.1. Scénarios liés au JoyPad

2.2.3.1.1. Changement du profil du pad

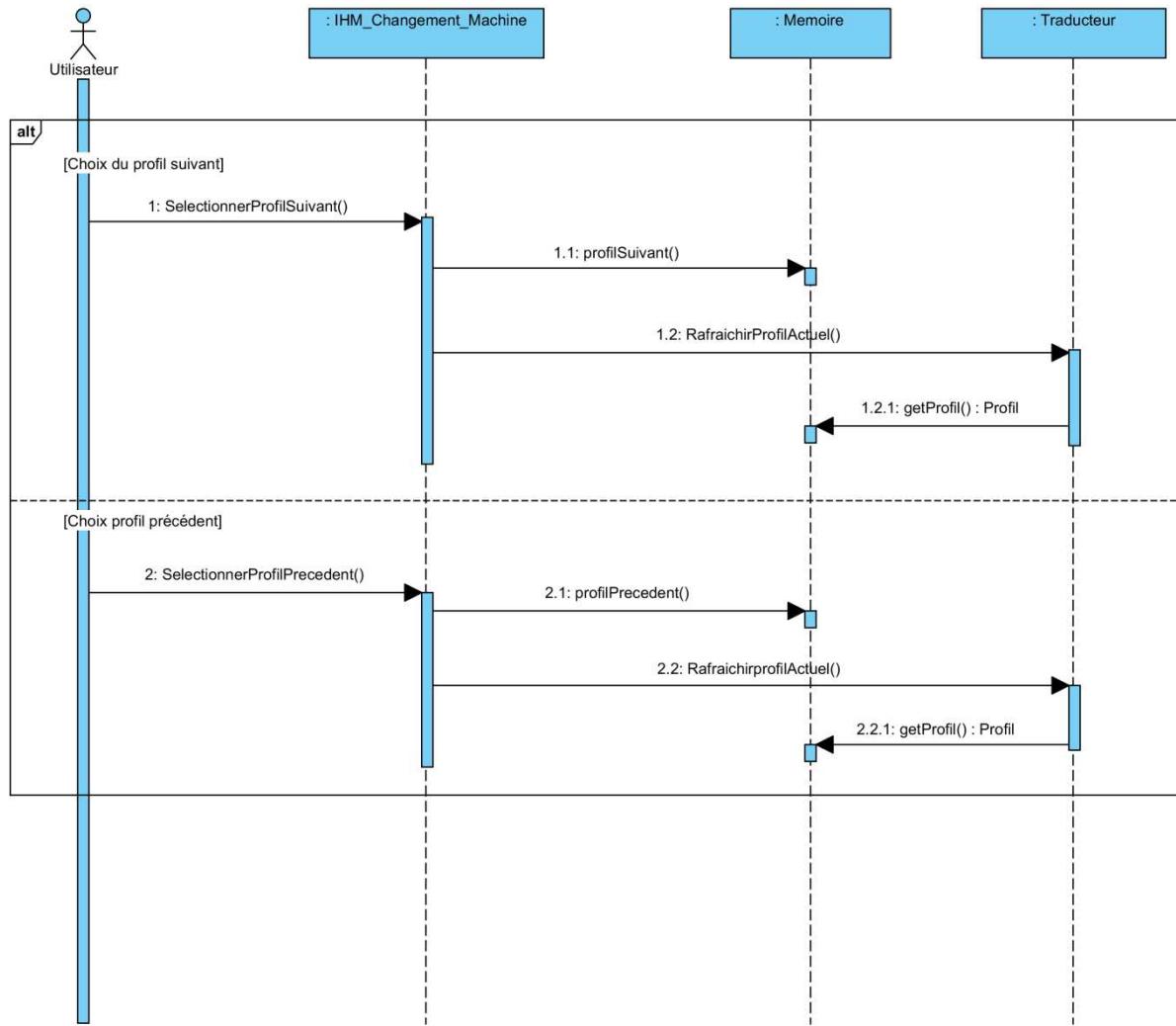


Figure 13 - Diagramme de scénario changement de profil

DEBUT : Sur l'écran LCD est affiché le profil sélectionné.

OPERATION : En appuyant sur un des deux boutons à côté de l'écran de sélection, on déclenche respectivement la méthode « sélectionnerProfilSuivant » ou « sélectionnerProfilPrécédent » de l'interface homme-machine. Celles-ci appelleraont les méthodes correspondantes dans la mémoire : on change l'id du profil sélectionné en l'incrémentant ou le décrémentant selon le bouton choisi.

Enfin, la mémoire appellera la méthode « rafraîchirProfilActuel » de la classe Traduction : celle-ci récupérera le profil actuel via un appel à la méthode « getProfil » de la classe mémoire.

FIN : le profil sélectionné a changé.

2.2.3.1.2. Interaction de l'utilisateur sur le pad

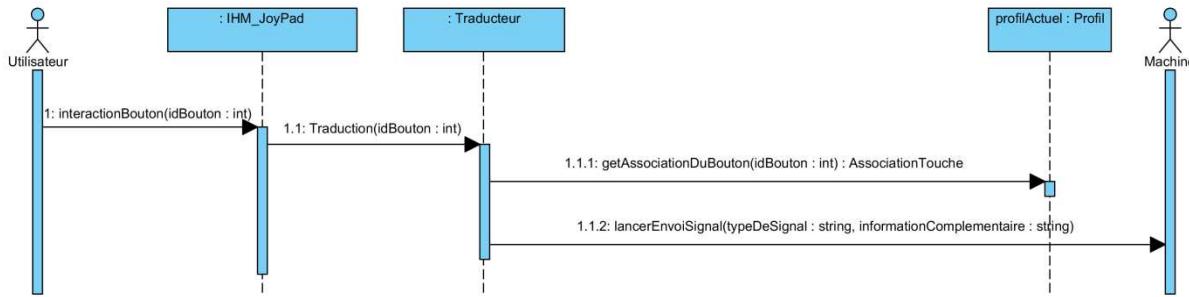


Figure 14 - Diagramme de scénario interaction

DEBUT : le joueur utilise un bouton (ou le joystick, qui est en réalité composé de 4 interrupteurs, et fonctionne donc comme quatre boutons).

OPERATION : le fait d'appuyer sur un bouton appelle une méthode « interactionBouton » de l'IHM, avec en paramètre l'identifiant du bouton.

L'IHM va alors appeler la méthode « traduction » de la classe Traducteur (toujours avec l'identifiant du bouton en paramètre). Le traducteur appelle la méthode « getAssociationDeBoutons » avec l'identifiant du bouton en paramètre, et récupère une structure de type « associationTouche ».

Grace aux informations contenues dans cette structure associationTouche, on appelle une fonction « lancerEnvoiSignal » avec en paramètre toutes les informations nécessaires à l'envoi du bon signal.

FIN : La machine a reçu le signal correspondant à l'appui sur le bouton.

2.2.3.2. Scénarios liés à l'IHM de configuration

2.2.3.2.1. Ajout d'un profil sur la mémoire du dispositif de configuration du pad

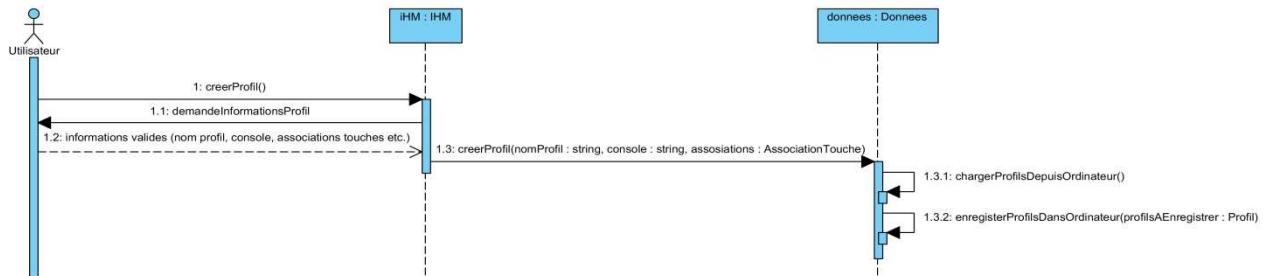


Figure 15 - Diagramme de scénario ajout de profil

DEBUT : On veut créer un profil, on clique sur « créer un nouveau profil ».

OPERATION : On appelle la méthode « creerProfil » de l'IHM. Celle-ci exécute une autre méthode qui va récupérer auprès de l'utilisateur les informations à mettre dans le profil, puis déclencher la méthode creerProfil de la classe Données.

Celle-ci va charger tout les profils depuis l'ordinateur via sa méthode « chargerProfilsDepuisOrdinateur » puis tout enregistrer via sa méthode « enregisterProfilsDansOrdinateur » (avec en paramètre la liste des profils).

FIN : On a créé et enregistré un nouveau profil.

2.2.3.2.1. Gestion des profils entre la mémoire du dispositif de configuration et le pad

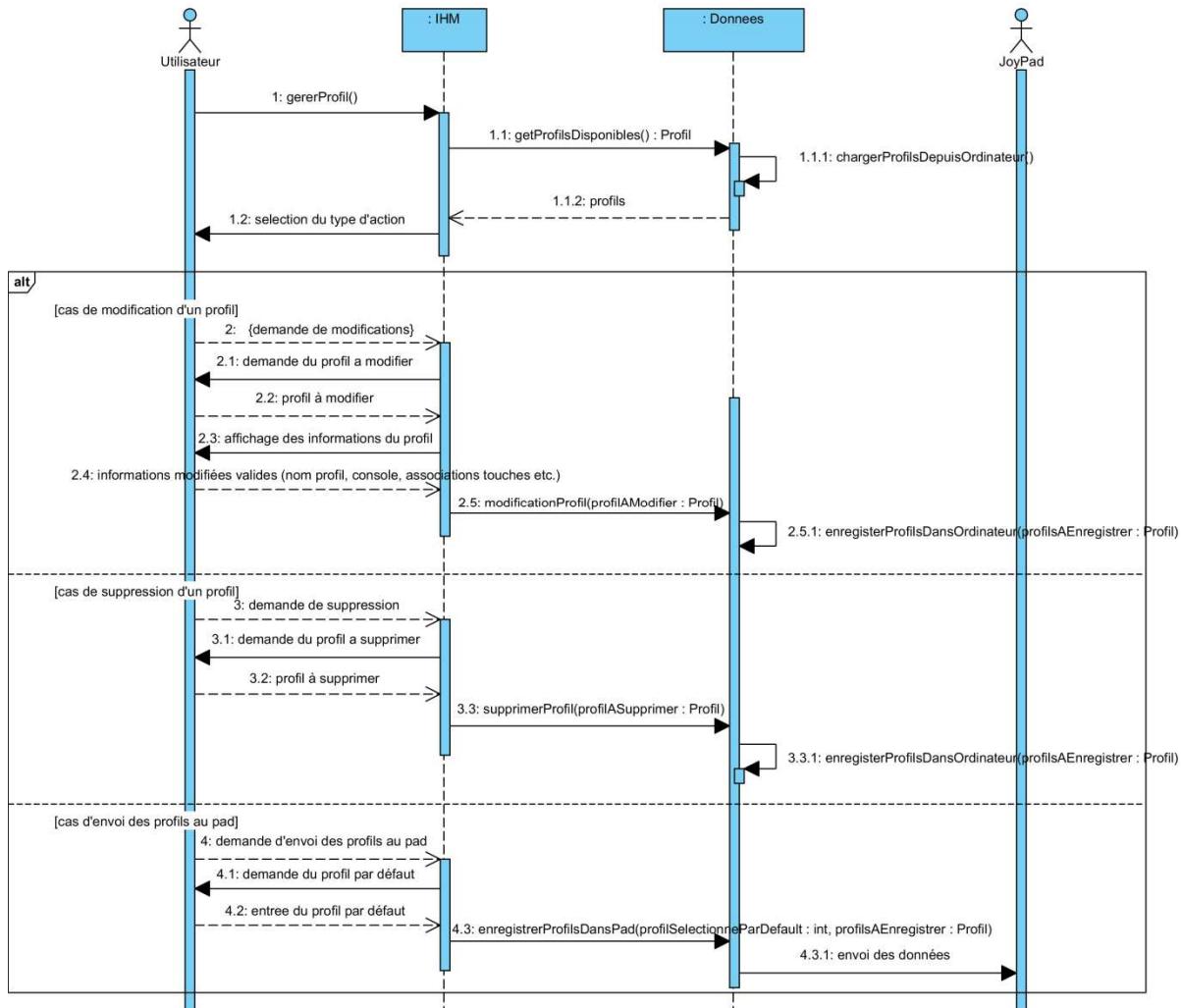


Figure 16 - Diagramme de scénario gestion de profil

DEBUT : On veut gérer (modifier, supprimer ou envoyer sur le pad) les profils, on a cliqué sur le bouton ou onglet « gérer les profils ».

OPERATION : la méthode « gererProfil » de l’IHM se lance et appelle la méthode « getProfilsDisponibles » de la classe Données. Celle-ci charge en mémoire les profils via sa méthode « chargerProfilsDepuisOrdinateur », et l’IHM les récupère (sous forme de tableau) en retour de la méthode « getProfilsDisponibles », puis attend que l’utilisateur sélectionne le type d’action qu’il désire effectuer.

Trois choix sont possibles :

1. Modifier un profil.

À sa sélection, l’IHM reçoit une demande de modifications. Elle demande quel est le profil à modifier et récupère l’information. Elle affiche les données du profil, et récupère des données modifiées valides (nom du profil, console associée et associations de touches). Le profil ainsi modifié est envoyé à la classe Données via la méthode « modificationProfil » (avec le profil modifié en paramètre). La méthode « enregisterProfilsDansOrdinateur » enregistre la nouvelle liste des profils.

2. Supprimer un profil.

À sa sélection, l'IHM reçoit une demande de suppression. Elle demande alors quel est le profil à supprimer, et récupère cette donnée en retour. Elle appelle donc la méthode « supprimerProfils » et supprime le profil de la liste des profils. Enfin, la classe Donnée enregistre la nouvelle liste via sa méthode « enregistrerProfilsDansOrdinateur ».

3. Envoi des profils au pad.

À sa sélection, l'IHM reçoit une demande d'envoi des profils au pad. Elle demande alors le profil par défaut et récupère en retour cette information. La méthode « enregistrerProfilsDansPad » est appelée, avec en paramètres un tableau contenant tous les profils, ainsi que l'identifiant du profil par défaut. La partie de l'initialisation dans le code source du programme de l'arduino est modifiée pour intégrer les nouvelles données, puis le tout est compilé et envoyé au JoyPad.

FIN : on a modifié et/ou supprimé des profils et/ou chargé sur le JoyPad les données.

2.2.4. Données utilisées

Les données utilisées sont multiples. Elles concernent avant tout les profils de mappage de touche. Par exemple, pour chaque profil il faut lui déterminer :

- Un nom de profil de mappage.
- La machine à laquelle est rattaché le profil. Cela permet d'adapter le protocole des signaux émis vers la machine destinatrice. Ainsi, les protocoles d'envoi d'instructions pour la machine devront être programmés à l'avance pour pouvoir en ajouter une nouvelle. Cette information peut notamment être accompagnée d'une photographie ou d'un schéma représentant les touches à affecter aux touches du JoyPad.



Figure 17 - Exemple de photographie de manette d'Atari 2600

- Les diverses affectations des touches du JoyPad vers les touches de la manette d'origine. Cela peut également être une affectation d'une macro de touches, d'un combo ou d'un auto-fire.

Ces données seront stockées sur deux supports différents : la mémoire du dispositif de configuration du JoyPad et la mémoire persistante du Joypad. Cependant, il ne sera possible de modifier les données que sur les profils enregistrés sur le dispositif de configuration du JoyPad.

2.2.5. IHM

2.2.5.1. IHM de configuration

2.2.5.1.1. Paramétrer les profils

Le logiciel sera présenté comme tel : il sera constitué de deux onglets, l'un permettant de paramétrer les divers profils existants sur l'ordinateur (A), l'autre de gérer les profils enregistrés sur le Pad (B).

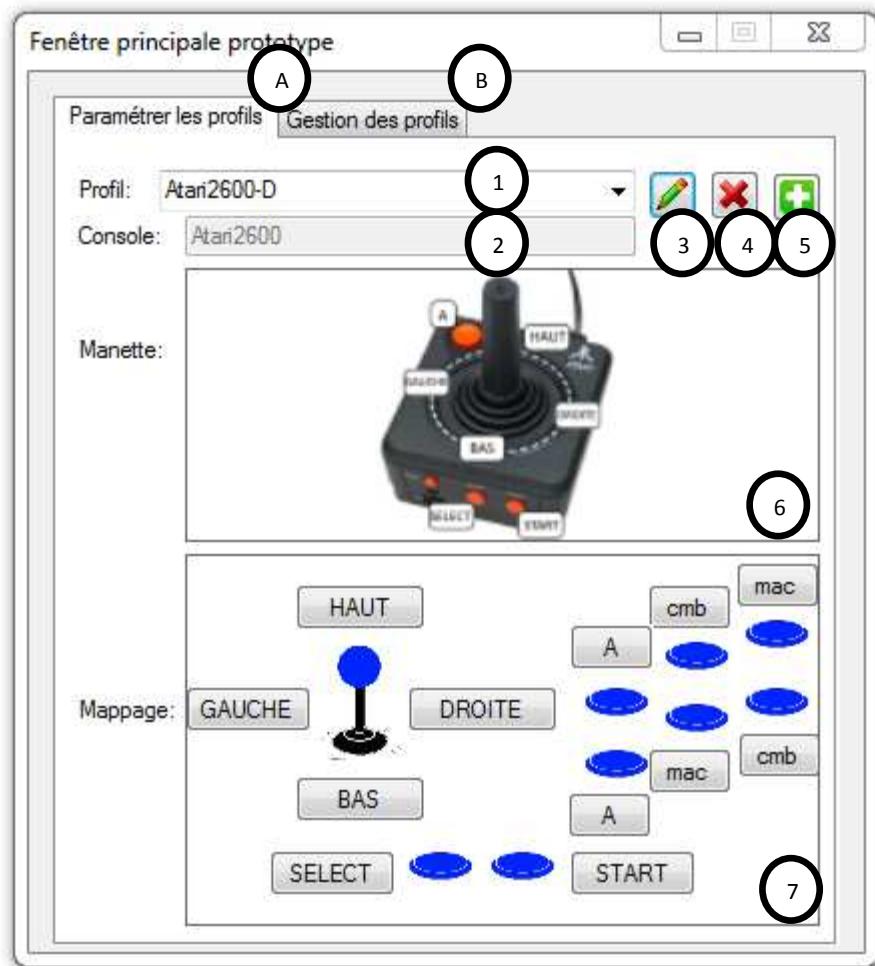


Figure 18 - Fenêtre principale de l'IHM de configuration

Le profil en cours d'édition est celui sélectionné dans la liste «profil » (1). Chaque profil correspond à une console (2) précise ce qui permet de mapper uniquement les touches disponibles sur la manette d'origine (6).

Cette interface permettra :

- D'éditer le nom du profil (3)



Figure 19 - Modification du nom de profil

- De supprimer le profil courant (4)



Figure 20 - Suppression d'un profil

- D'ajouter un nouveau profil (5)

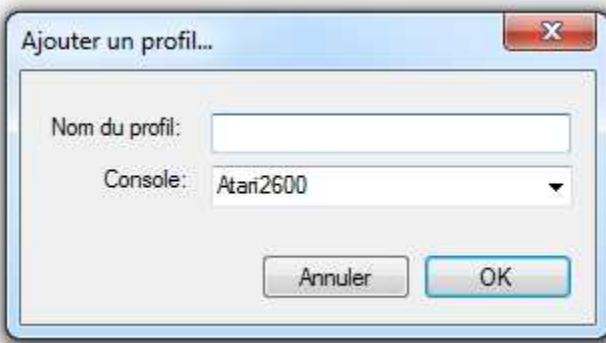


Figure 21 - Ajout d'un profil

De paramétriser (mapper) les diverses touches du Pad (7) :

- Affecter une touche du pad à une touche de la manette d'origine (7A)
- Affecter une touche du pad à une combinaison de touches de la manette d'origine (7B)
- Affecter une touche du pad à une macro de touches de la manette d'origine (7C)

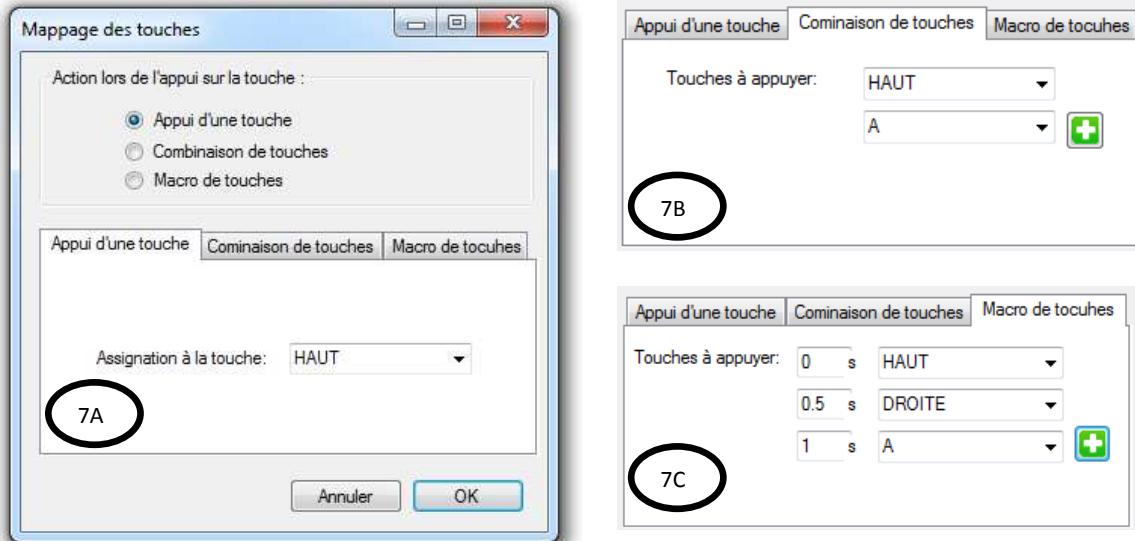


Figure 22 - Association des touches

2.2.5.1.2. Gestion des profils

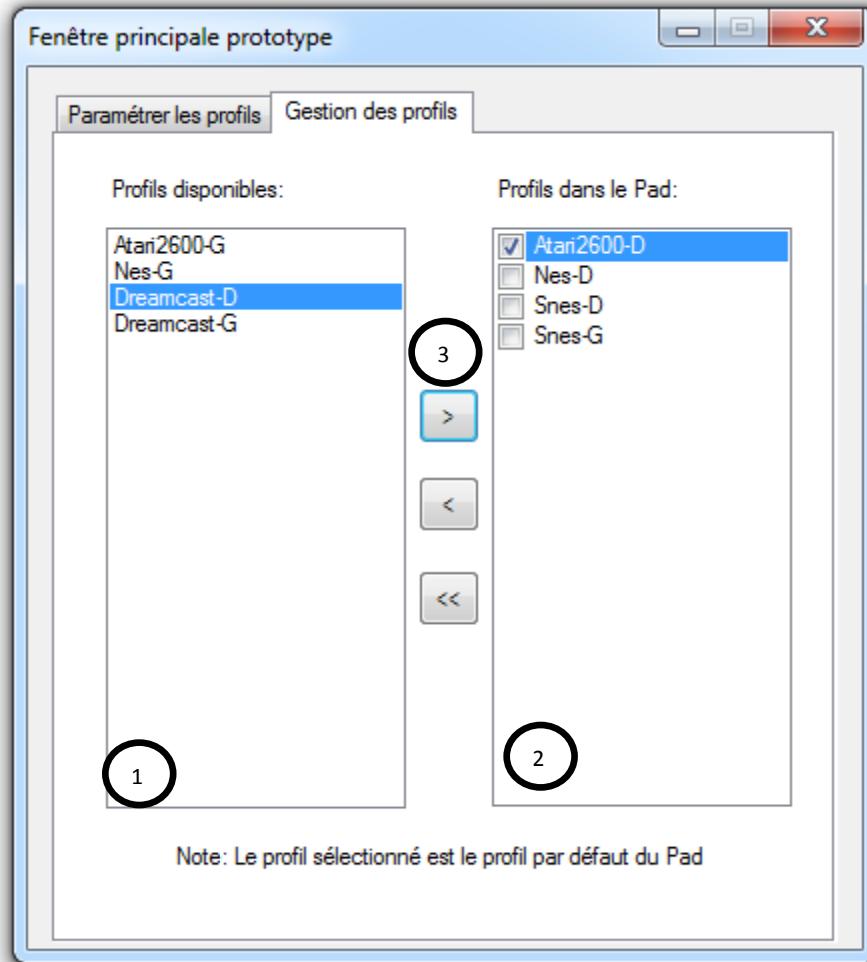


Figure 23 - Gestionnaire des profils

L'interface de gestion des profils présentera la vue de la liste des profils disponibles dans l'ordinateur (1)(qui n'ont pas été transférés sur le Pad) et ceux présents sur le Pad (2). Les boutons servent au transfert des profils de l'ordinateur au Pad (3). La sélection du profil utilisé par défaut au démarrage du Pad est effectuée dans la liste des profils du Pad, il sera sélectionné au démarrage du Pad.

2.2.5.2. IHM changement machine

L'IHM pour changer le profil choisi devra permettre de pouvoir changer le profil interprété d'une simple pression de bouton. Dans le cas où l'on utiliserait un écran LCD accompagné de deux boutons on pourrait le représenter ainsi :

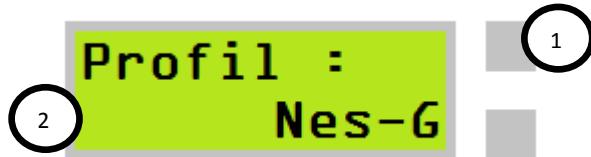


Figure 24 - IHM changement machine

Ainsi, on pourrait choisir le profil via les deux boutons (1) et le nom du profil actuellement utilisé serait affiché(2).

2.2.5.3. IHM Joypad

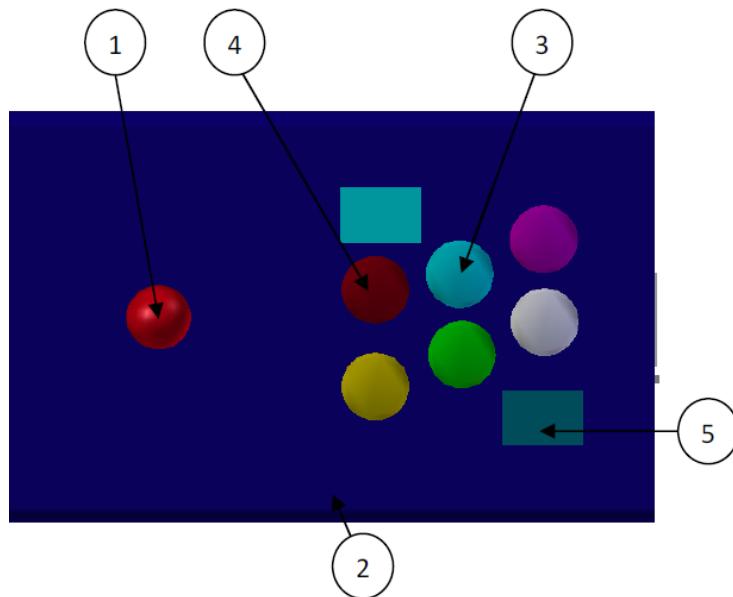


Figure 25 - IHM JoyPad

(1) Stick analogique

Un stick 4 directions mécanique. Les diagonales seront matérialisées par une association de touches.

(2) Boutons programmables start & select

Deux boutons programmables placés de la même manière que les boutons start & select d'un pad classique.

(3) Boutons programmables fixes

Boutons programmables qui seront placés de la même façon, quelle que soit l'orientation de la manette.

(4) Boutons programmables sur glissières

Boutons programmables qui seront placés sur des glissières, de sorte à ce que l'on puisse changer leur position selon l'orientation de la manette.

(5) Cales

Pièces de plastique permettant de fixer les boutons sur glissières. (cf. fig 3)

2.3. Documents supplémentaires

Il sera fourni un manuel permettant la fabrication du contrôleur par l'utilisateur dans un FabLab.

C. Analyse de différentes connectiques

1. La connexion à l'Arduino

1.1. La partie interne au contrôleur.

Afin de pouvoir faire communiquer le contrôleur, il sera nécessaire de le relier aux divers composants. Cette partie sert à délimiter les branchements qui seront opérés.

Dans un premier temps, l'Arduino sera alimenté en électricité par une batterie externe reliée en **USB (A)**. Afin de pouvoir réaliser la communication entre le contrôleur et la console, il sera nécessaire de la relier via **8 pins (C)**. De même, le composant permettant de communiquer avec la NES/SNES aura besoin de **8 autres pins (D)**. L'Arduino nécessitera également d'être connectée à l'IHM de jeu constitué de 12 entrées (6 touches d'actions, une touche Start, une touche Select et 4 touches de directions) qui seront reliés à **12 pins différents (E)**. Deux entrées seront également réservés à l'IHM de sélection de profil **(G)**. Au niveau du stockage, **4 pins (F)** sont nécessaires afin de connecter un lecteur de carte SD. Enfin, pour l'affichage, il sera nécessaire d'utiliser 6 pins pour l'envoi des données **(B1)** et un pin pour l'activation du rétroéclairage **(B2)**.

Bien évidemment, afin de faire circuler le courant il est nécessaire que chaque composant soit relié d'un coté à la terre et de l'autre au +5V. De ce fait, chaque composant de pins C à F se doivent d'être reliés à **GND**. De même, la masse envoyée par machine à contrôler doit être reliée à la masse de l'Arduino **GND**.

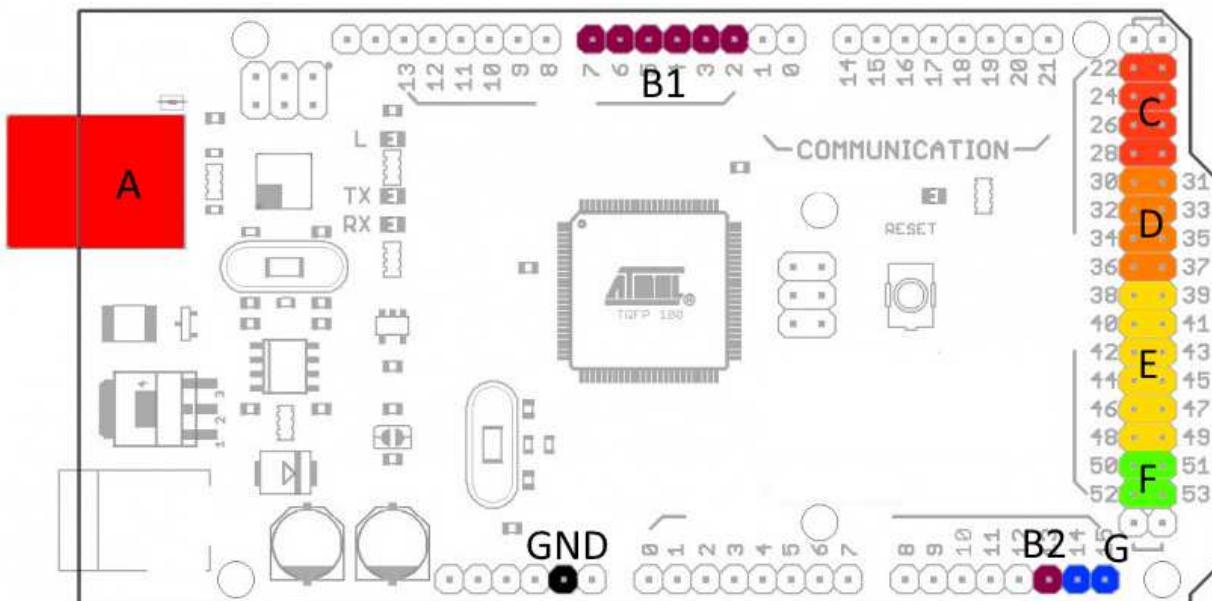
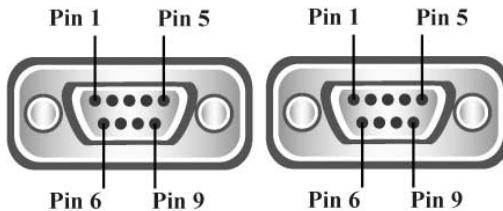


Figure 26 - plan d'utilisation des pins de l'arduino

1.2. La communication entre machine et contrôleur

Les 9 pins qui seront connectés en B à l'Arduino serviront à la communication entre la machine et le contrôleur. Afin d'établir cette communication, chaque pin sera relié à un câble RS232 afin de réaliser un connecteur universel.

N° Pin Arduino	N°fiche câble
GND	1
23	2
...	...
30	9



N°fiche câble	Correspondance
1	GND
2	CLOCK
3	DATA
...	...

Figure 27 - exemple de connectique entre l'arduino et la NES/SNES

2. Atari 2600/SMS/Mega Drive/CPC

2.1. Connecteur

Le connecteur possède 9 pins (dont trois non-utilisées).

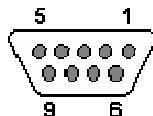


Figure 28—Connecteur 9 points D-SUB femelle

Pin	Atari 2600	Master System/megaDrive	Amstrad CPC
1	Up	Up	Up
2	Down	Down	DOWN
3	Left	Left	LEFT
4	Right	Right	RIGHT
5	n/c	n/c	n/c
6	Fire	Button 1	FIRE2
7	NC	N/C	FIRE1
8	Ground	GND	GND
9	NC	Button 2	GND

Figure 29 – Fonctions des pins sur diverse consoles

2.2. Protocole

Le protocole de transmission de l'appui des touches sur la manette se base sur le principe du court-circuitage. En effet, les boutons de la manette sont en réalité des interrupteurs. Soit ils sont pressés, et le courant est redirigé vers la terre, soit ils ne le sont pas et le courant passe dans la console. Lorsque le courant est redirigé vers la terre, il ne passe pas dans la console, c'est un court-circuitage qui est pour elle le signal indiquant l'appui sur la touche.

3. NES/SNES

Les manettes possèdent 7 pin (dont 2 inutilisés) : le 5V et le ground, le clock par lequel la console va envoyer un signal d'horloge, le latch qui annonce l'envoie de données par la manette et le data par lequel la manette va envoyer l'état des boutons.

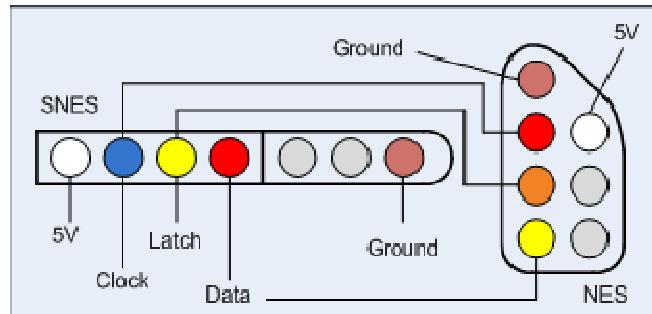


Figure 30 - pin (I/O) manettes NES/SNES

L'état des boutons stocké sur 8 bit (deux fois 8 bit pour la snes(1 = relâché, 0 = appuyé)). La console envoie un signal clock, et la manette répond par un 1 envoyé sur le pin latch. Les bits de données sont ensuite envoyé sur le pin data. Pour la snes, les 2*8 bit sont envoyé à la suite.

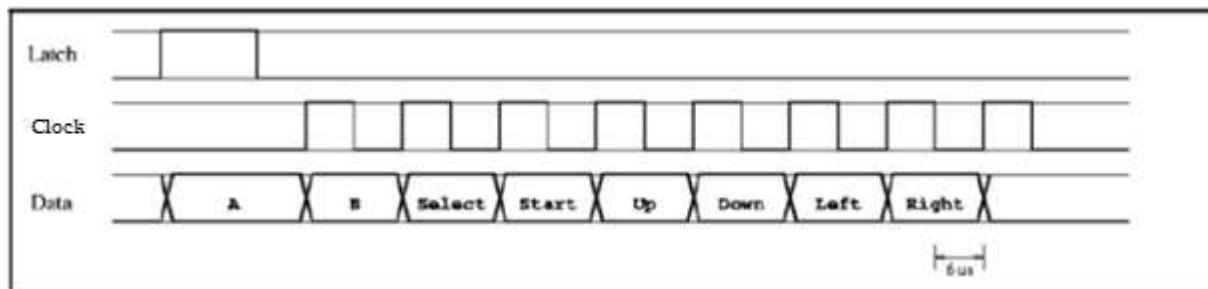


Figure 31 - Signal d'horloge NES

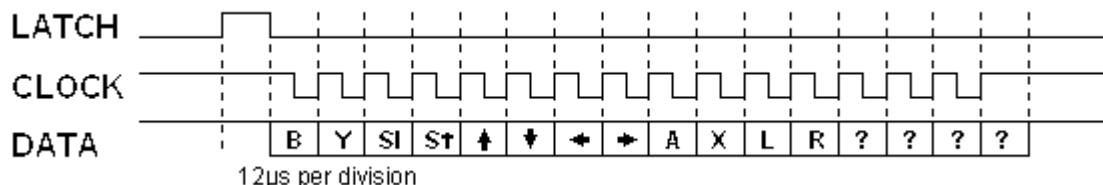


Figure 32 - Signal d'horloge SNES

D. Annexes

1. Vocabulaire

JoyPad : Appelé plus communément « Pad d'arcade », il s'agit d'un contrôleur de machine utilisé pour les jeux vidéo. L'objectif d'un JoyPad est de transmettre les instructions de jeu vers la machine destinatrice. Il a une ergonomie étudiée tout particulièrement pour le jeu.



Figure 33 - exemples de manette typée arcade

Profil de mappage : Un mappage correspond à une association d'une touche physique ou virtuelle (tactile) d'un dispositif vers une touche prévue sur une touche sur un périphérique d'une machine donnée. Par exemple, dans notre cas une touche de jeu du JoyPad peut être associée à une touche d'action « feu » sur une machine comme une Atari. Le profil de mappage correspond à la compilation de plusieurs données de mappage de touches associé à une machine précise et un nom Cf. 2.2.4 pour plus de détails.

Combo : Un combo est l'appui simultané de plusieurs touches d'un contrôleur de jeu. L'objectif d'un combo dans un jeu vidéo est de pouvoir faire des coups spéciaux.



Figure 34 - Exemple de combo

Macro: Une macro est une séquence d'appui de touches d'un contrôleur de jeu avec un timing précis. De la même façon qu'un combo, il sert dans un jeu vidéo à pouvoir faire des coups spéciaux.



Figure 35 - Exemple de macro

Autofire : L'autofire est une fonction, souvent implémentée dans les contrôleurs de machines non-officiels, permettant de simuler l'appui répété et rapide d'une touche. Il permet avant tout dans un jeu vidéo de faire des actions assez répétitives comme l'appui répété de la touche « fire » (feu), très utilisé dans des jeux de type « shot-em up »



Figure 36 - le shoot'em up Jamestown

IHM : Abréviation d'Interface Homme-Machine, il s'agit d'une interface physique ou virtuelle (tactile) permettant de faire un lien entre l'utilisateur et les processus de la machine. Elle peut se présenter sous diverses formes, dans notre cas le panneau contenant les touches de jeu du JoyPad est considéré comme une IHM : elle permet de transmettre au JoyPad les requêtes de l'utilisateur vers la machine destinatrice. Un autre exemple d'IHM est le programme (ou l'application) permettant de configurer les profils de mappage du JoyPad.

Arduino : Un arduino est une petite carte mère composée principalement d'un processeur peu vêloce et de nombreuses interfaces de programmations. Il est notamment utilisé dans le domaine de l'embarqué pour des tâches spécifiques à des questions de faible coûts, de contraintes d'espace et d'énergie consommée. Il est programmable en C / C++.



Figure 37 - Photo de l'arduino Uno

2. Table d'illustration

Figure 1 - Pad Classique.....	4
Figure 2 - Schéma explicatif de l'utilité de la manette	4
Figure 3 – Réversibilité Gaucher/Droitier.....	6
Figure 4 - Pad Classique.....	7
Figure 5 - Schéma explicatif de l'utilité de la manette	7
Figure 6 - Réversibilité Gaucher/Droitier	8
Figure 7 - Diagramme des cas d'utilisation réaliste.....	9
Figure 8 - Diagramme des cas d'utilisations optimal.....	10
Figure 9 - Diagramme d'architecture	11
Figure 10 - répartition sur les sites.....	13
Figure 11 - Diagramme de classe du JoyPad	14
Figure 12 - Diagramme de classes de l'IHM de configuration.....	15
Figure 13 - Diagramme de scénario changement de profil.....	17
Figure 14 - Diagramme de scénario interaction.....	18
Figure 15 - Diagramme de scénario ajout de profil.....	19
Figure 16 - Diagramme de scénario gestion de profil	20
Figure 17 - Exemple de photographie de manette d'Atari 2600.....	22
Figure 18 - Fenêtre principale de l'IHM de configuration	23
Figure 19 - Modification du nom de profil	24
Figure 20 - Suppression d'un profil	24
Figure 21 - Ajout d'un profil	24
Figure 22 - Association des touches	25
Figure 23 - Gestionnaire des profils	26
Figure 24 - IHM changement machine	27
Figure 25 - IHM JoyPad.....	27
Figure 26 - plan d'utilisation des pins de l'arduino	29
Figure 27 - exemple de connectique entre l'arduino et la NES/SNES.....	30
Figure 28 – Connecteur 9 points D-SUB	31
Figure 29 – Fonctions des pins sur diverse consoles.....	31
Figure 30 - pin (I/O) manettes NES/SNES.....	32
Figure 31 - Signal d'horloge NES.....	32
Figure 32 - Signal d'horloge SNES.....	32
Figure 33 - exemples de manette typée arcade.....	33
Figure 34 - Exemple de combo.....	33
Figure 35 - Exemple de macro.....	33
Figure 36 - le shoot'em up Jamestown.....	34
Figure 37 - Photo de l'arduino Uno	34