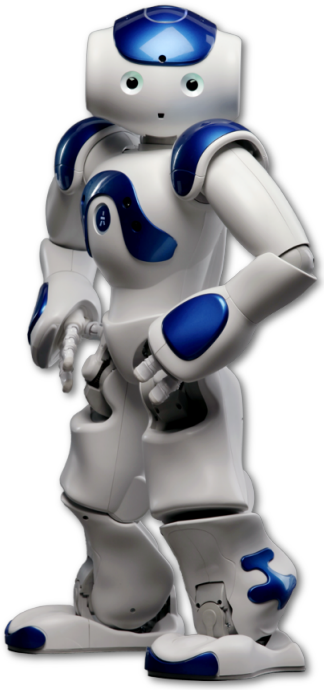
NAO, un soutien dans le traitement de l'autisme infantile ?



Koç Murat, Lepicard Céline, Michalak Stanislas

25 mai 2016

Sommaire

Remerciements

À l'occasion de ce projet de fin d'études, nous avons eu le plaisir de collaborer avec certains membres du laboratoire de l'**ETIS** de l'Université de Cergy-Pontoise, ainsi qu'avec l'association **Autisme Ensemble 95**.

Dans un premier temps, nous tenons en particulier à remercier Ghilès Mostafaoui pour nous avoir encadré et guidé tout au long de ce projet.

Nous remercions également Éva Ansermin pour ses conseils et son aide précieuse sur le développement de notre script et tout particulièrement sur l'utilisation de Prométhée ; ainsi qu'Arnaud Blanchard qui nous a guidé pour notre travail sur l'interfaçage entre Prométhée et le robot.

Merci également à Nils Beausse, Fanny Larradet et Caroline Grant d'avoir également pris le temps de nous aider.

Nous tenions à remercier chaleureusement les membres de l'association Autisme Ensemble 95 ainsi que les familles et les enfants qui ont pris — ou qui prendront part — à ce projet. Merci à Isabelle Rolland et à son équipe, qui nous ont accueilli à plusieurs reprises et sans qui le projet ne pourrait avoir de valeur.

Enfin, ce projet n'aurait jamais existé sans l'investissement du groupe des anciennes Eistiennnes : Fanny Larradet, Mylene Le Hen et Aurelie Fajardo qui sont à l'origine de ce travail de collaboration avec le laboratoire de l'ETIS et l'association Ensemble 95. Merci à Nga Nguyen de nous avoir soutenu durant ces six derniers mois.

Introduction

En août 2014, lors de la 23e édition du Symposium international sur la communication interactive entre le robot et l'Homme, les résultats d'une étude américaine menée par des chercheurs de la Viterbi School of Engineering suggèrent que le recours à des robots éducatifs pourraient améliorer la prise en charge des enfants atteints de troubles du spectre autistique [1]. Ces résultats, très encourageants, sont en accord avec d'autres études similaires et incitent les professionels du domaine de la santé et de l'éducation à avoir recours à la thérapie robotique pour lutter contre l'autisme, en particulier chez les jeunes enfants.

Dans le cadre de notre projet de fin d'étude (PFE) à l'EISTI, nous avons eu l'opportunité — en collaboration avec l'ETIS — de travailler sur une étude de synchronisation entre le robot NAO d'Aldebaran et un enfant atteint d'autisme.

Le principe de ce projet : programmer Nao afin qu'il puisse alterner entre une série de mouvements simples reproductibles par un enfant autiste. La navigation entre les différents mouvements et la vitesse de ceux-ci sont gérés par une wiimote qui joue le rôle de télécommande. Nous avons également travaillé sur l'aspect de synchronisation afin que Nao puisse s'adapter au mieux à la vitesse d'exécution des enfants.

Ce document est divisé en deux parties majeures : la première expose la conception du projet à l'aide de diagrammes SysML, un langage de modélisation. La seconde présente l'implémentation du projet, comportant deux aspects : le script sous Coeos et le driver Python.

1. `http://viterbi.usc.edu/news/news/2014/august-28-2014.htm`

Chapitre 1

Présentation du projet

1.1 Contexte

Bien que récents, les travaux sur l'autisme et les résultats des différents laboratoires de recherche sur la thérapie robotique pour lutter contre l'autisme donnent des résultats très prometteurs.

Intéressés par ces recherches, nous avons choisi de reprendre les travaux entrepris par d'anciennes étudiantes de l'option INEM. Ces travaux visaient à démontrer que l'on pouvait apporter un soutien dans le traitement de l'autisme chez les enfants, par l'utilisation du robot NAO.

Ce choix est justifié par l'aspect visuel attractif du robot, et sa capacité à produire/reproduire des gestes de façon continue et sans altération — contrairement à un être vivant. Le but de cette étude était donc d'établir une synchronisation entre le robot et l'enfant, en faisant reproduire par ce dernier des gestes simples et répétés par le robot. NAO, pour aider l'enfant, devait adapter la vitesse d'exécution du geste à celle de l'enfant : on parle de *couplage* entre l'enfant et NAO.

À la suite de ce projet de fin d'études, le laboratoire de recherche avec lequel elles avaient travaillé — l'ETIS — a continué à avancer sur ce sujet avec les mêmes méthodes. De ces travaux ont abouti un modèle calibré pour effectuer cette tâche de synchronisation, basé sur les oscillations d'un pendule dont on vient ajuster la fréquence.

Encadrés par l'ETIS, nous avons pour notre part récupéré les travaux réalisés ; et nous cherchons à améliorer l'expérience par les points suivants :

— **Pérenniser le programme** développé en le faisant donctionner sur un NAO plus récent : celui utilisé pour les précédents travaux n'est plus utilisable de par son état, et il s'agissait d'un modèle de première génération, donc âgé d'au moins 7 ans.
— **Utiliser une Wiimote** comme contrôleur à disposition d'un praticien pour faciliter les opérations (le modèle proposé par nos prédécesseurs était basé sur une IHM déportée sur un smartphone).
— **Réduire les distractions** de l'enfant autiste par l'amélioration de la gestion du câblage du robot et l'intégration d'interactions pour capter leur attention.
— Intégrer d'**autres mouvements** possibles.

1.2 Problématique

La robotique humanoïde peut-elle avoir un impact positif sur le traitement du spectre autistique ?

1.3 État de l'art

1.3.1 Autisme

L'autisme est une maladie neurologique qui affecte le fonctionnement du cerveau, le système immuniataire et biologique, altère les capacités de reconnaissance des expressions, des codes sociaux et affectifs, génère hypersensibilité émotionnelle et troubles du comportement. Elle affecte 643 000 personnes en France dont 160 000 enfants [1].

L'autisme est souvent diagnostiqué dès l'entrée en société. Un enfant autiste se distingue des autres par ses difficultés d'intégration sociale, ses problèmes de communication verbale et non verbale, un comportement répétitif voir obsessionnel et des centres d'interêts restreints. Le spectre autistique est large et hétérogène, chaque enfant autiste est atteint différemment. Les caractéristiques citées apparaissent toutefois chez la majorité des enfants autistes, à des degrés plus ou moins importants. [2]

1. Chiffres officiels : `http://www.vaincrelautisme.org/content/une-maladie-neurologique-qui-detruit-la-vie`
2. Plus d'information sur : `http://www.vaincrelautisme.org/`

Bien qu'handicapant, l'autisme n'est aujourd'hui plus une fatalité. Il n'existe effectivement pas de rémède miracle, mais de nombreuses recherches sont effectuées à ce sujet et les traitement éducatifs — notamment à l'aide de robots — ont donné des résultats très encourageants.

1.3.2 La robotique et l'autisme

Dès leur plus jeune âge, les enfants s'amusent à reproduire les comportements ou les gestes de leur entourage. L'imitation constitue un modèle d'apprentissage important chez l'enfant. Les enfants autistes ont justement des difficultés à imiter les comportements observés à cause de leurs problèmes d'ordre relationnel : il se mettent alors en retrait et n'arrivent pas à s'intégrer à leur environnement.

Partant de ce constat, des scientifiques ont eu l'idée de restreindre la complexité des mouvements, en éliminant notamment toutes les micro-expressions parasites et autres stimulis propres à l'être humain qui gênent les enfants autistes pour intéragir avec ces derniers. Les robots humanoïdes constituent actuellement la meilleure alternative à l'être humain : ils permettent en effet de reproduire les gestes effectués par l'Homme, en éliminant tout caractère parasite complexe. Le but est d'inciter l'enfant atteint d'autisme à imiter les gestes effectués ou de répondre aux consignes et donc d'interagir avec le robot comme s'il le faisait avec une personne ordinaire.

Les scientifiques affirment que le fait de se retrouver face à un robot permettrait en quelque sorte de « libérer » les autistes d'une pression sociale qui les paralyse. Le robot, généralemment contrôlé par un praticien, facilite le contact avec les enfants et stimule l'apprentissage de ceux-ci. Un reportage très interessant sur ce sujet, mené par une équipe de journalistes de la chaîne de télévision ARTE dans une école de Birmingham est d'ailleurs disponible sur Internet [3].

3. `https://www.youtube.com/watch?time_continue=66&v=8BiayzQxTQA`

1.4 Collaborateurs

1.4.1 ETIS

L'ETIS, *Équipes Traitement de l'Information et Systèmes*, est une unité de recherche commune au CNRS, à l'ENSEA Cergy et à l'Université de Cergy-Pontoise. Elle compte environ 50 personnels permanents (enseignants-chercheurs et personnels support) et accueille une cinquantaine de doctorants et post-doctorants.

Principalement rattaché à l'*Institut des Sciences Informatiques et leurs Interactions* (INS2I), l'ETIS est structurée en quatre équipes de recherche [4] :

- Indexation Multimedia et Intégration de données (MIDI)
- Information, Communications, Imagerie (ICI)
- Architectures, Systèmes, Technologies pour les unités Reconfigurables Embarquées (ASTRE)
- Neurocybernétique

Dans le cadre de notre projet, nous avons travaillé en collaboration avec l'équipe de Neurocybernétique située des les locaux de l'université de Cergy-Pontoise. Cette équipe travaille sur la modélisation des structures cérébrales et implémente les modèles effectués sur différents robots. Nous avons notamment utilisé leurs travaux sur la synchronisation pour notre projet.

1.4.2 Autisme Ensemble 95

Autisme Ensemble 95 est une association basée à Pontoise qui regroupe des parents et des professionnels. Leur objectif est de créer un pôle actif en matière d'autisme et de *Troubles du Spectre Autistique* (TSA). Nous avons travaillé en collaboration avec cette association qui nous a donné un terrain favorable et un cadre pour l'expérimentation de nos travaux sur un enfant atteint de TSA.

4. http ://www-etis.ensea.fr/index.php/recherche.html

Première partie

Conception

Cette partie a pour vocation de présenter notre travail de conception sur ce sujet : il présente le cahier des charges, ainsi que les différents diagrammes SysML modélisant le système étudié et les interactions possibles entre le praticien, l'enfant autiste et celui-ci.

Chapitre 2

Cahier des charges

NAOTISME_1	Sûreté
Le système ne doit pas être dangereux pour les sujets.	

NAOTISME_2	Ergonomie
Le système doit être facile d'utilisation pour un praticien.	

NAOTISME_3	
Le système ne doit pas perturber les sujets.	

NAOTISME_4	
Le système doit proposer des interactions imitables par les sujets.	

NAOTISME_5	Réactivité
Le système doit être réactif aux changements de paramètres.	

NAOTISME_6	Autonomie
Le système doit avoir une autonomie d'au moins une heure.	

NAOTISME_7	
Le système ne doit pas changer brutalement d'état.	

NAOTISME_8	Robustesse
Le système doit être suffisamment robuste pour résister aux enfants.	

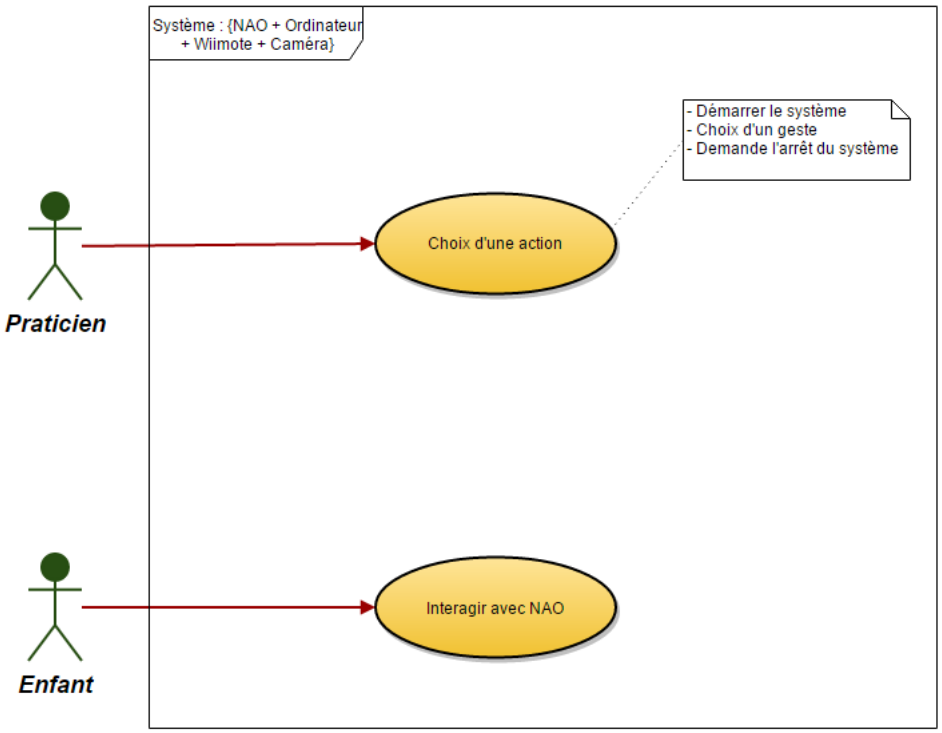
NAOTISME_9	Portabilité
Le système doit être facilement transportable.	

NAOTISME_10	Fréquence
La fréquence du mouvement doit être aussi proche que possible de celle du sujet	

Chapitre 3

Diagrammes SysML

3.1 Diagramme de cas d'utilisation



Le diagramme de cas d'utilisation a pour objectif de définir les interactions possibles avec le système, en fonction de chaque acteur isolé par le besoin.

Dans notre cas, on ne peut avoir que deux acteurs : le praticien — médecin spécialisé dans le traitement de l'autisme — et l'enfant.

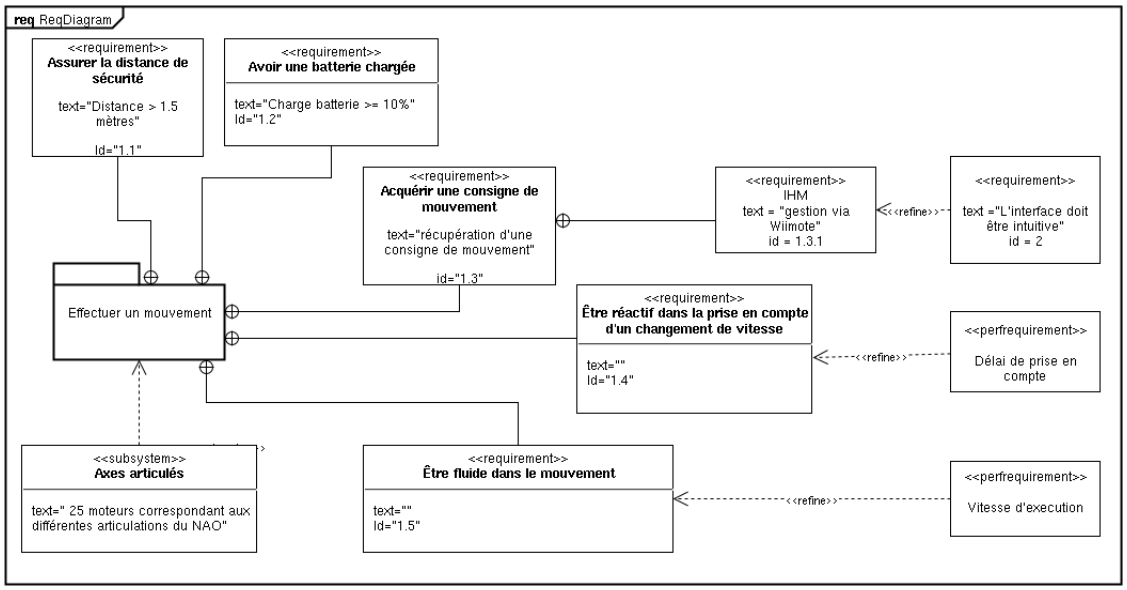
Le système avec lequel ils interagissent est constitué de NAO, un ordinateur, une Wiimote — pour le contrôle du système — et une caméra grand angle.

On distingue deux actions possibles avec le système :

— Le praticien peut choisir une action à effectuer : soit pour démarrer/arrêter le système ; soit pour changer le geste réalisé.

— L'enfant peut seulement interagir avec NAO.

3.2 Diagramme d'exigences



Pour ce diagramme, nous nous intéressons aux exigences requises pour effectuer un mouvement. Cette action en requiert cinq :

— La batterie doit être chargée à plus de 10% de sa capacité pour assurer un fonctionnement sur une durée suffisante, pour un traitement normal d'un ou plusieurs mouvements.

— L'enfant doit être à une distance suffisamment grande (de 1,5 mètres ou plus) afin d'effectuer un mouvement en toute sécurité (sans que le robot ou une partie du robot ne rentre malencontreusement en contact avec l'enfant et ne le brusque).

— Un mouvement ne peut être effectué que sur commande. Afin d'obtenir cette consigne, cela passera nécessairement par l'IHM - la Wiimote qui nous a été imposée. Cette dernière doit être user-friendly. En effet, la personne qui se chargera de faire bouger NAO sera un(e) praticien(ne). L'utilisation d'une Wiimote ne lui sera pas forcément intuitive, et notre interface doit de facto être très simple d'utilisation.

— Il est important d'avoir une bonne réactivité dans la prise en compte d'un changement de vitesse. En effet, la partie synchronisation ne doit pas perturber le

déroulement de l'interaction, afin que l'enfant ne se déconcentre pas. Le délai de la prise en compte doit être suffisamment rapide.

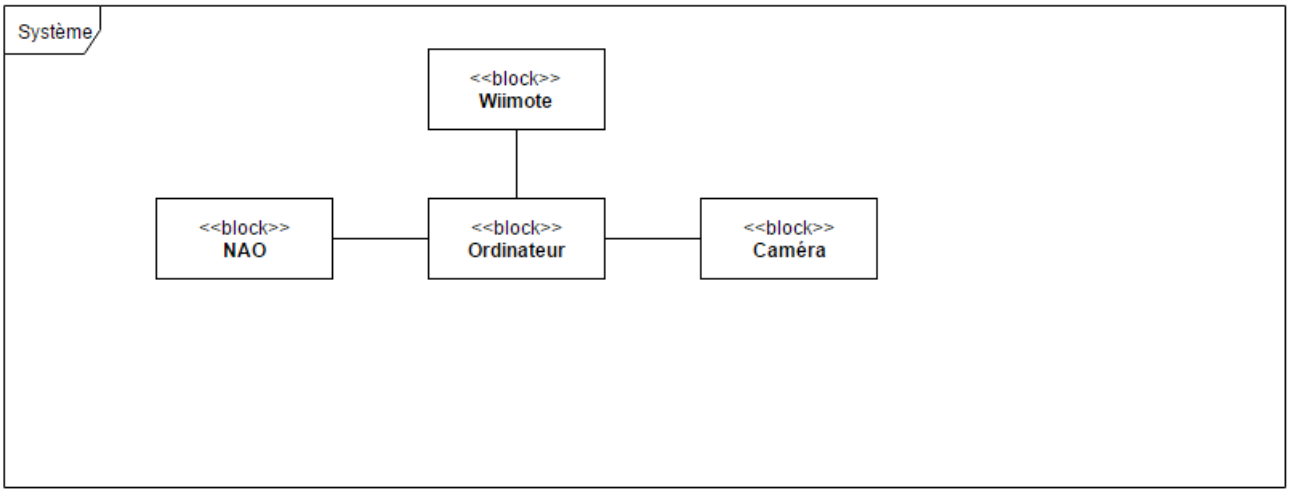
— Enfin, le mouvement doit rester fluide tout le long de son exécution afin, encore une fois, de ne pas brusquer l'enfant.

Dans le diagramme nous avons éludé les éventuels problèmes physiques, notre NAO étant assez récent et en bon état. La vitesse d'exécution doit rester stable et suffisamment rapide.

Bien évidemment, l'exécution d'un mouvement requiert un ensemble fonctionnel constitués d'axes articulés correspondant aux différents moteurs, qui permettent la mobilité des membres de NAO.

3.3 Diagramme de définition de blocs

3.3.1 Blocs principaux



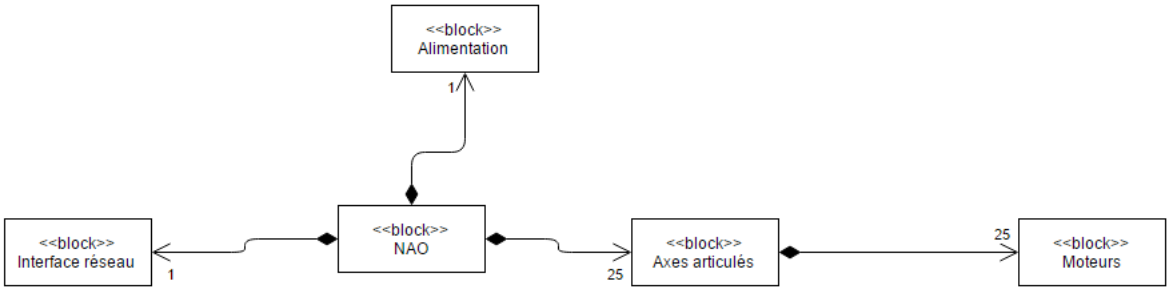
On distingue quatre blocs principaux composant notre système :

— La wiimote

— NAO

— L'ordinateur

— La caméra

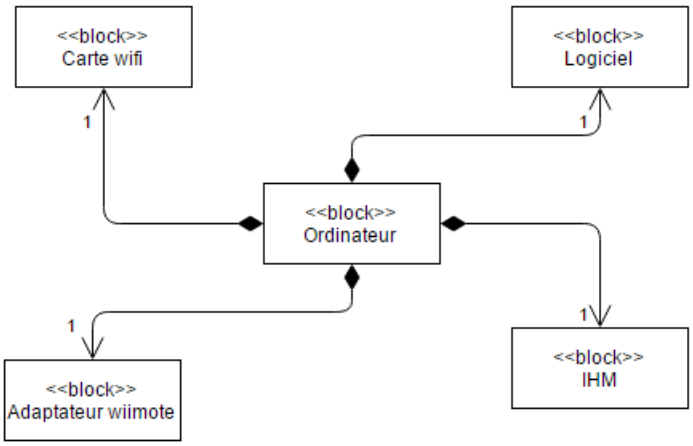


3.3.2 Détail du bloc NAO

Le bloc « NAO » est composé :

- D'une interface réseau pour la gestion des connexion avec l'ordinateur en wifi/ethernet
- D'une batterie chargée
- De 25 axes articulés comportant chacun un moteur

3.3.3 Détail du bloc Ordinateur

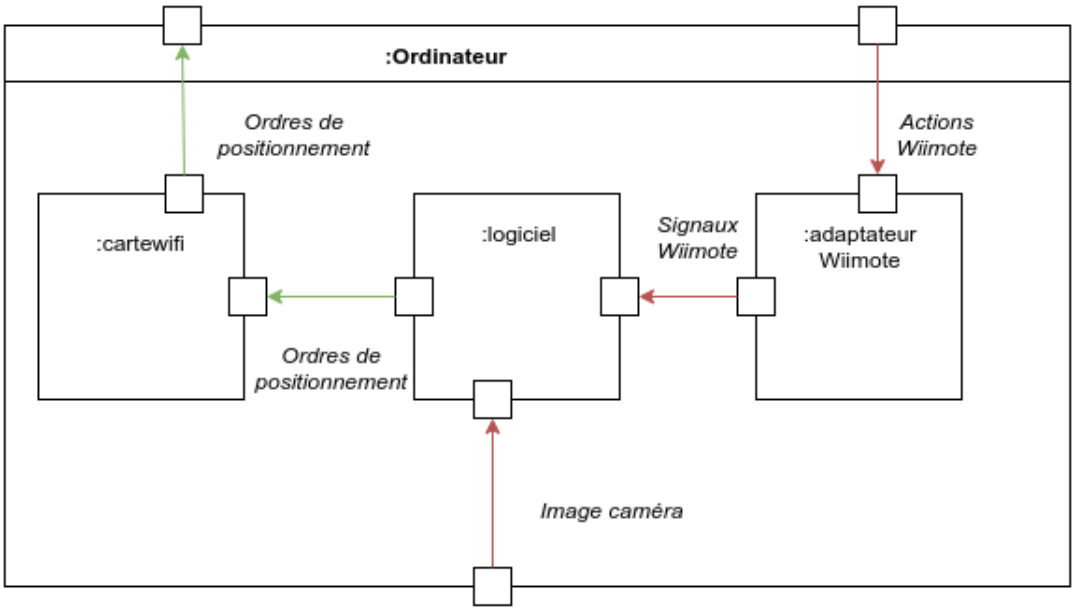


Le bloc « Ordinateur » comporte :

- Une carte wifi, indispensable pour établir la connexion avec le NAO
- Un logiciel pour gérer le comportement du NAO et traiter les données reçues par la caméra
- Un adaptateur wiimote

— Une IHM

3.4 Diagramme de bloc interne pour le bloc Ordinateur



Le diagramme de bloc interne a pour vocation de préciser les liens entre les composants internes d'un bloc défini dans un diagramme de blocs ; en indiquant les flux de données transitant entre les différents composants.

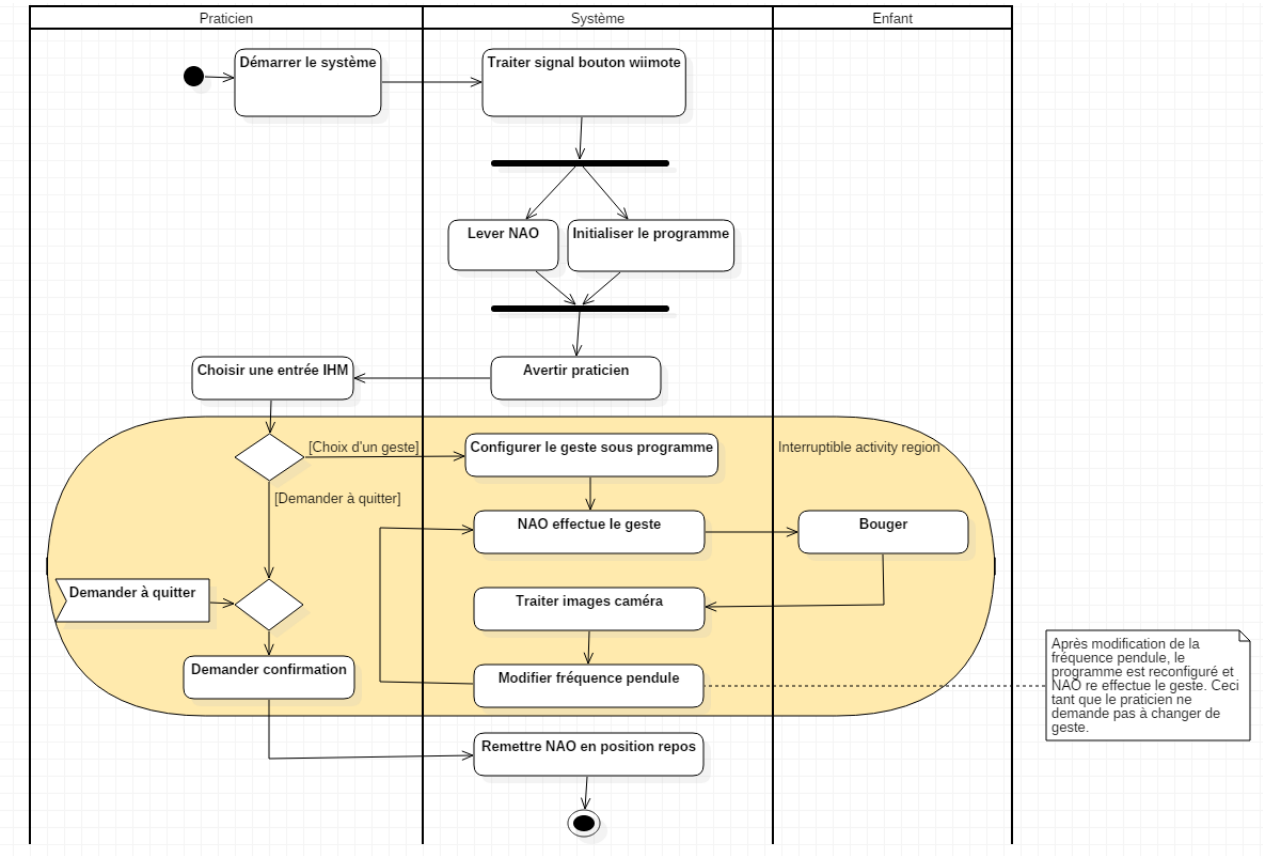
Ici, on détaille le bloc « ordinateur », tel que défini dans le diagramme de blocs ci-dessus. Celui-ci contient (pour notre usage) une carte WiFi, un adaptateur de Wiimote ; et le logiciel réalisant le lien entre NAO, la Wiimote et la caméra.

On distingue deux entrées et trois sorties de notre bloc Ordinateur :

— l'image de la caméra en flux continu (en entrée)
— les événements ou actions sur la Wiimote (en entrée)
— les ordres de positionnement envoyés au robot (en sortie)

Le logiciel récupère les actions sur la Wiimote via un adaptateur Bluethooth ; sous forme de signaux (essentiellement de type booléens - bouton pressé ou non). En parallèle, il récupère les images de la caméra grand angle, en continu. De ces informations, on obtient après traitement les ordres de positionnement à envoyer aux moteurs de NAO. Ces ordres de positionnement transitent via WiFi par la carte réseau de l'ordinateur, pour arriver vers la carte réseau de NAO (suivant le même fonctionnement en réception).

3.5 Diagramme d'activités



On distingue trois couloirs d'activités, un pour chaque responsable d'activités.

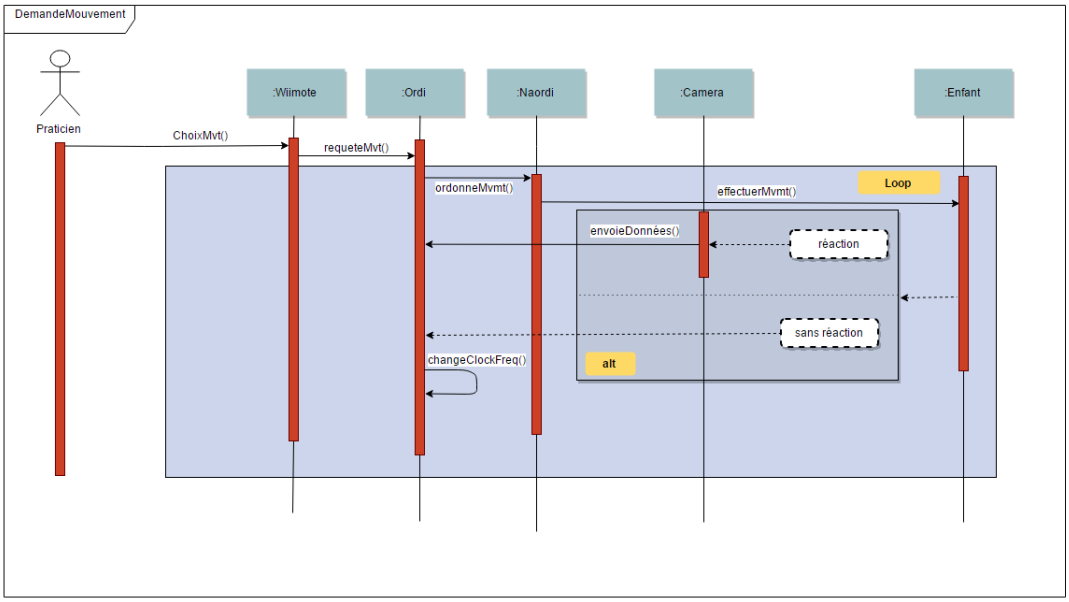
Le praticien démarre le système par le biais d'un bouton Wiimote. À la réception du signal, le système s'initialise : le programme est initialisé et le NAO levé. Une fois l'initialisation terminée, le praticien est averti par un message de bienvenue prononcé par le NAO.

Le praticien choisit ensuite le mouvement qu'il souhaite faire effectué au NAO par le biais de la Wiimote. Les possibilités de mouvement lui seront communiquées avant le début de la première séances avec les enfants. Le geste demandé par le praticien est envoyé au programme qui va effectuer les configuration nécessaires. Le programme va ensuite communiquer le geste à effectuer au NAO et ce dernier va l'effectuer. L'enfant va tenter d'imiter le mouvement effectué par le NAO, notre caméra va capturer les images et les envoyer à l'ordinateur. Le flot optique sera traité sous Prométhée et la fréquence du pendule sera adapté en conséquence. Le NAO va chercher à adapter sa vitesse d'exécution à celle de l'enfant.

Tant que le praticien ne demande pas de nouveau geste, le NAO continuera d'effectuer le même geste à vitesse plus ou moins différente selon la vitesse d'exécution de l'enfant autiste.

À tout moment, le praticien peut demander à quitter. Si la demande est confirmée, le mouvement est alors interrompu et NAO repasse en position repos.

3.6 Diagramme de séquence

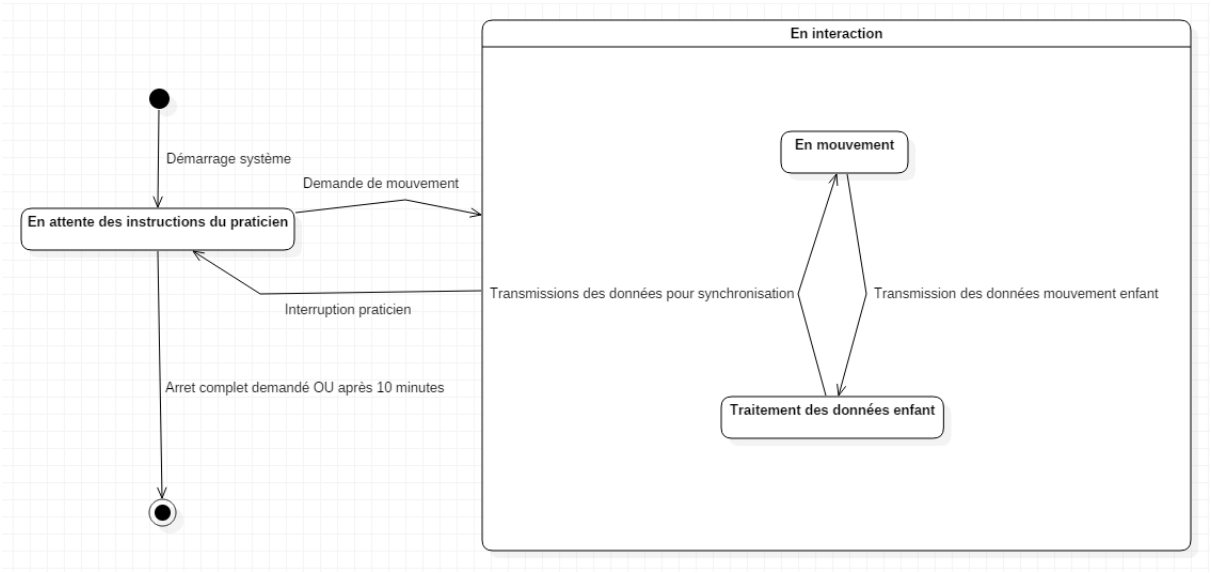


Dans ce diagramme, nous avons décidé de nous focaliser sur le traitement d'une demande d'exécution d'un mouvement par le robot NAO.

L'acteur (sur le diagramme « Praticien ») choisit le prochain mouvement à exécuter à l'aide de la Wiimote. L'ordinateur traite alors la requête qu'il renseigne à NAO (sur le diagramme « Naordi » correspondant au process interne à NAO) ; une fois le mouvement effectué la caméra récupère la réaction de l'enfant. Étant donné que nous travaillons avec des enfants — d'une part de bas âge et d'autre part atteints d'un trouble autistique — il y a de fortes chances qu'ils se déconcentrent facilement [1]. Le traitement d'image permet alors de régler la fréquence de l'horloge suivant le rythme de l'enfant. S'agissant aussi d'un exercice sur la durée, Nao sera amené à répéter plusieurs fois ce mouvement dans le but d'exercer l'enfant.

1. Malgré un travail sur la réduction des distractions et la captation de leur attention

3.7 Diagramme d'état-transition



Le système comporte deux états principaux :

- En interaction
- En attente des instructions du praticien

Après l'initialisation lors du démarrage, le système reste en attente tant que le praticien ne donne pas d'instruction.

Dès que le praticien sélectionne un geste sur la Wiimote, le système passe en mode interaction. Il s'agit d'un état assez complexe donc représenté sous forme d'état composite avec deux sous états : « En mouvement » et « Traitement des données enfant ».

NAO va d'abord effectuer un mouvement avec une vitesse d'exécution arbitraire. L'enfant va essayer d'imiter ce mouvement et les images seront transmises à l'ordinateur du système. Le système passe alors à l'état « Traitement des données enfant ». Une fois les données traitées, la vitesse d'exécution du NAO est ajustée par le biais des oscillateurs en interne selon la vitesse d'exécution de l'enfant pour le mouvement considéré.

Tant qu'une interruption praticien n'a pas lieu, le système va alterner entre les deux états précédents.

Si une interruption praticien a lieu, le système repasse en attente d'instructions. Le praticien peut demander un autre mouvement ou arrêter le système. Si aucune instruction n'est donné par le praticien, le système s'arrête automatiquement au bout de 10 minutes pour éviter de consommer inutilement de l'énergie.

Deuxième partie

Implémentation

Chapitre 4

Préambule et considérations techniques

4.1 NAO, le robot

NAO est un petit robot humanoïde de 58 cm, créé par la société française Aldebaran. Celui-ci est équipé de deux caméras, d'un processeur ATOM Z530 1.6 GHz, de microphones, sonars et autres capteurs ; ainsi que de moteurs pour les différentes articulations. [1]

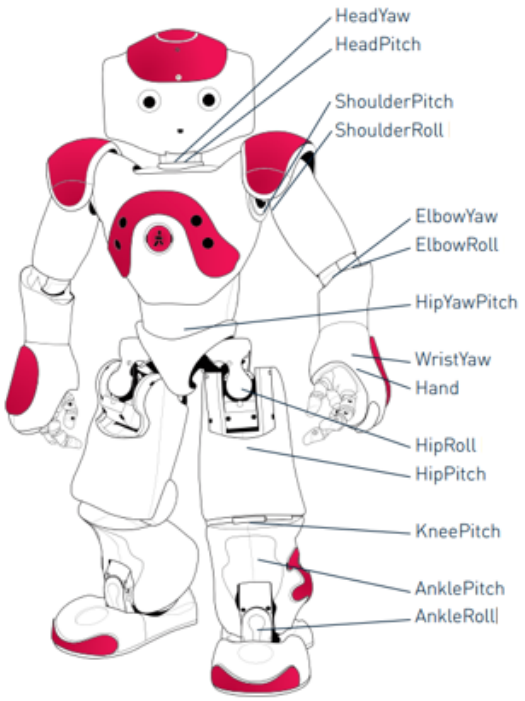


FIGURE 4.1 – Le robot Nao et ses différents moteurs

L'ETIS disposait d'un NAO de première génération (et a fait l'acquisition d'un NAO evolution — 5ème génération), et celui de l'EISTI est de la 4ème génération.

Le robot, ainsi que les autres produits d'Aldebaran — Roméo et Pepper — fonctionnent sur une distribution GNU/Linux basée Gentoo, avec une surcouche logicielle :

1. `http://doc.aldebaran.com/2-1/family/robots/index_robots.html`

NAOqi. Cette partie software est celle qui a le plus évolué entre les robots de la première et de la 5ème génération. Les différences entre la 4ème et la 5ème sont bien moindres, ce qui rend les problèmes de compatibilité quasi-nuls. De ce fait, notre travail sur le NAO de l'EISTI est totalement compatible avec le nouveau robot acquis par l'ETIS, ce qui assure une suite au projet.

4.2 Prométhée

Afin de commander Nao à distance, nous nous sommes servis du logiciel Prométhée et non de Chorégraphe (le logiciel fourni avec NAO par Aldebaran).

Il s'agit d'un simulateur de réseaux de neurones programmé en C par la branche neurocybernétique, qui ne fonctionne (à ce jour) que sous **Ubuntu 14.04 LTS**. Il permet à la fois de simuler des réseaux de neurones et de contrôler une grande variété de matériel, notamment des robots mais aussi des capteurs et effecteurs en tout genre.
Cette plateforme fonctionne avec trois autres programmes : **Coeos**, **Themis** et **Pandora**.

4.2.1 Réseau de neurones

En biologie, un neurone est une cellule nerveuse constituant la base du système nerveux et spécialisée dans le traitement des signaux électriques. Le cerveau humain contient des milliards de neurones en groupes fortement interconnectés, consituants des réseaux de neurones. Chaque neurone est une entité autonome qui fait la somme des informations qui lui parviennent, traite cette information et renvoie finalement le résultat sous forme de signal électrique. Les signaux transitent à travers des axones et les informations sont envoyées vers les autres neurones à travers les synapses et reçues par les autres neurones grâce aux dendrites.

Le traitement de l'information par chaque neurone montre que celle-ci n'est pas stockée dans les neurones mais est bien le résultat du comportement de tout le réseau interconnecté. L'information est donc principalement dans l'architecture et dans la force des connexions neuronales.

Les réseaux de neurones en intelligence artificielle sont fortement inspirés du fonctionnement du système nerveux biologique. En parallèle, chaque neurone d'un réseau sera capable d'apprendre, de mémoriser et de traiter l'information. Chaque neurone artificiel sera représenté par des entrées affectés de poids, d'une fonction d'activation qui va traiter

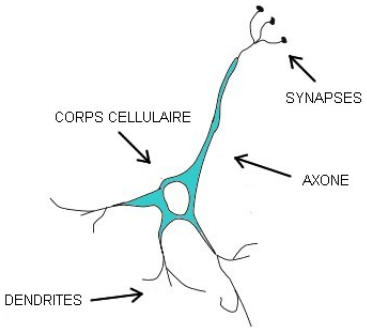


FIGURE 4.2 – Représentation d'un neurone biologique

les informations et d'une sortie. La fonction d'activation va affecter les différents poids aux liens rentrants et ainsi influencer la sortie. Les poids en entrées vont permettre au neurone d'apprendre et de modifier sa sortie en conséquence.

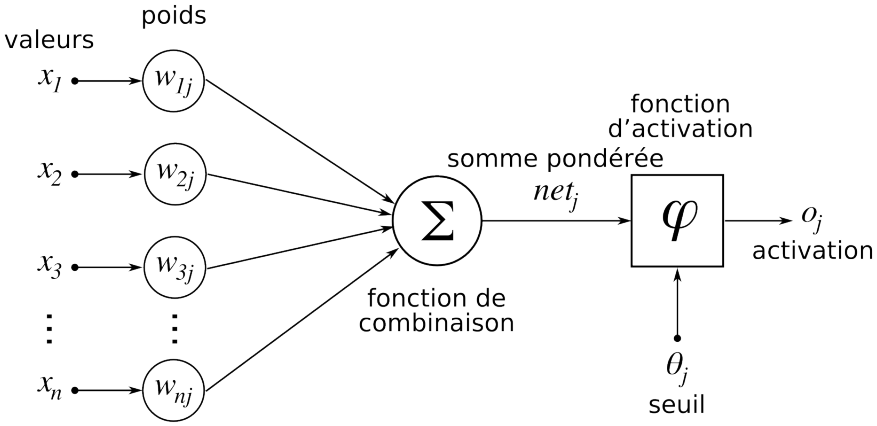


FIGURE 4.3 – Représentation d'un neurone artificiel

Comme l'illustre la figure ci-dessus, une fonction d'activation (ϕ) reçoit une somme pondérée des entrées (x_i) affectées de poids (w_{ij}) et ressort en conséquence une sortie (o_j) selon le traitement effectué en interne.

4.2.2 Coeos

Coeos est la plateforme où l'on va pouvoir construire le *script* à l'aide de boîtes. [2] Ces boîtes représentent différents neurones faisant office de fonctions de transfert et s'exécutent l'une à la suite de l'autre. Pour chacune d'entre elles, le fonctionnement et les entrées/sorties sont gérés par un code en C. Une large bibliothèque de fonctions est mise à

2. Fonctionnement similaire à celui de LabView

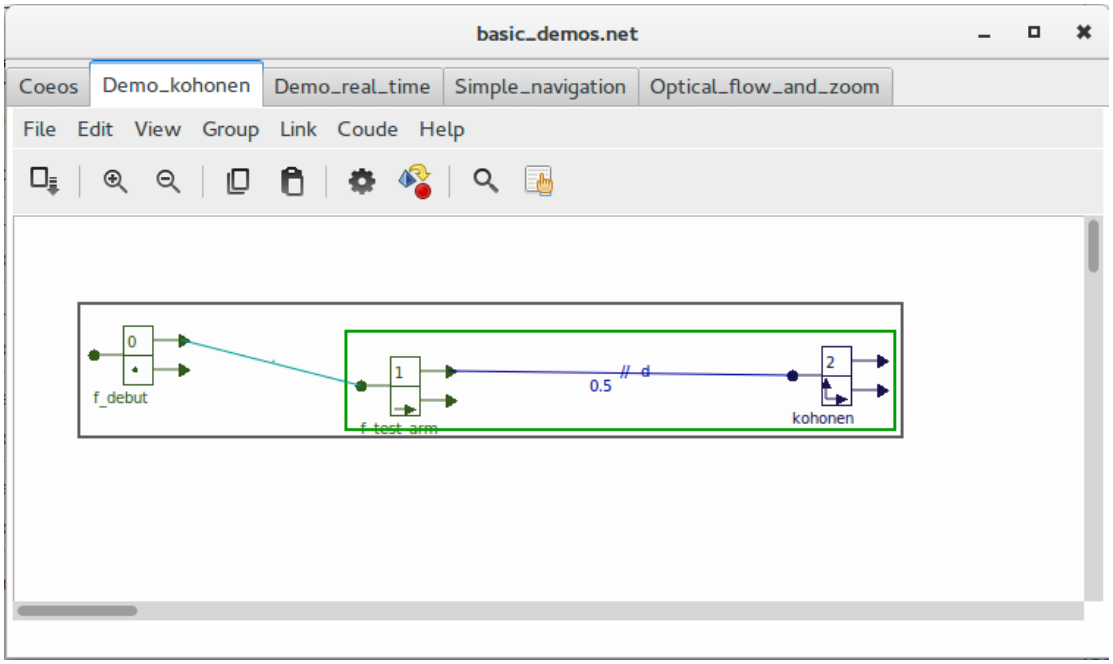


FIGURE 4.4 – Un script sous Coeos

disposition pour réaliser des opérations des plus basiques (par exemple une multiplication des entrées) aux plus complexes (comme le calcul du flot optique).

De plus, elles sont reliées entre elles par différents types de liens :

— algorithmique

— faisant la liaison entre la sortie de la boîte précédente à la boîte suivante, le tout multiplié par une norme (modifiable en cours d'execution en bleu foncé ou fixée avant l'exécution en rouge). Pour les liens « rouges », on distingue deux types de liens « one to one » et « one to all ».

— primaires (traits pleins) ou secondaires (pointillés). Contrairement aux liens primaires, les liens secondaires permettent à la boîte suivante de s'éxecuter sans tenir compte de l'état de la boîte précédente. En sortie de ce programme, on obtient des fichiers **.symb** (s'ils incluent des variables globales modifiables dans un fichier texte **.var**) sinon **.script**.

4.2.3 Themis

Themis est l'outil qui va générer les fichiers binaires exécutables finaux depuis les fichiers **.net** [3] et qui commande leur exécution.

3. écrits en XML, regroupant les scripts et les **.dev** associés

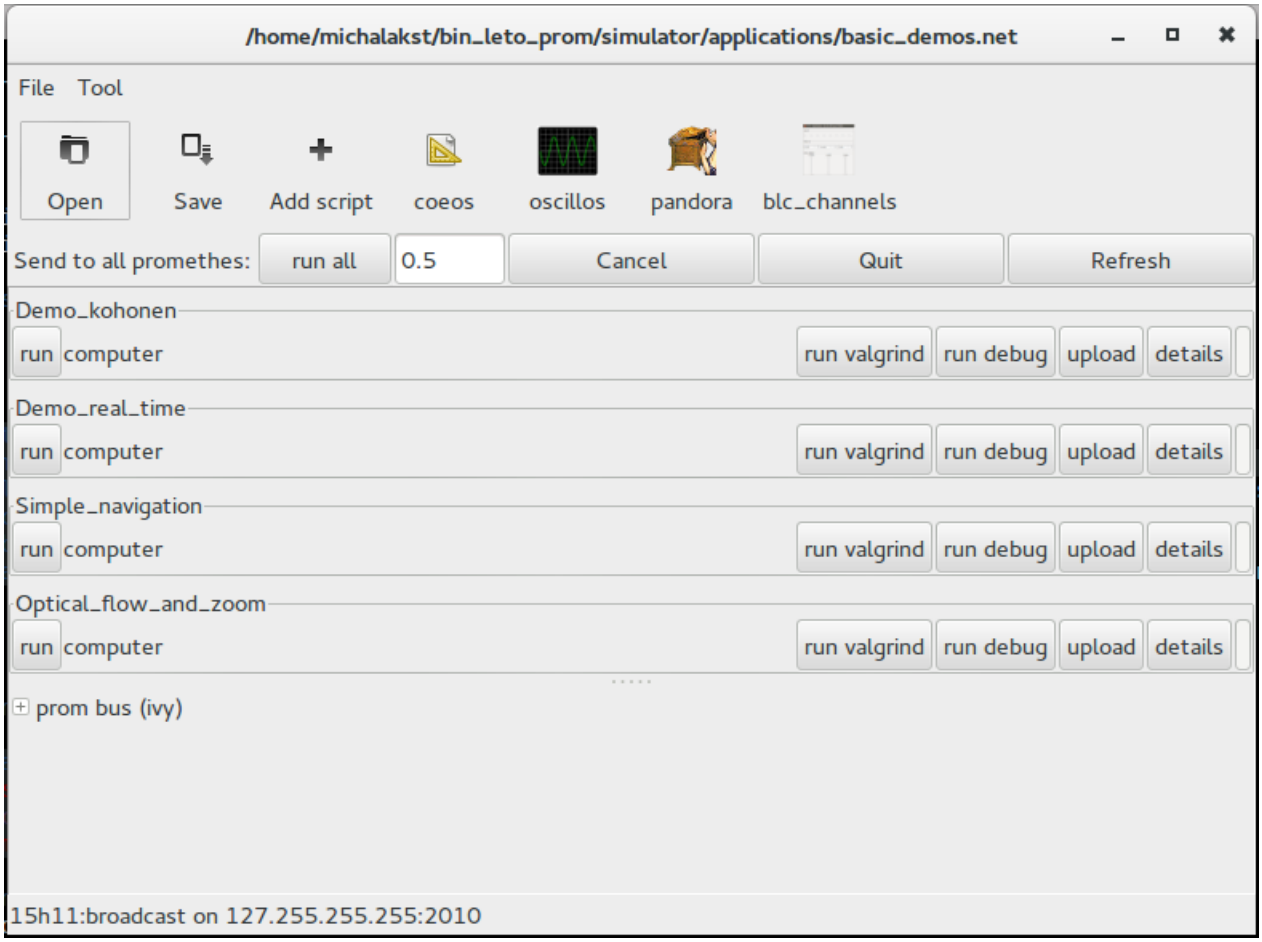


FIGURE 4.5 – Une liste de scripts sous Themis

4.2.4 Pandora

Une fois le script .net exécuté, la plateforme Pandora donne accès aux valeurs en temps réel de tous les neurones du réseau.
Il est possible de les afficher sous forme de textes, de graphiques mais aussi de les enregistrer dans un fichier texte pour une future exploitation.

4.3 Modèle de synchronisation

L'utilisation du logiciel maintenant connue, nous allons éclaircir les notions liées à la synchronisation : en effet, afin d'interagir avec un enfant, nous faisons bouger le robot de trois façons différentes ; mais de base, il effectue l'un de ces trois mouvement de façon « continue ».

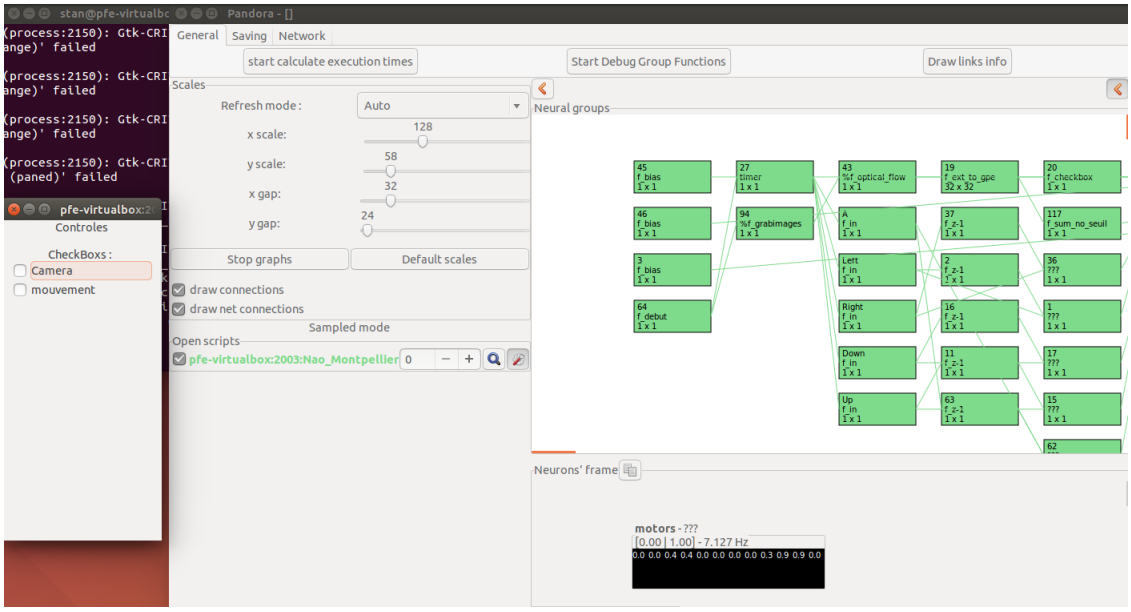


FIGURE 4.6 – L'exécution d'un script sous Pandora

4.3.1 Oscillateur

Le robot effectue un mouvement dit oscillant, c'est-à-dire dont la fréquence doit pouvoir varier en fonction de ce qu'il observera via une caméra, afin de se caler sur le rythme de l'enfant.
Pour pouvoir animer le robot ainsi, il est important de mettre en place un oscillateur. Le modèle utilisé dans le cadre du projet est relativement simple : les deux neurones s'activent et s'inhibent mutuellement, proportionnellement à la variable β selon :

$$N_1(n+1) = N_1(n) - \beta N_2(n) + \alpha_1$$

$$N_2(n+1) = N_1(n) - \beta N_2(n) + \alpha_2$$

Cet oscillateur est connecté au(x) membre(s) que nous cherchons à faire osciller à une fréquence fixe et à une amplitude qu'il est possible de faire varier.

4.3.2 Flot optique

À l'aide d'une caméra externe [4], nous pouvons visualiser les mouvements effectués par l'enfant. Ces mouvements sont ensuite traités dans la première partie de notre script grâce au flot optique.

4. La qualité de la caméra intégrée au NAO étant médiocre

Très utilisé dans le traitement d'image, ce dernier permet de décrire le mouvement apparent des motifs d'intensité de l'image sous forme d'un champ de vecteurs.

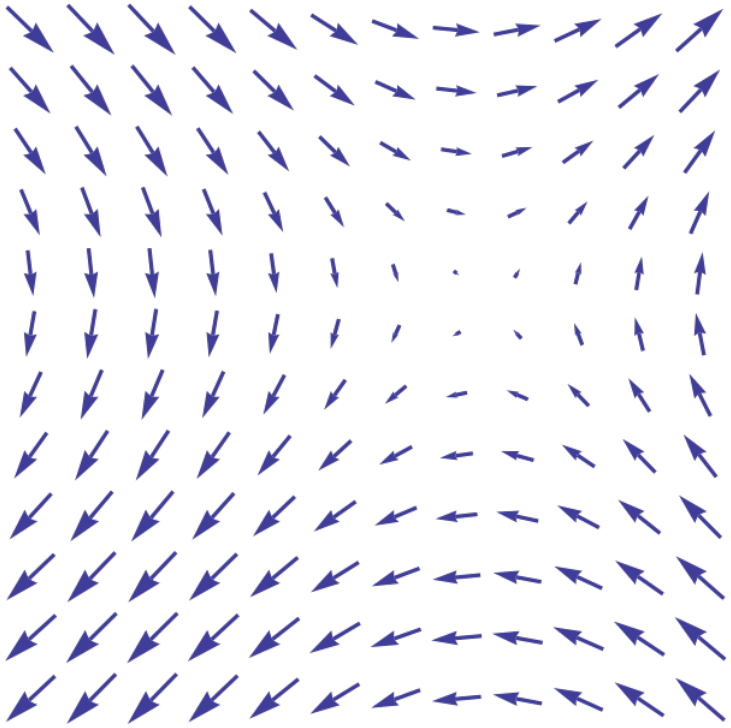


FIGURE 4.7 – Un champ de vecteurs – *source : Wikipédia*

Dans ce projet, le flot optique utilisé est calculé par des méthodes différentielles. Ces dernières sont basées sur le calcul des dérivées spatio-temporelles des intensités d'une image sur une région de l'image. Nous pouvons ainsi récupérer le flot optique vertical(vecteur $\vec{v}$) ou horizontal (vecteur $\vec{u}$).

Selon les mouvements effectués par le robot ceci va s'avérer très utile. Malheureusement, cette méthode est sensible aux bruits et aux changement lumineux (que ce soit des lumières naturelles, ou artificielles comme les néons).

Sachant que la différence d'éclairement d'un motif ne peut être dûe qu'à un mouvement dans ce motif et non à un changement d'éclairement, l'éclairement est donc un point indépendant du temps, on obtient donc :

$$\frac{\Delta E}{\delta x}u + \frac{\delta E}{\delta y}v + \frac{\delta E}{\delta t} = 0$$

avec $u = \frac{\delta x}{\delta t}$ et $v = \frac{\delta y}{\delta t}$.

Nous obtenons aussi l'équation linéaire à deux inconnues qui nous indique le mouvement de l'image :

$$E_x u + E_y v + E_t = 0$$

En sortie de la boîte, la valeur du flot optique est *positive* lorsqu'il s'agit d'un mouvement montant ou vers la droite et *négative* pour des mouvements descendant ou vers la gauche. Ce résultat est ensuite envoyé vers l'oscillateur afin de pouvoir en modifier sa fréquence.

Chapitre 5

Driver Python

Le laboratoire ETIS travaillait avec une très vieille version de NAO, et son état ne permettait pas d'envisager un travail sur le long terme, et encore moins dans un milieu où il serait confronté à des enfants autistes — dont la réaction est assez imprévisible.

De ce fait, nous avons cherché à faire fonctionner le script existant sur le robot que possède l'EISTI, un NAO de la génération précédente à l'actuelle. Nous nous sommes rendus compte que Prométhée travaillait avec un driver que proposait le robot dans les générations précédentes, mais qui n'était plus supporté sur les nouvelles générations. Plus que non-supporté, celui-ci n'est tout simplement plus présent.

Forts de ce constat, nous avons cherché à écrire un nouveau driver pour assurer le support avec le nouveau robot. En parallèle, l'ETIS a commandé un nouveau robot pour la suite des expériences, et nous devions donc nous assurer que ce driver soit totalement compatible avec le nouveau robot.

5.1 Essais en C++

Pour des besoins de performances, nous avons commencé par travailler avec le langage C++. Prométhée est fourni avec des bibliothèques qui permettent une meilleure interaction avec lui, aussi nous avons commencé par écrire une interface de communication en C++ travaillant en mémoire partagée. Ce dernier point est une exigence de l'ETIS, puisque le laboratoire cherche à harmoniser l'ensemble des communications entre le logiciel Prométhée et les autres outils.

La deuxième partie consistait à récupérer un vecteur de positions placé en mémoire partagée par Prométhée, pour les envoyer à NAO via notre driver. Nous avons donc cherché des solutions pour communiquer avec NAO :

— Une première option consistait à écrire un pilote pour l'ordinateur, et un autre pour NAO. Le procédé consiste simplement en un échange client-serveur : le serveur

(l'ordinateur) envoie les vecteurs de positions au robot, et le pilote placé sur le robot écoute en continu un port défini pour récupérer les positions.

— La seconde option — celle que nous avons choisie — consiste à utiliser le SDK fourni par Aldebaran avec ses robots. De cette façon, nous ne réinventons pas la roue, et si le système du robot évolue, il suffit de mettre à jour le SDK.

Le SDK demandait l'installation d'une toolchain pour cross-compiler le programme. Après quelques essais, nous avons réussi à la faire fonctionner. Les programmes écrits pour le robot fonctionnaient bien, mais il nous a été impossible de compiler la partie serveur sur l'ordinateur. La documentation ne nous a été d'aucune aide, et les contacts que Stanislas avait gardés suite à son stage chez Aldebaran nous ont confirmé que cette partie était très difficile à gérer, même en interne à la société.

De ce fait, et pour des raisons de temps, nous avons du nous replier sur une solution dans un autre langage bien mieux supporté : Python.

5.2 Réalisation en Python

Contrairement à la version C++, la version Python — bien que plus lente — a été simple à installer et à configurer. Prométhée n'était conçu que pour un système d'exploitation GNU/Linux, et plus précisément la version 14.04 LTS d'Ubuntu [1], nous n'avons pas eu de mal à gérer l'aspect mémoire partagée (puisque sous GNU/Linux, cela se gère comme des fichiers).

Le programme réalisé en Python se trouve en annexe.

Voici comment le driver fonctionne :

— La première étape est d'établir une session de communication avec le robot.
— Ensuite, on donne au robot une position initiale.
— On vient alors lire la mémoire partagée, pour récupérer le vecteur de positions.
— On envoie alors les positions au robot via les méthodes du SDK.

1. Voir Problèmes rencontrés **??**.

5.2.1 Détail du fonctionnement du pilote Python

Ouverture d'une session de communication avec le robot

L'écriture d'un module pour les robots d'Aldebaran commence toujours de la même manière : on utilise la lib **qi** du SDK pour ouvrir une session de communication avec le robot. Pour se faire, on utilise *qi.Application*() qui parse les paramètres envoyés au programme, de manière à extraire l'URI de connexion et le port. Ces paramètres se trouvent dans le *run.sh* fourni.

```
#!/bin/bash
python main.py --qi-url tcp://nao.local:9559
```

On lance ici le script *main.py*, en communiquant avec le robot dont l'adresse est **nao.local**, en TCP et sur le port par défaut 9559.

La session étant ouverte (si la connexion réussit à s'établir), on peut travailler avec le robot. Il ne reste plus qu'à charger notre module Python, en passant en paramètre l'instance d'application.

```
if __name__ == '__main__':
    app = qi.Application()
    app.start() # Start session

    nao_caller = NaoCaller(app)

    # [...]

    app.run()  # block until session close or ctrl+c
```

Le reste de l'initialisation du module consiste juste à stocker une référence vers des services du SDK pour un accès rapide :

```
self._load_services({
            "tts": "ALTextToSpeech",
            "motion": "ALMotion",
            "posture": "ALRobotPosture"
        })
```

Ici, on charge trois services :

ALTextToSpeech qui permet de faire parler le robot

ALMotion qui contrôle (entre autres) les positions des moteurs du robot.

ALRobotPosture un service plus haut niveau permettant de positionner le robot dans des états pré-définis (debout, assis, etc.)

Initialisation

Dans un premier temps, on cherche à positionner le robot de façon stable. Pour se faire, on envoie simplement des positions initiales aux moteurs.

```
def _init_position(self):
    self.services['motion'].wakeUp()
    self.services['posture'].goToPosture("Stand", 0.5)
    self.move('LShoulderPitch', 1.5, 0.25, False)
    self.move('RShoulderPitch', 1.5, 0.25, False)
    self.move('LShoulderRoll', 0.4, 0.25, False)
    self.move('RShoulderRoll', -0.4, 0.25, False)
```

L'envoi des positions ne se fait pas directement, on utilise pour cela une méthode **move** que nous avons écrite. Celle-ci effectue deux opérations avant d'envoyer les positions :

— dé-normaliser les positions au besoin (les positions envoyées par Prométhée étant normalisées)
— contrôler l'existance du moteur demandé et vérifier que la position se trouve bien dans la plage autorisée.

Lecture de la mémoire partagée

Les positions des différents moteurs sont définies sous Coeos. Pour chaque mouvement on utilise douze moteurs différents : deux moteurs pour l'épaule droite, deux moteurs pour le coude droit, un moteur pour le poignet droit, un moteur pour le genou droit et par symétrie l'équivalent de ces moteurs du côté gauche.

L'ensemble des positions sont ensuite concaténées dans un vecteur de position. Ce vecteur est écrit dans un segment de mémoire partagée avec le pilote Python. Ce dernier

va donc lire en continu le segment de mémoire partagée pour récupérer les positions des différents moteurs, continuellement mis à jour.

```
def listen_shared_memory(self):

    speed = 0.25

    print "Listening shared memory..."
    while True:
        try:
            data = np.memmap(self.SHM_MOTORS_FILE, dtype=np.
                float32, mode='r')
            for i in xrange(0, 12):
                self.move(self.MOTORS_ORDER[i], float(data[i]),
                    speed)
        except IOError:
            print self.SHM_MOTORS_FILE + " not found"
        time.sleep(0.05)
```

Pour lire la mémoire partagée, on utilise la bibliothèque Python numpy [2]. Celle-ci fournit une méthode **memmap** qui permet de lire un vecteur de valeurs en mémoire partagée [3]. Voici les paramètres utilisés :

filename Le chemin du pseudo-fichier de mémoire partagée

dtype Le type des données à récupérer (ici du float 32 bits)

mode Le mode de lecture (comme celui d'un fichier classique — r pour lecture seule, r+ pour lecture et écriture d'un fichier existant, etc).

Sachant que le vecteur de positions fait toujours la même taille, on la fixe à 12, et on lit les positions dans l'ordre. Cet ordre est fixé dans la configuration du pilote, dans le tableau **MOTORS_ORDER**. Pour chaque position, il ne reste qu'à envoyer l'ordre de mouvement au moteur via la méthode **move**.

En cas d'erreur de lecture du segment de mémoire partagée, on indique que le fichier n'est pas lisible/trouvable et on continue de boucler jusqu'à la fermeture du programme.

2. http://www.numpy.org/
3. http://docs.scipy.org/doc/numpy/reference/generated/numpy.memmap.html

Envoi des positions au robot

Les positions normalisées lues en mémoire partagée sont converties en positions angulaires puis envoyées au robot via la méthode **setAngles** du service ALMotion fourni par le SDK. Le robot exécute ainsi le mouvement demandé.

```
def move(self, name, position, speed, normalize=True):
    if normalize:
        real_position = self._unnormalize_pos(name, position)
    else:
        real_position = position
    if self._check_motor(name, real_position):
        self.services['motion'].setAngles(name, real_position,
            speed)
    else:
        print "Invalid move " + name + ": " + str(real_position)
```

Chapitre 6

Script sous Prométhée

Dans le cadre de nos expériences, nous devions faire faire à NAO des gestes simples comme un « Bonjour » de la main, se toucher la tête ou lever les bras mais de façon continue pour que l'enfant reproduise ce geste.

L'arrivée de nouvelles collaborations (avec un praticien) nous poussa à améliorer l'Interface Homme-Machine (IHM) qui était peu ergonomique jusqu'alors. Un de nos objectifs fut donc de rendre possible l'utilisation de la Wiimote qui agit comme une télécommande, afin de passer d'un mouvement à un autre et de modifier deux paramètres. Il s'agit d'un outil simple à prendre en main et peu encombrant pour l'utilisateur (ici praticien).
Notre script est donc divisé en trois parties : le traitement de la synchronisation, le contrôle des moteurs du robot et la gestion des commandes de la Wiimote.

6.1 Synchronisation

Il s'agit de la partie où l'on cherche à rapprocher la vitesse d'exécution du robot à celle de l'enfant. Il faut donc que NAO puisse « voir » l'enfant et analyser ses mouvements.
Cette partie a déjà été traitée par le groupe précédent, mais a été améliorée par Eva Ansermin pour son projet de 2ème année. Nous avons donc récupéré son travail et n'avons ajouté qu'une petite modification à celui-ci. En effet, étant donné que les mouvements ne sont pas forcément verticaux (du haut vers le bas ou inversement), nous traitons deux cas : l'un où le flot optique est calculé par rapport à la verticale et l'autre par rapport à l'horizontale.

6.2 Mouvements

6.2.1 Utilisation de la Wiimote

Afin d'améliorer l'Interface Homme-Machine (IHM) du projet, et pour qu'il soit utilisable pour une personne néophyte ; il a été décidé d'utiliser une Wiimote : la manette de la Wii de Nintendo®.

Cette manette comporte un bouton A, un bouton B, un joystick et un pavé directionnel. Dans notre cas, nous devions pouvoir choisir un mouvement à exécuter dans une liste de mouvements définis, pouvoir changer la fréquence d'oscillation — c'est-à-dire la vitesse d'exécution du mouvement en cours — et le facteur de couplage — le paramètre qui détermine la capacité d'adaptation au rythme de l'enfant.

Pour changer de mouvement, nous avons utilisé le bouton A de la Wiimote : on boucle simplement sur la liste des mouvements en appuyant sur le bouton. La fréquence d'oscillation peut être changée via les flèches directionnelles gauche (pour diminuer) et droite (pour augmenter). De même, le facteur de couplage utilise les flèches directionnelles haut et bas.

Le service Wiimote

La lecture de la Wiimote se fait grâce à un service qui nous a été fourni. Le principe est le suivant : le service s'associe à une Wiimote via un adaptateur Bluetooth — on parle d'appairage. Une fois la Wiimote et l'ordinateur en communication, le service écoute en boucle les événements de la Wiimote (changements d'état des boutons et capteurs). Cet état est mis à jour et stocké dans un segment de mémoire partagée */wii*.

Lecture des valeurs sous Prométhée

Une fois l'état stocké en mémoire partagée, on récupère ses différentes composantes avec la boîte *f_ in* en précisant la composante qui nous intéresse.

La figure 6.1 donne le détail de la lecture du bouton A de la Wiimote, qui est décomposée en 5 boîtes :

La boîte A est une fonction *f_ in* qui vient lire la composante A du segment de

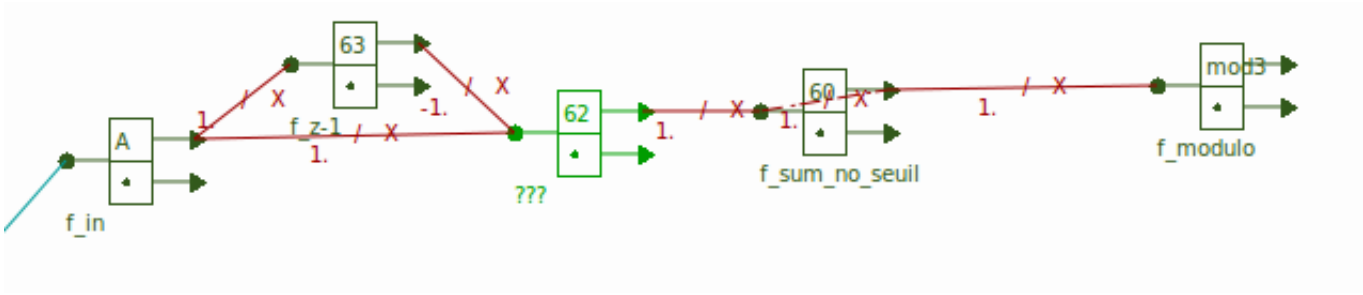


FIGURE 6.1 – Lecture de l'état du bouton A de la Wiimote

mémoire partagée */wii*. On récupère alors la valeur du bouton.

La boîte 63 est une fonction *f_ z-1* qui va lire la valeur de l'itération précédente.

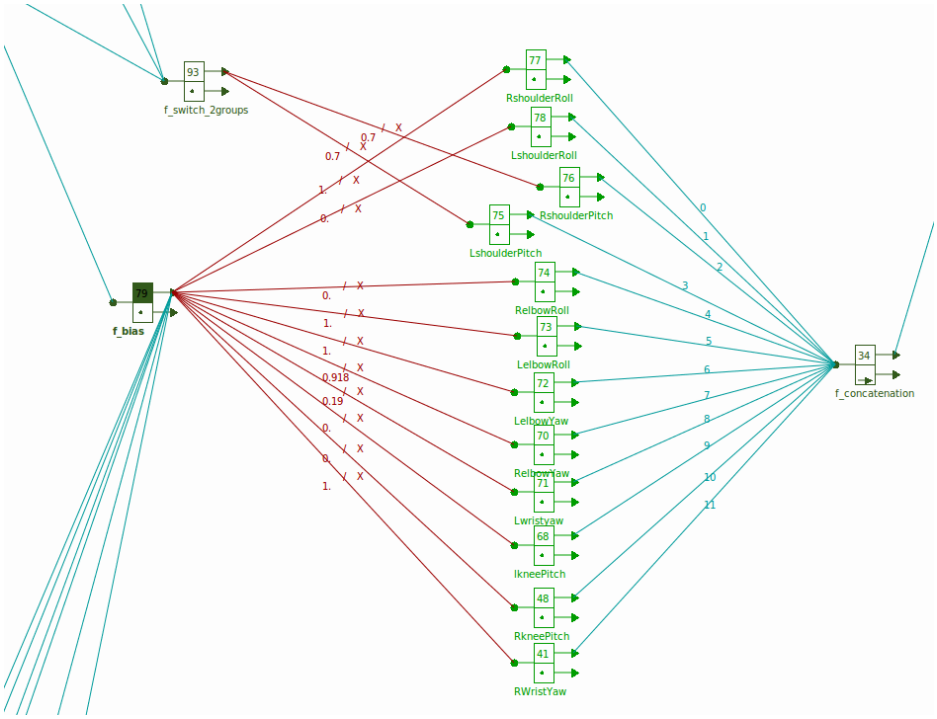
La boîte 62 est une fonction de *Hebb*.

La boîte 69 est une fonction somme non-seuillée.

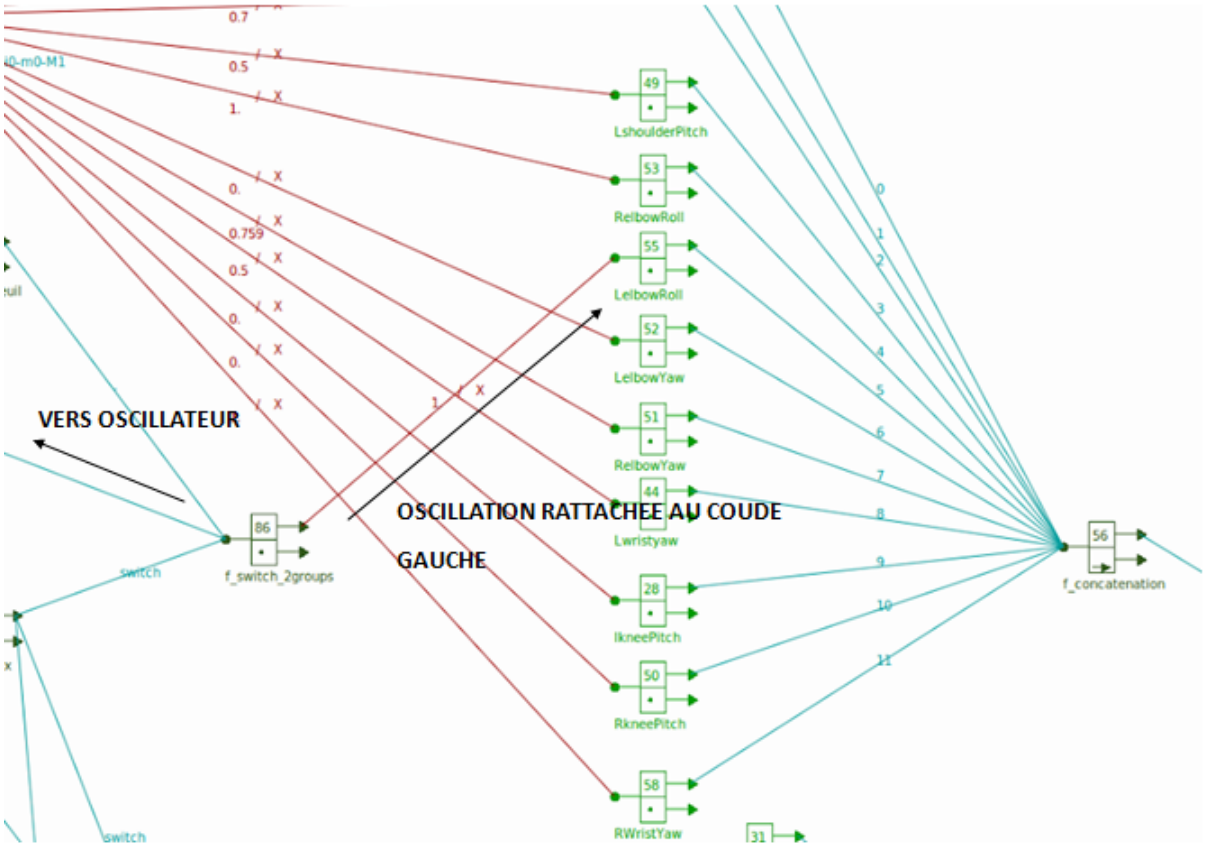
La boîte mod3 enfin est une fonction modulo.

6.2.2 Contrôle moteur

Comme présenté par le premier groupe, les moteurs sont controlés par des boîtes (une boîte par moteur) qui envoient une position angulaire. Ces positions sont alors concaténées par la boîte *f_ concatenation* et envoyées en sortie pour être écrites en mémoire partagée.



Nous avons alors défini trois positions de départ pour trois mouvements différents. Nous connectons l'oscillateur aux moteurs qui doivent bouger continuellement, ce sont les moteurs concernés par la synchronisation. Par exemple, dans le cas du « Coucou », l'oscillateur sera connecté à la boîte correspondant au coude.



Chapitre 7

Problèmes rencontrés

Au cours de notre projet de fin d'études, nous avons du faire face à un certain nombre de problèmes, tant techniques que pratiques.

7.1 Problèmes techniques

7.1.1 Incompatibilité NAO nouvelle génération

Lors des premières séances de travail, nous avons cherché à prendre en main les outils mis à notre disposition, et à faire fonctionner l'existant.

Le script fonctionnait bien sur le robot que possède l'ETIS, un NAO de première génération, mais ne fonctionnait pas sur le robot de l'EISTI — un modèle de la $4^{\text{ème}}$ génération. De ce fait, notre mission fut de trouver une solution pour faire fonctionner le script sur le nouveau robot, en prévision de l'achat d'un nouveau robot par l'ETIS.

Nous avons trouvé la cause de l'incompatibilité du script, ou plutôt de Prométhée, avec les nouvelles générations : ces dernières n'utilisent plus la même couche logicielle de communication, et le support pour le pilote alternatif — URBI — utilisé par Prométhée n'est plus d'actualité depuis au moins 3 ans.

7.1.2 Compatibilité restreinte de Prométhée

Prométhée est un logiciel développé par l'ETIS. De ce fait, c'est le laboratoire qui en assure le support. Ce dernier est restreint, pour des questions de temps et d'argent consacrés au projet.

Le support de la suite logicielle n'est donc assuré que sur le système d'exploitation Ubuntu, et sur une version spécifique : Ubuntu 14.04 LTS. Nous avons essayé de la faire

fonctionner sur Debian Jessie et sur Ubuntu dans les versions supérieures, mais un bug ne nous a pas permis de continuer dans ce sens. De plus, les essais en utilisant VirtualBox sur la bonne version d'Ubuntu n'ont été qu'à moitié concluants : les performances n'étaient pas très bonnes, et la communication avec la caméra et avec le robot ne fonctionnaient pas bien.

La seule solution a donc été de prévoir une installation séparée de cette version d'Ubuntu.

7.1.3 SDK C++ mal supporté

Lorsque nous nous sommes attelés au développement du nouveau pilote de communication avec NAO, un des impératifs était de rendre la solution pérenne. Pour éviter de réinventer la roue, nous avons utilisé le SDK fourni par Aldebaran avec le robot : NAOqi. De cette manière, les changements à effectuer à l'avenir seront documentés par l'entreprise.

Pour des besoins de performances, nous avons cherché à utiliser le SDK C++. Après quelques essais d'installation de la toolchain, nous avons réussi à exécuter des programmes pour une cible Atom (le processeur équipant NAO) ; mais la compilation pour une cible x86 n'a pas fonctionné, et ce même pour les exemples fournis avec le SDK.

Nous nous sommes renseignés auprès d'Aldebaran, qui nous ont répondu que le SDK C++ était encore mal supporté, et que seul un produit l'utilisait pour une cible x86 : Choregraphe, leur éditeur de scripts. Le problème étant que même ce logiciel devait passer par des astuces pour le faire fonctionner.

Pour éviter de perdre plus de temps, nous nous sommes tournés vers la solution actuelle, utilisant le SDK Python.

7.1.4 Pannes matérielles

Au cours des essais pour faire fonctionner le nouveau pilote, nous avons essuyé deux pannes matérielles ; ce qui a eu pour effet d'immobiliser le robot pour au moins une semaine à chaque fois. Nous avons toutefois réussi à faire jouer les relations de Stanislas, qui a effectué un stage chez Aldebaran l'été 2015, pour faire réparer le robot rapidement.

La première panne est survenue lors d'une chute du robot de l'EISTI : le robot n'est

pas tombé de très haut, mais il ne pouvait plus plier la jambe correctement ; ce qui était handicapant pour les expériences.

La seconde panne s'est produite juste avant de réaliser la première expérience avec un enfant autiste. Celle-ci a touché le robot que venait d'acheter l'ETIS, qui avait été utilisé le week-end juste avant. Un des engrenages du bras du robot tournait dans le vide, ce qui immobilisait le bras. Nous avons choisi d'utiliser le robot de l'EISTI pour pouvoir faire au moins une expérience.

Le dernier problème s'est produit lors de cette solution de secours : le robot de l'EISTI n'avait plus beaucoup de batterie (peut-être par usure de celle-ci), et de ce fait, la communication par WiFi n'a pas pu se faire.

7.2 Problèmes pratiques

7.2.1 Accès à l'ETIS

L'accès aux laboratoires de l'ETIS est sécurisé par un badge, que nous avons mis plusieurs semaines à obtenir. Cette sécurité s'est renforcée suite aux attentats, et il a été encore plus difficile de se rendre aux laboratoires : il était nécessaire que d'autres personnes soient présentes, même en possédant le badge.

En outre, nous n'avons eu qu'un seul badge pour trois personnes, ce qui impliquait des retards si la personne qui avait gardé le badge n'était pas présente à la même heure.

7.2.2 Disponiblité des enfants

Comme nous travaillions dans un but d'aider les enfants autistes, nous avions besoin de ces derniers pour réaliser nos expériences. Il a été difficile de trouver des créneaux sur lesquels les enfants pouvaient être disponibles, et un temps d'adaptation et d'acceptation du robot et de la praticienne qui travaillait avec nous a été nécessaire.

De ce fait, nous n'avons pu commencer les expériences qu'assez tard. Celles-ci continuent néanmoins grâce à nos travaux, et les retours sont pour le moment très positifs.

7.2.3 Gestion d'autres projets

Outre les problèmes liés au projet, nous avons du gérer un problème de disponibilité : Stanislas travaillait en effet sur le projet AREL, pour le compte de l'EISTI ; et n'était donc pas tout le temps disponible. Cela a empiré sur la fin du projet, puisque l'entretien d'AREL et la passation de la direction du projet — qu'il assurait seul — est devenu très chronophage.

Nous avons donc du nous débrouiller sans lui sur plusieurs séances, et il a fallu trouver des créneaux de travail supplémentaires pour terminer le projet à temps.

7.3 Organisation des séances

7.3.1 Locaux

Pour des raisons pratiques, la grande majorité de nos séances ont eu lieu à l'Université de Cergy-Pontoise dans les locaux de l'ETIS. Pour accéder à ces derniers, nous avons effectué les démarches nécessaires pour obtenir un [1] badge pour le groupe. Malheureusement, ces démarches ont pris beaucoup de temps. Cela a eu un impact sur notre projet puisque nous étions dépendant des membres de l'ETIS pour accéder aux locaux avant l'obtention du badge et suite aux évènements en Novembre 2015, l'accès à l'ensemble de l'Université fut restreint et contrôlé.

7.3.2 Séance de mise en pratique

Afin de mettre en pratique nos réalisations, le partenariat avec l'association a été relancé. Nous étions donc dépendant de l'emploi du temps des enfants (de leur parent mais aussi des représentants de l'association) pour lancer les expérimentations.

Ces expérimentations n'ont commencé que depuis le 13 Avril 2016 (la tentative du 11 avril ayant été avortée suite un dysfonctionnement du WiFi de notre robot lié à la dégradation des performances de la batterie).

1. seul et unique

Conclusion

— L'utilisation de la wiimote comme contrôleur à disposition d'un praticien, afin que ce-dernier dispose d'un outil très simple d'utilisation pour animer ses séances
— La réduction des distractions en connectant notamment Nao via le WiFi au lieu d'une connexion cablée
— L'intégration de trois mouvements : *Coucou*, *Toucher la tête* et *Lever les bras*. Le passage entre les différents mouvement a été rendu le plus simple possible : en pressant la touche A de la wiimote le praticien alterne entre les mouvements dans un ordre donné.
— Le passage au support pour les Nao v4-v5 au lieu d'un V1 auparavant à l'aide d'un driver python

Malgré la quantité considérable de problèmes rencontrés au cours du projet, nous avons donc pu atteindre les objectifs que nous nous étions fixés avec l'ETIS.

Nous sommes toutefois déçus de ne pas avoir pu assiter aux expériences menées avec les enfants en collaboration avec l'association Autisme Ensemble 95. Les expériences ont effectivement commencées quelques jours après notre départ, sur la base de nos travaux, repris par l'ETIS.

Bibliographie

[1] **Socially-assistive robots help children with autism learn imitative behavior by providing personalized encouragement**, http://viterbi.usc.edu/news/news/2014/august-28-2014.htm

[2] https://fr.wikipedia.org/wiki/Estimation_de_mouvement

[3] http://www.autisme-france.fr/

[4] https://www.ald.softbankrobotics.com/fr

[5] **Pilote : Thérapie robotique pour l'autisme**, Fanny Larradet, Mylene Le Hen et Aurelie Fajardo

[6] **Synchronie intentionnelle et non intentionnelle pour l'imitation des gestions : application aux interactions entre un robot NAO et un enfant autiste,** Éva Ansermin

Table des matières