



MASTER MIASHS « HANDI »
Parcours : Technologie et Handicap

Domaine : Sciences, Technologie, Santé (STS)

Mémoire de Master

FacePal

Un robot compagnon convivial et autonome pour les enfants atteints de troubles neurodéveloppementaux

Massyl Touat

Enseignant référent : Dominique Archambault

Encadrant : Salvatore A. Anzalone

Paris, le September 12, 2024



Laboratoire Chart

FacePal

Un robot compagnon convivial et autonome pour les enfants atteints de troubles neurodéveloppementaux

Massyl Touat

le 12 septembre 2024

Table des matières

1	Introduction	3
1.1	Etat de l'art	4
1.1.1	Robotique sociale pour les TND	4
1.2	Objectifs	11
2	Méthode	12
2.1	Hardware	12
2.1.1	Conception générale	12
2.1.2	Joints et moteurs	13
2.2	Contrôle de bas niveau	17
2.3	Visages expressifs	20
2.3.1	iClone et personnages	21
2.3.2	Expressions faciales	22
2.3.3	Contrôle de l'image	24
2.4	Discours	25
2.4.1	Reconnaissance vocale	25
2.4.2	Synthèse vocale	25
2.5	Dialogue ouvert	26
2.5.1	Contrôle de haut niveau via OpenAI	26
2.5.2	Personnalité artificielle	27
3	Résultats	28
4	Discussion	34

Annexes	39
A Code	39

Chapitre 1

Introduction

Les troubles du spectre de l'autisme (TSA) débutent précocement durant le développement de l'enfant, dès ses premiers mois, et entraînent des difficultés dans les sphères personnelle, sociale, scolaire ou encore professionnelle. Elles se caractérisent par des difficultés dans la communication et l'interaction sociale et par des comportements restreints et répétitifs [Reis, 2023]. Les personnes ayant des TSA peuvent présenter plusieurs comorbidités : des déficits intellectuels, des déficits d'attention avec ou sans hyperactivité (TDA/H) (Clark et Bélanger, 2018), des troubles spécifiques des apprentissages, ou encore des troubles de la communication [American Psychiatric Association, 2023].

Les avancées et les progrès actuels en robotique offrent des alternatives crédibles aux méthodes conventionnelles dans le cadre de la thérapie du TSA. Les robots ont l'avantage d'être répétitifs et prévisibles [Diehl et al., 2012] ce qui est rassurant pour les enfants atteints de TSA, souvent sensibles aux changements [Feil-Seifer and Matarić, 2008]. Les expressions des robots sont en général plus simples que les humains [Pennisi et al., 2015] encourageant ainsi l'enfant à poursuivre pleinement sa démarche d'interaction avec le robot. L'un des principaux atouts de la robotique dans un tel cadre est sa capacité à proposer des interactions contrôlées grâce à la variété des scénarios possibles. De plus, ces scénarios peuvent être des jeux ludiques et captivants, augmentant ainsi l'implication des enfants et réduisant les comportements de retrait tout en permettant d'accroître la participation active [Kim et al., 2012].

Durant les deux dernières décennies, plusieurs études ont été menées sur l'utilité de la robotique dans la thérapie du TSA. Les technologies et les méthodes d'approches ont toutes évolué à travers ces années et ont permis de bâtir tout un écosystème collaboratif entre chercheur-patient-parent.

1.1 Etat de l'art

1.1.1 Robotique sociale pour les TND

De plus en plus d'intérêt est dirigé vers l'utilisation de la robotique pour l'aide à la thérapie des troubles neurodéveloppementaux, et plus particulièrement le TSA. En effet, plusieurs études indépendantes font état de 60 à 70 cas sur 10000 (soit 0,6% à 0,7%)¹ démontrant une prévalence plus importante. La nature des symptômes du TSA varie d'un enfant à un autre, mais en général, les enfants qui en sont atteints éprouvent des déficiences cognitives et motrices, des difficultés à traiter les informations sensorielles et dans certains cas un retard du développement du langage [Chen et al., 2021]. Ces symptômes peuvent significativement nuire aux activités quotidiennes de l'enfant, en particulier les activités sociales [Anzalone et al., 2018]

La notion de **robotique sociale** fait référence à tout robot capable d'interagir socialement avec les individus, que ce soit par la parole, les gestes ou d'autres moyens significatifs [Scassellati et al., 2012]. Durant ces 20 dernières années, La **robotique d'assistance sociale (SAR)**, rencontre de la robotique sociale et de la robotique d'assistance², se concentre sur la création et la mise en oeuvre de robots afin d'assister l'humain par l'interaction sociale plutôt que physique [Scassellati et al., 2012].

Selon la revue littéraire [Santos et al., 2023], les stratégies thérapeutiques conventionnelles pour ce trouble encouragent cette interaction sociale dans des scénarios trop contrôlés et très standardisés, les rendant ainsi non reproductibles dans la pratique quotidienne, la SAR s'est alors vite distinguée comme étant une alternative thérapeutique intéressante. Dans de nombreux cas, l'enfant adopte des comportements sociaux plus appropriés avec les robots qu'avec les adultes [Scassellati et al., 2018] et les technologies actuelles permettent aux robots d'être stylisés dans leur apparence et d'adopter des comportements répétitifs si besoin, ce qui les rend plus confortables émotionnellement et plus prévisibles, tirant ainsi partie des tendances des enfants atteints de TSA [Diehl et al., 2012]. Dans cette même revue, il est souligné que 75% des études incluaient un robot d'apparence humanoïde. Rassurant et proche du réel, l'aspect du robot est le premier signal perçu par l'enfant, cette apparence anthropomorphique³ aide à rendre les comportements plus distinguables, facilitant ainsi l'interaction avec l'humain [Anzalone et al., 2015].

Lors de la conception d'un SAR, d'après [Dautenhahn, 2021] lors de l'ICRA⁴ 2020, cinq stratégies de contrôle peuvent être envisagées. Wizard of Oz (WOZ)⁵ : le robot est contrôlé par un opérateur caché. La stratégie de contrôle hybride : le robot peut avoir quelques aspects autonome, mais l'utilisation finale en général se fait par l'adulte ou l'enfant. Le modèle autonome : le robot ne nécessite aucune intervention pour fonctionner. Enfin, le modèle autonome adaptatif : le robot

1. <https://www.autismeinfoservice.fr/adapter/essentiel/chiffres-statistiques> : études faites essentiellement en Europe et Amérique du Nord. Certaines études indiquent une prévalence de 1% ou plus. Cette différence est justifiée par des différences d'organisation de l'étude ainsi que les critères de sélection et la taille de l'échantillon

2. définie par [Miller, 1998], la robotique d'assistance est tout système robotique pouvant être utilisé afin de compléter une ou plusieurs capacités sensorielles et/ou motrices d'une personne

3. a rapport à l'anthropomorphisme : "L'anthropomorphisme a été défini comme l'attribution de caractéristiques humaines à un non-humain tel que les émotions, la logique, la conscience..etc", [Spatola, 2019]

4. International Conference on Robotics and Automation, l'ICRA est l'une des conférences les plus prestigieuses dans le domaine de la robotique. Organisée par l'IEEE Robotics and Automation Society, elle rassemble chaque année des chercheurs, ingénieurs et professionnels du monde entier pour présenter les dernières avancées en robotique et en automatisation.

5. Le magicien d'Oz en français est une méthode utilisée dans l'IHM et de l'ergonomie informatique, le système contrôlé par un opérateur caché est présenté et pensé autonome par les participants, le "magicien" contrôle le robot "par magie" - [Baccino et al., 2005]

calibre son comportement en fonction des réactions du sujet. Dans le but d'étudier les travaux de recherches en cours sur la thérapie assistée par robot pour le TSA, [Santos et al., 2023] a pu analyser le type de contrôle de 146 études différentes en reprenant la classification de Dautenhahn.

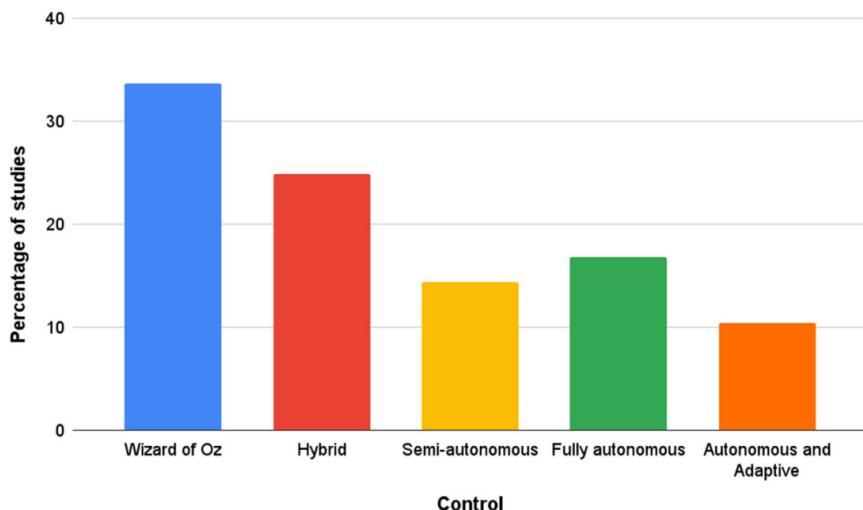


FIGURE 1.1 : Répartition des systèmes de contrôles selon [Santos et al., 2023]

Selon leurs résultats (cf. Figure 1.1), l'utilisation de WOZ est retrouvée dans la majeure partie ($> 30\%$) des études. Ce système permet de tester des interactions et des applications thérapeutiques sans avoir à concevoir d'autonomie directe, permettant ainsi un développement rapide. Il reste néanmoins très contraignant car nécessite la présence constante d'un opérateur, augmentant ainsi la charge des professionnels de santé [Puglisi et al., 2022]. Cette contrainte limite aussi l'efficacité et le potentiel du système en général lors d'un déploiement à grande échelle, rendant ainsi le développement de robots autonomes une alternative plus qu'envisagée. En effet, l'autonomie joue un rôle essentiel pour exprimer l'intelligence d'un robot et dans notre cas, il s'agit de **l'intelligence sociale**⁶. La perception de cette caractéristique apparaît lorsque le robot est capable de manifester des indices sociaux : comportements cohérents, respect des règles sociales et communication naturelle avec les humains [Anzalone et al., 2015] et dans cette optique, le robot doit être suffisamment équipé afin de capter les informations pertinentes de son environnement, et grâce à un système pensé pour, de pouvoir les relier à des entités abstraites de haut niveau.

D'après la littérature, et notamment l'étude approfondie [Puglisi et al., 2022], mettant en revue les principaux robots commerciaux utilisés dans la thérapie du TSA, les robots autonomes les plus souvent retrouvés dans les recherches actuelles en thérapie du TSA sont Nao⁷, Pepper⁸ et Kaspar⁹

— NAO

Selon ces deux études, les auteurs soutiennent que les forces de NAO reposent sur trois points : son autonomie, sa mobilité et sa validité clinique prouvée. Le robot est utilisable sous différents modes : autonomie complète, contrôle hybride (semi-autonomie) et WOZ.

Sa mobilité vient de ses 13 articulations qui lui permettent d'imiter les mouvements humains (cf. Figure 1.2) et sa flexibilité d'utilisation couplée aux différents capteurs dont il est équipé

6. référence à la capacité d'un individu à comprendre, interpréter et répondre de manière appropriée aux interactions sociales (la perception des émotions, l'empathie, la communication efficace et la gestion des relations sociales)

7. Robot humanoïde bipède, développé en 2006 par SoftBank Robotics (anciennement Aldebaran Robotics)

8. Robot humanoïde, développé en 2015 par SoftBank Robotics

9. Développé en 2005 par (Robins et al. 2005) à l'Université d'Hertfordshire, Angleterre

(sonar pour évaluer les distances, capteurs tactiles au niveau des mains et de la tête) font de lui un atout majeur de la médiation comportementale. Il est souvent utilisé dans des thérapies d'interactions sociales.

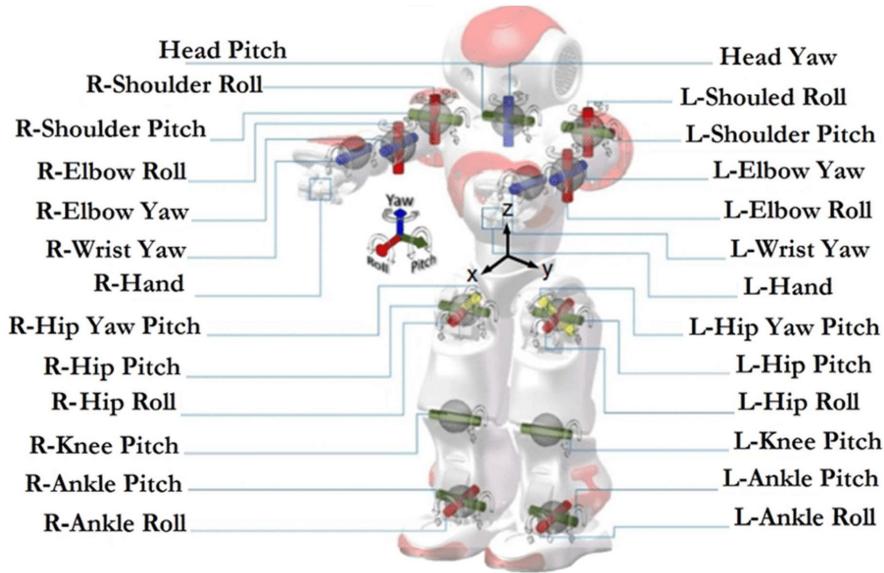


FIGURE 1.2 : Répartition des moteurs à l'intérieur de NAO [Puglisi et al., 2022]

Une étude de [Miskam et al., 2013] a montré des résultats qualifiés d'encourageants dans l'utilisation de NAO dans des exercices de reconnaissance gestuelle et émotionnelle (cf. Figure 1.3).



FIGURE 1.3 : NAO montrant des expressions à un enfant [Miskam et al., 2013]

Dans cette démarche, les auteurs remarquent la faible fréquence d'apparition des traits sociaux typiques des TSA chez les enfants lors de leur utilisation de NAO. Ils notent aussi l'enthousiasme et la réceptivité des enfants aux signaux et aux consignes du robot. Ils ajoutent enfin que tous les enfants ont pu aller à bout du module d'exercices et qu'ils n'avaient pas eu de difficultés particulières à répondre aux questions posées par NAO.

Les auteurs terminent la revue de Nao en soulignant qu'il est reconnu et utilisé dans diverses études cliniques, mais, que certaines limites subsistent : les yeux brillants de NAO sont essentiels pour stimuler l'attention et la concentration du regard mais peuvent s'avérer invasifs et peut exposer l'enfant à une surstimulation. Par ailleurs, NAO ne peut pas non plus montrer d'expressions faciales, et dans ce cas, n'est pas d'une grande utilité dans les thérapies de reconnaissance d'émotions.

— Pepper

Le robot Pepper est un robot autonome, décrit par [Bertacchini et al., 2023] comme étant très efficace dans thérapie du TSA. La tête du robot est dotée de caméras pour la reconnaissance visuelle, de micros pour la reconnaissance vocale et de capteurs tactiles pour les interactions physiques. Le corps intègre un sonar pour évaluer les distances et de joints motorisés lui permettant de bouger et d'adopter des postures (même si elles restent limitées en termes de libertés). Il se déplace à l'aide de roues fixées au bas du corps. (cf. Figure 1.4)

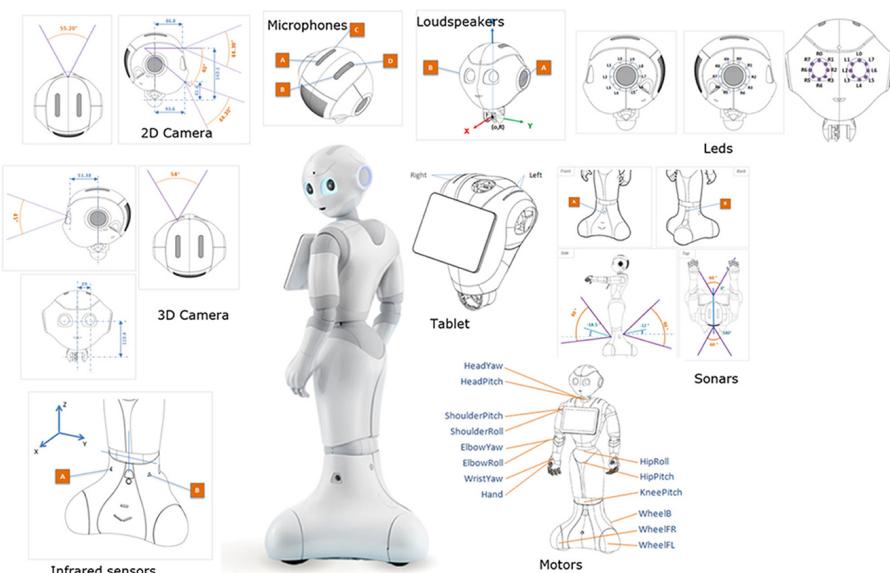


FIGURE 1.4 : Composants physiques du robot Pepper [Bertacchini et al., 2023]

Pepper a notamment été utilisé comme assistant d'éducation. Par exemple, dans une démarche d'intégration du robot Pepper dans un établissement d'éducation spécialisé dans le TSA, [Lemaignan et al., 2022] a étudié l'adoption du robot par les enfants et l'impact de sa présence sur leur bien-être. Le robot est à chaque fois installé à plusieurs endroits de l'établissement (cour de récréation, couloirs...etc) et est libre d'accès pour tous les élèves. Un observateur est également présent afin d'enregistrer le nombre d'interactions effectuées et leur nature (cf. Figure 1.5)

Ils constatent que les enfants passaient environ 20% du temps de leur journée d'école avec le robot, et ont remarqué plusieurs types de comportements différents et de relation avec le robot d'un enfant à un autre. Certains le voyaient comme un confident, lui racontant les événements d'une journée ou un exercice difficile effectué en classe. D'autres enfants voient plutôt en lui un compagnon de jeu, et ne viennent le voir que pour se divertir.

Pepper témoigne de nombreux succès, cependant, son prix (vendu 20.000 euros aux entreprises) limite son accès pour certaines familles et institutions.

— Kaspar



FIGURE 1.5 : Illustration d'une interaction entre le robot et un élève de l'école [Lemaignan et al., 2022]

Le robot Kaspar est un robot humanoïde. La tête est équipée de 8 joints moteurs lui permettant de bouger le cou et les yeux, de capteurs de force et de capteurs tactiles pour les interactions au toucher. Son corps est équipé de six joints moteurs pour les mouvements des bras et de capteurs tactiles (cf. Figure 1.6). La programmation du robot se fait via une interface simple d'utilisation mais très limitée car incompatible avec les autres plateformes [Puglisi et al., 2022].

Parmi les cas d'utilisation de Kaspar dans la thérapie du TSA, une étude de [Wainer et al., 2013] où le robot est déployé pour une utilisation autonome afin d'encourager et de motiver des enfants à effectuer des exercices d'imitation a montré des résultats concluants : une amélioration des compétences sociales et une meilleure compréhension des règles de la communication chez les enfants, et ce, en dehors de l'utilisation du robot.

Selon [Puglisi et al., 2022], les avantages de Kaspar se résument en trois points : sa simplicité de programmation, les multitudes de capteurs dont il dispose et son expression faciale simple. La simplicité de ce système réduit la complexité de l'échange et des informations cognitives perçues par l'enfant, rendant le robot plus prévisible et moins ambigu. Cependant, dans cette même simplicité résident les limites de Kaspar : la mobilité étant un facteur déterminant, Kaspar ne peut pas marcher, saisir d'objets ou effectuer des gestes fins avec ses mains réduisant ainsi les champs du possible en terme de scénarios d'usage et d'actions pouvant être effectuées.

D'autres modèles de systèmes autonomes existent, l'étude de [Scassellati et al., 2018] porte sur une intervention à domicile d'un robot social autonome visant à améliorer les compétences de communication sociale d'enfants atteints de TSA sur une durée d'1 mois. Durant cette période, des activités ayant pour but de développer des compétences sociales différentes (compréhension émotionnelle et sociale, prise de décisions, ordonnancement et séquençage) ont été réalisées, et entre chaque session, l'enfant est invité à échanger avec son accompagnateur. L'étude de ce système se différencie des autres travaux dans ce domaine sur quatre aspects importants et déterminants :

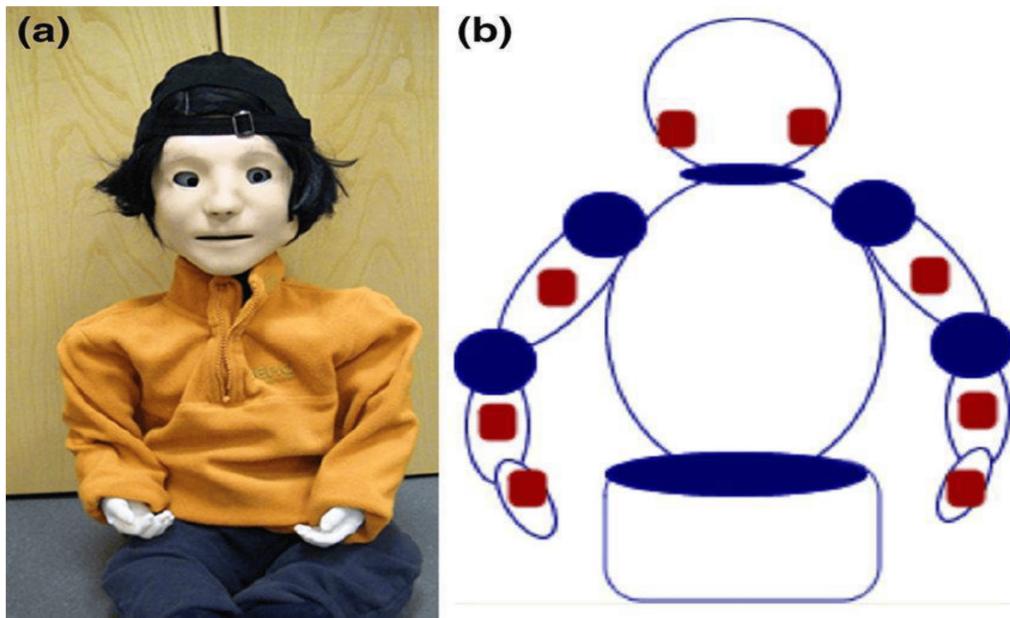


FIGURE 1.6 : Le robot Kaspar (a) et la localisation des capteurs (b) [Puglisi et al., 2022]

- le robot opère en toute autonomie sans requérir d'ajustements techniques extérieurs.
- les conditions environnementales ici diffèrent des conditions contrôlées retrouvées en laboratoire.
- le système offre une diversité des scénarios en s'adaptant aux compétences de l'enfant via une difficulté changeante.
- l'étude mesure en particulier la capacité de l'enfant à faire appel aux compétences acquises dans un contexte social plus général.

Les résultats récoltés ont montré une prise en main rapide du système de la part de l'enfant avec une amélioration de l'attention conjointe¹⁰ avec les adultes en dehors de la présence du robot. L'étude a également soulevé un défi "considérable" : le passage du modèle téléopéré habituel à un modèle autonome opérant dans un environnement incontrôlé.

En dehors du cas particulier d'un TSA, LOLA-2 (cf. Figure 1.7), un robot intelligent propulsé par l'IA, est développé par [López-Sastre et al., 2021] à destination d'enfants avec des TND (troubles moteurs plus particulièrement). Ce système analyse leurs comportements et propose un accompagnement aux côtés des professionnels de santé.

Ce robot thérapeutique est présenté novateur car connecté à un système **OAD**¹¹ (**Online Action Detection**) en ligne permettant la détection du début ou de la fin de l'action au moment même où elle est engagée ou achevée. Selon les chercheurs, cet ajout est motivé par une volonté d'anticiper les actions afin d'avoir une meilleure gestion du flux de données en temps réel. Toujours dans l'étude du même robot, des travaux de validation clinique, [Nasri et al., 2022], ont démontré

10. L'attention conjointe est définie comme la capacité de l'enfant à partager un évènement avec autrui, à attirer et à maintenir son attention vers un objet ou une personne dans le but d'obtenir une observation commune et conjointe (<https://www.inshea.fr/sites/default/files/fichier-orna/Descriptif%20pédagogique.pdf>)

11. l'OAD implique l'identification d'une action humaine en **temps réel** à partir d'une séquence vidéo pour détecter une action dès qu'elle commence. Différentes méthodes classiques qui analysent la vidéo entière pour détecter les actions



FIGURE 1.7 : Robot LOLA2 vu de face [López-Sastre et al., 2021]

l'efficacité du système lors d'une utilisation en contexte réel où les enfants sélectionnent la tâche journalière à effectuer et regardent ensuite une vidéo explicative qui découpe les mouvements un à un. Ils sont ensuite invités à effectuer les tâches devant le robot qui analyse les mouvements et dresse un rapport avec : l'action demandée, l'action détectée par le système, la durée de réalisation de l'action, et une image de l'enfant pendant l'exécution de l'action. Les résultats ont démontré une bonne adaptation des enfants à l'outil et une compréhension facile des instructions. Les auteurs précisent que les données récoltées pourraient, par exemple, être utilisées par les professionnels de santé pour une surveillance de thérapie à domicile.

En somme, les études citées précédemment sur l'utilisation des SAR dans la thérapie des TND montrent que ces systèmes ont un potentiel non négligeable pour **faciliter les interactions sociales** en proposant un environnement plus **prévisible et rassurant** pour l'enfant. [Anzalone et al., 2015] et [Puglisi et al., 2022] soulignent aussi l'importance de **l'apparence, la mobilité et de l'intelligence sociale** que le robot est capable d'exprimer. Ces éléments, lorsqu'ils sont réunis permettent de susciter **l'attention conjointe**, d'encourager **les comportements d'imitation** et d'améliorer **la reconnaissance des émotions**. Aussi, le mode de contrôle joue un rôle important et décisif et doit être choisi au cas par cas. Les conditions cliniques peuvent varier d'un patient à un autre et

d'un environnement à un autre. Les robots nécessitant l'intervention d'un opérateur sont certes plus rapides à mettre en place mais ne permettent pas de généraliser la thérapie et constraint donc son utilisation à plus petite échelle et à des cadres différents. D'autre part, la communauté scientifique encourage le développement de robots plus autonomes capables de mieux s'intégrer aux environnements quotidiens, cependant, ces systèmes doivent surmonter des obstacles liés à **la détection précise** et à la gestion des interactions des **scénarios et contextes variés**.

1.2 Objectifs

La robotique sociale et ses nombreux avantages sont cependant limités. Les systèmes actuels déplorent un manque de moyens humains et financiers afin d'être déployés et testés à grande échelle ; la fabrication d'un robot est coûteuse et son utilisation nécessite la présence d'un professionnel qualifié.

Mon travail consiste à mettre en pratique mes connaissances en programmation et en robotique afin de concevoir et réaliser un robot compagnon programmable, communicatif et engageant. Ce robot se différencie des autres (Nao, QTRobot...) par sa personnalisation simple à travers un langage naturel qui sera ensuite interprété via l'interface d'IA mise en place, mais aussi par son faible coût matériel : un kit amateur de robotique, un écran, une carte Raspberry Pi 4, un haut-parleur ainsi qu'un micro. Tous ces éléments sont accessibles au grand public, des composants physiques aux technologies logicielles.

Afin d'être en adéquation avec les principes actuels de robotique sociale et particulièrement pour les personnes atteintes de TND, le robot doit atteindre un antropomorphisme suffisant pour stimuler au mieux l'enfant. Pour ce faire, tout est mis en oeuvre pour établir et mettre au point ces propriétés : postures, expressions faciales, traits émotifs, voix et intelligence artificielle. Autant d'aspects importants à soigner car ils permettent d'améliorer la concentration et l'apprentissage des enfants en produisant des signaux verbaux et non verbaux simples à comprendre.

La plateforme sera testée en collaboration avec les équipes soignantes du service de Psychiatrie de l'Enfant et de l'Adolescent de l'Hôpital de la Pitié-Salpêtrière. Les résultats serviront à réviser le robot et à l'orienter afin de respecter au mieux ses objectifs thérapeutiques pour à terme en faire un outil programmable en autonomie et utilisable dans la prise en charge des troubles du neuro-développement.

Chapitre 2

Méthode

2.1 Hardware

2.1.1 Conception générale

La conception de la structure du robot se fait entièrement avec des pièces fournies dans le kit Ultimate 2.0 (cf. Figure 2.1-2.2) commercialisé par Makeblock.



FIGURE 2.1 : Emballage du kit



FIGURE 2.2 : Contenu du kit (1)

Le kit dispose de tous les éléments nécessaires à des projets robotiques débutants ou amateurs et fournit également un manuel pratique détaillant l'utilité de chaque pièce, et un guide de programmation et d'introduction à leur application de programmation par bloc.

Cela a été d'une grande aide, notamment en guise d'introduction à la robotique mais aussi à réalisation d'un premier croquis du robot (cf. Figure 2.4).

À cette étape, il apparaît que la conception du robot dépend grandement de deux parties ; le " joint " qui contient les deux axes principaux du bas (x , y) et le système de contrôle. En ce qui concerne le joint, le défi à relever est de rapprocher au maximum ces axes en un seul point afin de simuler le fonctionnement des hanches chez l'humain (cf. Figure 2.5) et de satisfaire mon objectif d'avoir un robot anthromorphe.

Ce joint peut également être repris pour en faire un " cou " attaché à l'extrémité haute, offrant encore plus de degrés de liberté.

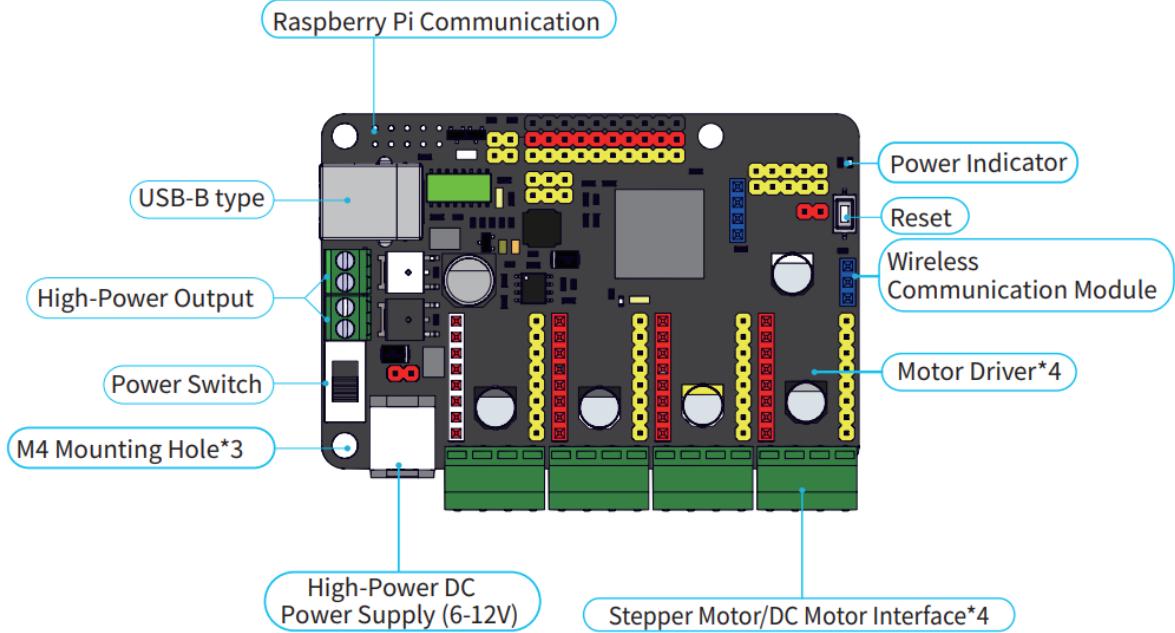


FIGURE 2.3 : Schémas de la MegaPi incluse dans le kit

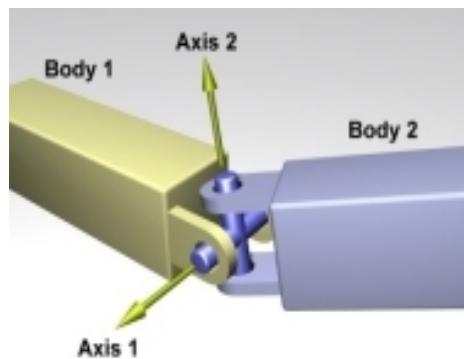
Le robot doit être programmable de façon simple et autonome, et cette partie représente aussi l'interface de communication entre l'enfant et le robot. Pour ces motifs, il est donc impératif d'avoir un écran et de mettre en place un environnement facilitant l'accès à la boucle de contrôle à toute personne compétente désirant apporter des changements.

2.1.2 Joints et moteurs

Le système est composé d'une base rotative, de deux joints (hanches et cou) et d'un axe supplémentaire au centre. Ceci porte le nombre total de moteurs à six.

Avant d'entamer l'assemblage du joint, j'effectue d'abord des recherches sur la façon d'avoir deux axes superposés en puisant mes sources notamment dans certains travaux en IoT (internet des objets) ainsi qu'en mécanique.

Russel Smith lors de sa présentation à Palo Alto en 2004^a traite la simulation dynamique des corps rigides. Il y a mentionné l'importance des joints lors de la conception de charnières ou de pivots lors d'une simulation. Mon travail ne concerne pas directement la simulation 3D, cependant, les schémas et les différents exemples de joints qu'il a présentés m'ont inspiré et m'ont permis d'avoir une connaissance de base sur le sujet. Le joint représenté dans la figure de droite a notamment attiré mon attention car assez simple à réaliser avec le matériel disponible.



a. <https://ode.org/slides/parc/dynamics.pdf>

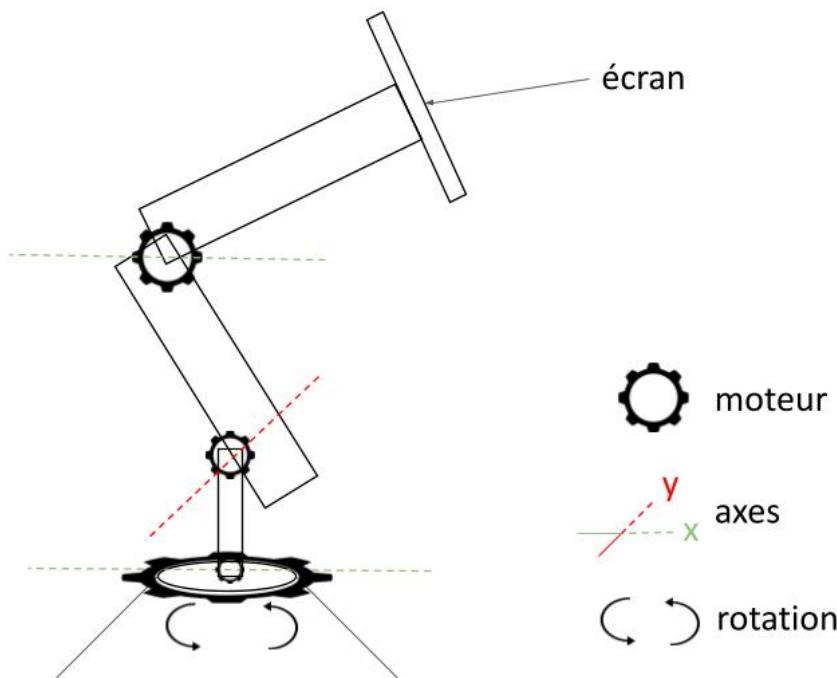


FIGURE 2.4 : Aperçu général du robot

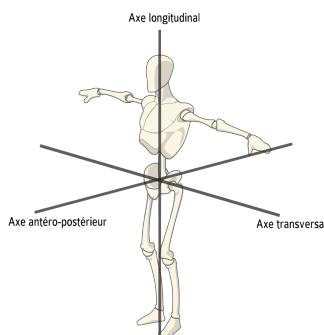


FIGURE 2.5 : Axes anatomiques de référence chez l'humain

Le joint à deux axes doit ensuite être monté sur une plateforme tournante et le tout doit former un bloc homogène, j'oriente alors mon attention sur des travaux utilisant ce type de mécanismes. Un tutoriel de conception en ligne, intitulé "Système de Suivi Solaire à Deux Axes" (cf. Figure 2.6), m'a fortement aiguillé, particulièrement pour ce qui est de la disposition des moteurs d'axes et de la la base tournante.

Ce type de mouvement et de configuration est typiquement ce que je recherche pour apporter à mon robot le comportement désiré. Tout comme le capteur solaire, la majorité des travaux dans ce domaine font appel à de l'impression 3D (cf. Figure 2.7) car cela permet de personnaliser le montage et de répondre au mieux à l'objectif voulu. Pour atteindre le même résultat avec le matériel fourni dans le kit, je fais d'abord des essais pour déterminer ce qui est possible avec.



FIGURE 2.6 : Capteur solaire, [Parajuli, 2024]

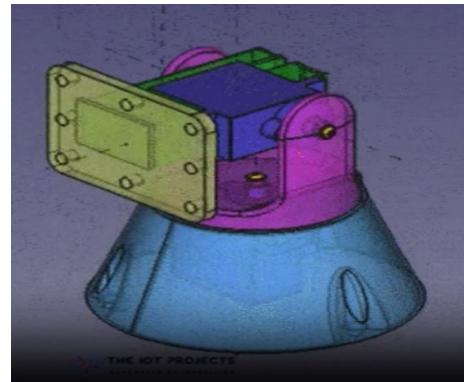
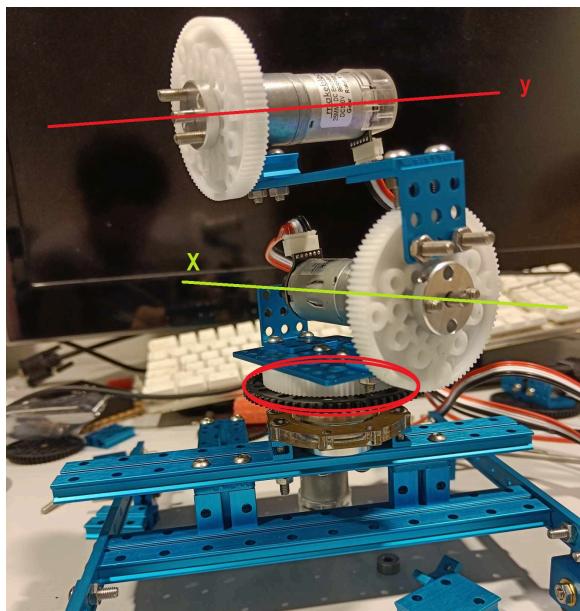
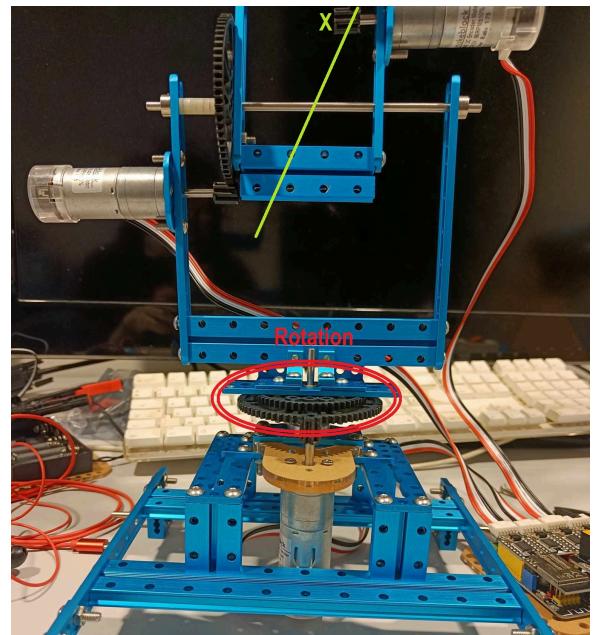


FIGURE 2.7 : Capteur solaire en schémas, [Parajuli, 2024]

La première configuration à droite inclut une rotation et un axe x mais je réalise très vite que le point d'équilibre se trouve en un seul point au centre du disque tournant ce qui affecte la stabilité de la structure. L'axe x quant à lui est encore loin du disque ce qui en fait plus une articulation seule qu'un axe du joint.

Il manque aussi le second axe y, mais pour le moment, cette configuration semble la plus simple à réaliser en premier.

Trois améliorations sont donc à apporter : une proximité cohérente entre le disque tournant et l'axe, une meilleure stabilité au point d'équilibre et l'ajout de l'axe y.



La deuxième configuration à gauche se rapproche un peu plus du résultat voulu. En renforçant la base à l'aide d'un roulement à billes et d'une plaque plus large, l'effort est moins concentré donc plus de stabilité. Ensuite, en m'inspirant du schémas 3D du capteur solaire (fig.10), je change la fixation des moteurs et les visse directement sur le corps que je veux faire bouger. Les deux axes x et y sont présents mais ils restent encore assez éloignés l'un de l'autre. Un meilleur agencement est donc nécessaire.

Toujours à partir de 2.62.7 et m'inspirant de la forme en **U** des arcs reliés aux deux côtés de chaque moteur, j'ai pu modifier le joint pour avoir la forme finale que j'utiliserais pour la suite du projet (cf. Figure 2.8). Pour faute de robustesse de la structure, je ne pourrai pas installer le joint sur la tête, cela ferait une trop grande charge à supporter pour le robot.

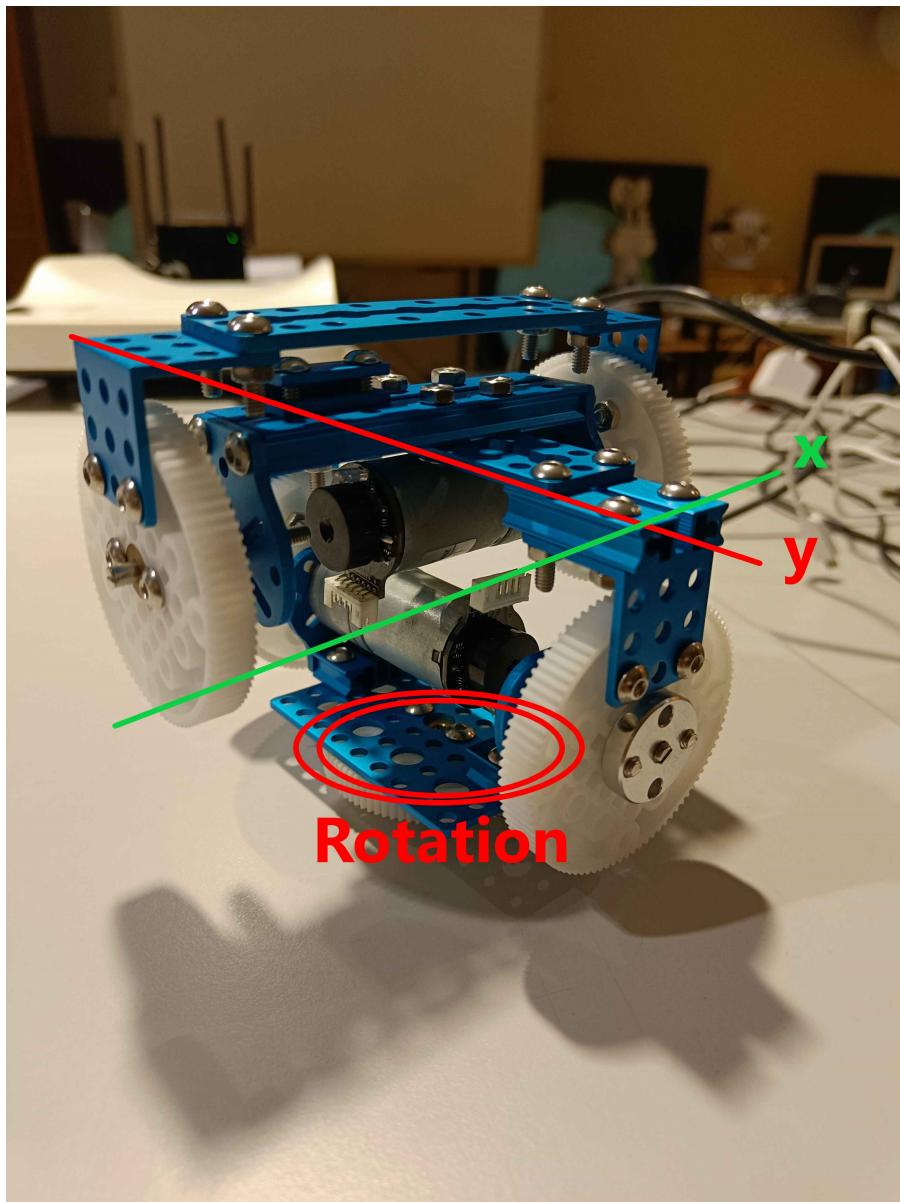


FIGURE 2.8 : Prototype final du joint

2.2 Contrôle de bas niveau

Afin de programmer le système et de permettre à la plateforme d'être reprise dans le futur, est nécessaire de mettre en place une interface de contrôle qui puisse être ensuite accessible et facilement ajustable.

La conception de cette interface commence par la communication entre la MegaPi et les moteurs. Makeblock offre une plateforme de développement par blocs nommée Mblock¹ plus ou moins complète et idéale pour tout projet débutant. L'interface est simple à utiliser et la prise en main assez rapide. La figure 2.9 représente un test que j'ai effectué afin de vérifier le fonctionnement des moteurs.

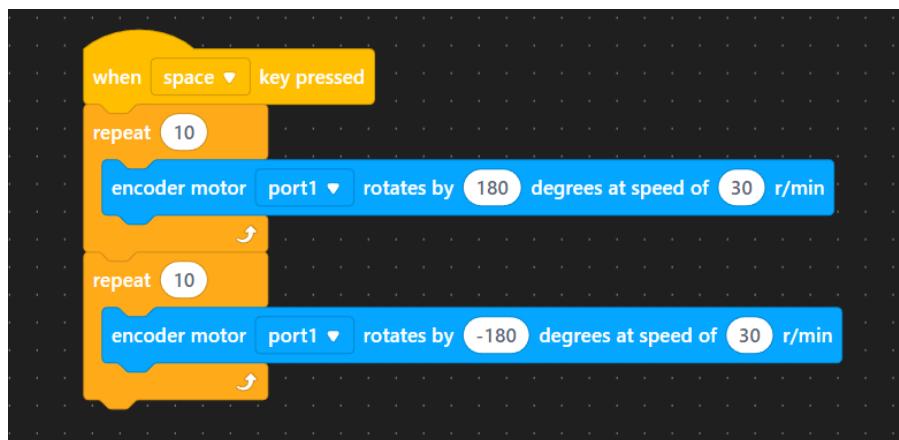


FIGURE 2.9 : Capture d'écran d'un bloc d'instructions sur Mblock

Une fois le code transféré sur la MegaPi, un clic sur la touche "espace" du clavier entraîne une succession de dix rotations à 180° à une vitesse de 30 tours/min. Ensuite une succession de dix rotations dans le sens inverse et à la même vitesse.

Malgré sa facilité d'utilisation, cette plateforme n'offre pas l'environnement nécessaire à la réalisation de mon projet : je dois pouvoir accéder à la boucle de contrôle via ROS depuis ma Raspberry. Afin de permettre des utilisations plus avancées, Makeblock fournit également, via des plateformes de partage en ligne², des bibliothèques et librairies en C++ pour tous les composants électroniques et microprocesseurs présents dans leurs kits. Pour la MegaPi utilisée, il s'agit de MakeBlockDrive.

Cette librairie contient les bibliothèques de contrôle et de surveillance des moteurs. Parmis les fonctions incluses, seules les suivantes m'intéressent :

1 boolean move(float angle, float speed);

et

1 float getCurrentPosition(boolean move(float angle, float speed));

La première fonction `move` sert à déclencher un moteur en indiquant un angle relatif³ et une vitesse, la seconde fonction `getCurrentPosition` sert à récupérer la valeur l'angle actuel du

1. <https://mblock.makeblock.com/en/>,
2. <https://github.com/Makeblock-official/Makeblock-Libraries/tree/master>

3. le moteur se déplacera en fonction de l'angle spécifié par rapport à sa position actuelle (Ex : si le moteur a tourné à 30°, l'utilisation de la commande le fera tourner d'encore 30°) contrairement à l'utilisation d'un angle absolu où les positions des angles sont fixes par rapport au référentiel (généralement 0°)

moteur. Dans la suite du projet, seules ces deux fonctions sont utilisées pour la communication Arduino/moteurs.

Ensuite, dans le cadre de mon stage, j'ai recouru à **ROS (Robot Operating System)** pour le système de contrôle du robot. ROS est un ensemble d'outils et de bibliothèques permettant de développer des logiciels pour la robotique. Tout comme un système d'exploitation pour ordinateur, une fois installé sur la Raspberry, ROS fournit une infrastructure complète afin de coordiner la communication entre cette dernière et la MegaPi.

La version de ROS utilisée dans le système est ROS 1.17 (Noetic), dernière version de ROS 1 avec support à long terme. Grâce à son support pour Python 3, et les différentes bibliothèques qu'elle offre, cette version est largement utilisée en robotique en particulier pour ces raisons :

- permet de programmer en C++, Python ainsi que d'autres langages, ce qui en fait un outil flexible selon les exigences du projet.
- offre des outils de visualisation en temps réel, permettant de surveiller les comportements du système en analysant les messages envoyés et reçus, et de déboguer si nécessaire.
- intègre des simulateurs 3D permettant de tester des architectures robotiques et des algorithmes avant de les déployer sur des robots réels.
- divise le système en plusieurs sous-composants appelés "nœuds"⁴. Grâce à cette modularité, la gestion du système et la surveillance des tâches en cours sont facilitées.
- assure la communication entre nœuds via des services appelés "publisher" et "subscriber" qui permettent respectivement de publier (envoyer) et d'écouter (recevoir) un message (tout type de données). Cette communication organisée permet de construire des systèmes robustes et capables de s'adapter à divers environnement matériels et logiciels.
- dispose d'une large communauté et ressources open-source facilitant grandement la documentation.

Enfin, il est nécessaire de se concentrer sur la communication ROS/Arduino qui sert à envoyer les données nécessaires aux fonctions précédentes (`move` et `getCurrentPosition`). Plusieurs bibliothèques et protocoles permettent cette communication et leur utilisation dépend de la nature du projet souhaité. Le tableau 2.1 représente les différents protocoles possibles pour y arriver.

La communication en série est ce que je recherche car mes deux cartes communiquent entre elles via USB, donc `rosserial` semble être la solution. Même si `ros2arduino` permet le même type de connexion, ce n'est compatible qu'avec la version 2.+ de ROS. Mon choix pour ROS 1 au lieu de ROS 2 a été motivé par l'impossibilité d'utiliser certains packages sur la Raspberry. Ce problème est lié à la différence d'architecture des processeurs (ARM et AMD) et fait que certains packages compilés à destination de l'une ne fonctionne pas sur l'autre. L'installation de `rosserial` se fait via le téléchargement des packages nécessaires sur la Raspberry, qui nécessite ensuite l'utilisation de la bibliothèque `RosserialArduinoLibrary` sur la MegaPi.

Les figures 2.10-2.11 représentent le fonctionnement des noeuds, plus particulièrement le contrôle et la lecture des données sur les moteurs. Premièrement, le contrôle.

4. processus autonome chargé d'une tâche : lecture de capteurs, le contrôle des moteurs ou le traitement des données. Les noeuds communiquent entre eux via des topics (communication asynchrone) ou des services (communication synchrone)(<https://wiki.ros.org/ROS/Tutorials/UnderstandingNodes>)

Outil	Description	Langages Supportés	Protocoles Supportés
rosserial	Permet la communication série entre ROS et un microcontrôleur.	C++ et Python (ROS), C/C++ (microcontrôleur)	Communication série (UART, USB)
roscontrol	Suite de packages pour la gestion des contrôleurs dans ROS.	C++	-
ros2arduino	Facilite la communication entre ROS 2 et Arduino.	C++ (ROS 2), C++/Arduino (microcontrôleur)	Communication série (UART)
rosbridgeSuite	Fournit une interface entre ROS et des applications externes via WebSockets.	Fonctionne avec n'importe quel langage utilisant WebSockets	WebSocket, JSON

TABLE 2.1 : Tableau comparatifs des protocoles de communication ROS/Arduino

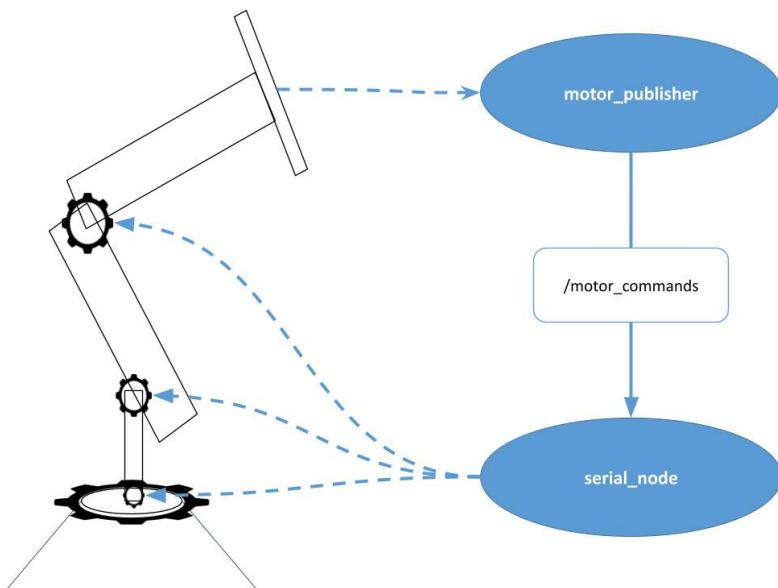


FIGURE 2.10 : Schémas représentant l'actionnement des moteurs via rosserial

Pour communiquer avec la MegaPi et envoyer les commandes requises au contrôle des moteurs, il faut lancer sur la Raspberry un premier "publisher" `motor_publisher` (cf. Figure 2.10) qui sert à envoyer les données (dans notre cas il s'agit d'un tableau : `[angle_moteur1,vitesse_moteur1,angle_moteur2,vitesse_moteur2,angle_moteur3,vitesse_moteur3,angle_moteur4,vitesse_moteur4]` à travers le topic `/motors_commands`. Ces données sont récupérées par la carte Arduino et sont ensuite extraites de la chaîne de caractère de cette manière :

```

1 commands.data_length = inputMessage.data_length;
2 commands.data = (int32_t*)realloc(commands.data, commands.data_length
3 * sizeof(int32_t));
4
5 for (uint16_t i = 0; i < commands.data_length; i++) {
6     commands.data[i] = inputMessage.data[i];
7 }

```

Dans ce code, `inputMessage` est la donnée récupérée via `rosserial`. La matrice est parcourue via une boucle et les données sont extraites et stockées dans un tableau local `commands`. Une fois le tableau rempli, je fais ceci :

```

1 Encoder_1.move(commands.data[0], commands.data[1]);
2 Encoder_2.move(commands.data[2], commands.data[3]);
3 Encoder_3.move(commands.data[4], commands.data[5]);
4 Encoder_4.move(commands.data[6], commands.data[7]);

```

Dans ce code, Encoder_1/2/3/4 représentent nos quatre moteurs et grâce à la commande move(angle, vitesse) et les données récupérées, il est possible de les faire bouger.

Deuxièmement, la réception des données sur la Raspberry. Pour ce faire, un "subscriber" motor_subscriber est lancé et se met à l'écoute des messages reçus via le topic motor_positions. Sur la carte Arduino, rosserial récupère les angles des moteurs grâce à la fonction getCurrentPosition puis les stocke dans un tableau positions et les publie ensuite via le topic /motors (cf. Figure 2.11)

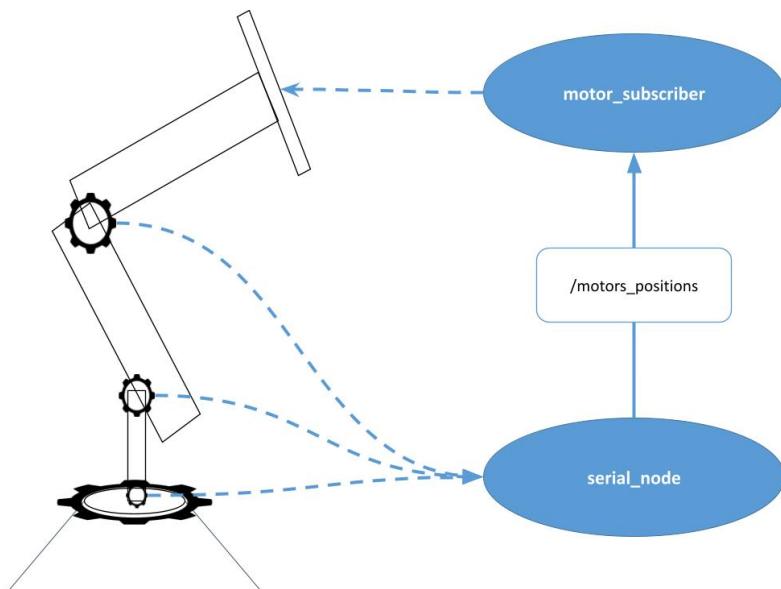


FIGURE 2.11 : Schémas représentant la lecture des données (angles) via rosserial

2.3 Visages expressifs

Afin de d'intégrer au mieux avec l'enfant, le robot doit offrir une communication crédible. L'utilisation d'un visage fournit un moyen visuel pour renforcer cette interaction car il permet au robot de montrer une large gamme d'expressions faciales. Ces expressions aident à transmettre des intentions à l'enfant qui pourrait avoir des difficultés avec les signaux non verbaux par exemple. De plus, cela permet aussi une adaptation des réactions en temps réel en modifiant les expressions en fonction des réactions de l'enfant et même d'afficher des récompenses visuelles pouvant l'encourager à plus d'engagement, nécessaire à la création d'interactions plus riches.

Plusieurs approches peuvent être envisagées pour afficher un tel visage sur l'écran. La première étant la modélisation 3D en temps réel. Rosserial et son système de nodes est aussi utilisable pour l'envoi de scripts personnalisés. Des applications telles que Blender, Unreal Engine ou Maya disposent d'API sous différents langages de programmation permettant d'animer des objets 3D par le biais de lignes de codes.

— Blender : Python.

- Unreal Engine : C++ et Blueprints principalement, Python partiellement.
- Maya : Python et MEL.

Cependant, le coût en ressources logicielles pour ce type de rendu dépassant grandement ce que la Raspberry peut fournir.

La seconde approche est l'utilisation de vidéos d'expressions et de comportement préenregistrées qui seront ensuite lancées grâce aux nœuds et aux bibliothèques media de ROS. Un logiciel d'animation est donc nécessaire.

2.3.1 iClone et personnages

iClone permet de créer et d'animer des modèles 3D avec une grande précision au niveau des mouvements du visage et du corps. Particulièrement grâce à son plugin Face Key qui simplifie l'animation du visage en le décomposant en plusieurs parties individuellement amovibles (cf.Figure 2.12) rendant ainsi l'expression d'émotions diverses plus facile.



FIGURE 2.12 : Capture d'écran du plugin Face Key sur iClone 8



FIGURE 2.13 : Capture d'écran d'un modèle 3D créé sur iClone 8

Le logiciel dispose de modèles 3D par défauts (corps, tête, cheveux, vêtements, accessoires, textures de peau...). La Figure 2.13, iClone facilite également l'import de modèle 3D déjà existants avec un support pour les formats OBJ⁵ et FBX⁶. Cette opportunité permet le stockage d'une multitude d'apparences pour le visage à afficher sur l'écran. En effet, il serait possible de créer différents profils avec des voix, des visages et des personnalités différentes.

Dans ce cas d'utilisation, voici une liste des personnages disponibles sur Blender (sous formats OBJ et FBX).

(Source : <https://studio.blender.org/characters/>)

5. OBJ : Ce format de fichier prend en charge l'encodage approximatif et précis de la géométrie de la surface. Peut coder des informations de couleur et de texture. Il ne supporte aucun type d'animation (<https://3dcrealab.fr/2019-most-common-3d-file-formats/>)

6. FBX : Ce format de fichier prend en charge les propriétés liées à la géométrie et à l'apparence, telles que la couleur et les textures. Il prend également en charge les animations squelettiques et les morphes (<https://3dcrealab.fr/2019-most-common-3d-file-formats/>)



FIGURE 2.14 : Sprite FIGURE 2.15 : Dog FIGURE 2.16 : Cat FIGURE 2.17 : Ellie

2.3.2 Expressions faciales

Avant d'entamer la création des vidéos, il est nécessaire d'avoir une base de connaissances sur la représentation des expressions faciale chez l'humain. Dans les années 70, Paul Ekman et Wallace V.Friesen développent le Facial Action Coding System (FACS). Ce système classe les expressions du visage et leurs mouvements en unités d'action spécifiques, ce qui en fait l'un des systèmes de codage les plus utilisés dans les domaines de la psychologie, de l'animation et de l'intelligence artificielle.

Selon cette classification, chaque expression engendre le mouvement d'une ou plusieurs parties précises du visage (cf. Figure 2.18).

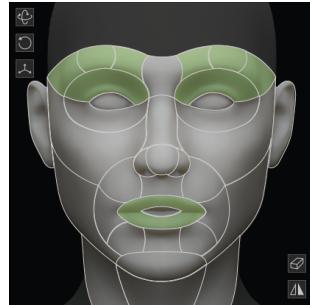
Unités d'action UA - Partie supérieure du visage					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
sourcil int. levé	sourcil ext. levé	sourcil abaissé	paupière sup. levée	joue levée	paupière tendue
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
paupière affaissée	yeux fendus	yeux fermés	plissement	clignement	clin d'oeil
Unités d'action UA - Partie inférieure du visage					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
nez plissé	lèvres sup. levée	sillon nasolabial	lèvres étirées	joues gonflées	lèvres avec fossettes
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
coins lèvres abaissés	lèvre inf. abaissée	menton levé	lèvres pinçées	bouche étirée	lèvres entonnoir
AU 23	AU 24	AU 25	AU 26	AU 27	AU 28
lèvres tendues	lèvres pressées	lèvres séparées	mâchoire abaissée	bouche étirée	lèvres aspirées

FIGURE 2.18 : Codage FACS de certaines actions du visage d'après Paul Ekman

En lien avec le FACS et en exploitant Face Key, il est possible de modifier des zones précises du visage afin de reproduire les six émotions de bases identifiées par Paul Ekman :

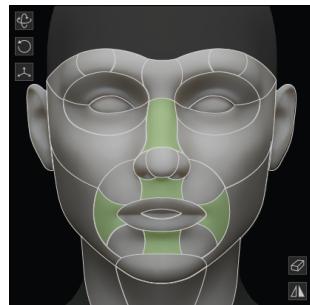
1. Colère

- **AU4** : Abaissement des sourcils
- **AU5** : Ouverture des paupières sup.
- **AU7** : Tension des paupières
- **AU23** : Tension des lèvres
- **AU24** : Pression des lèvres



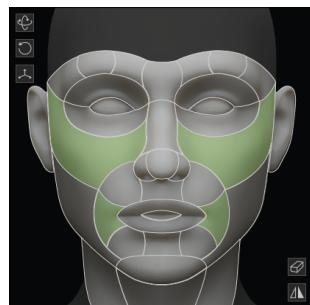
6. Dégoût

- **AU9** : Plissement du nez
- **AU10** : Soulèvement de la lèvre sup.
- **AU15** : Abaissement des coins de la bouche
- **AU16** : Abaissement de la lèvre inf.



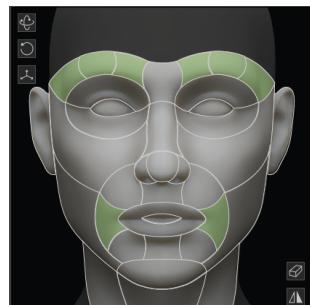
5. Joie

- **AU6** : Élévation des joues
- **AU12** : Élévation des coins de la bouche



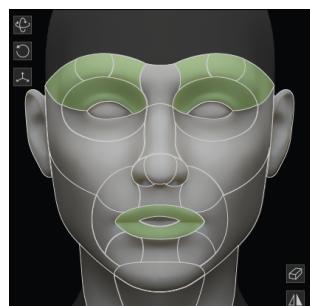
3. Tristesse

- **AU1** : Élévation des sourcils int.
- **AU15** : Abaissement des coins de la bouche



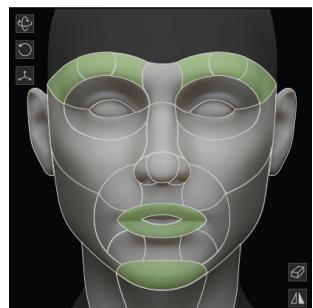
3. Peur

- **AU1** : Élévation des sourcils int.
- **AU2** : Élévation des sourcils ext.
- **AU5** : Élévation des paupières sup.
- **AU20** : Etirement de la bouche



5. Surprise

- **AU1** : Élévation des sourcils internes
- **AU2** : Élévation des sourcils ext.
- **AU5** : Élévation des paupières sup.
- **AU26 - AU27** : Abaissement de la mâchoire - Etirement de la bouche



La mise en place de cette relation entre Face Key et le FACS a permis la création de six vidéos pour chacune des émotions : colère, dégoût, joie, tristesse, peur et surprise. En complément, deux vidéos supplémentaires sont enregistrées : neutre et parole (lorsque le robot parle).

2.3.3 Contrôle de l'image

Pour que l'échange avec le robot soit fluide, il est nécessaire d'avoir un retour visuel en temps réel. Les expressions affichées sur l'écran doivent naturellement suivre le rythme de la conversation.

L'expression par défaut du visage est le neutre. Ainsi, en utilisant la commande en ligne rostopic pub sur la console de la Raspberry, il est possible de publier une chaîne de caractères au "subscriber" video_subscriber via le topic video_commands. Cette chaîne de caractère représente le chemin de la vidéo sur le disque. Une fois reçue, la vidéo est lancée sur l'écran. L'objectif est de lancer une vidéo en réaction d'un discours ou bien pour l'accompagner.

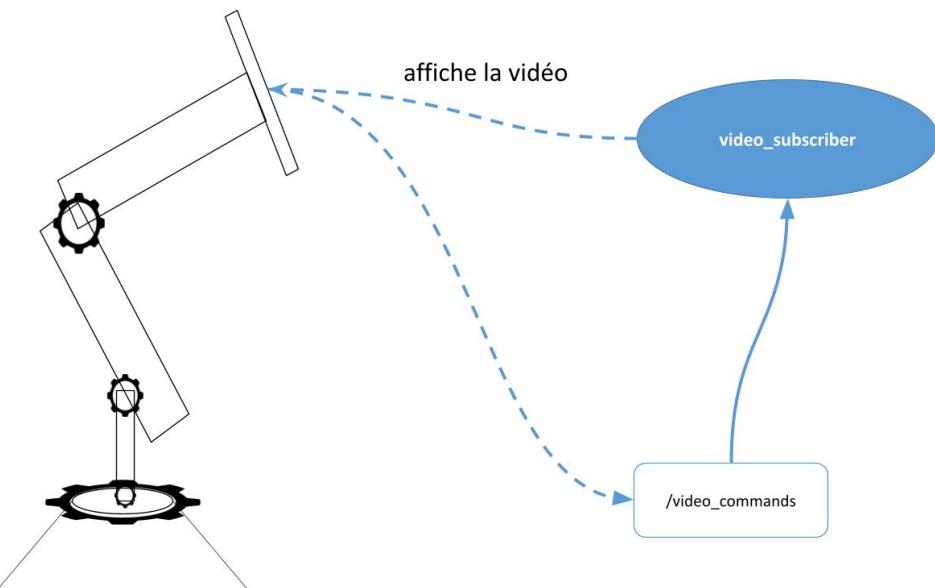


FIGURE 2.19 : Schémas représentant le mode d'affichage de la vidéo sur l'écran

Par exemple, la commande ci-dessus permet de publier le chemin sur le disque de la vidéo sad.avi via le topic /video_commands :

```
1 rostopic pub -1 /video_commands 'data=[chemin_de_la_vidéo]'
```

Dans cette exemple, `-1` est une option de la commande qui permet d'envoyer le message qu'une seule fois seulement (le comportement par défaut est d'envoyer indéfiniment le même message pendant 5 secondes). Cette chaîne de caractère est traitée par `video_subscriber` puis lue et affichée en plein écran sur le moniteur de la Raspberry.

2.4 Discours

2.4.1 Reconnaissance vocale

La reconnaissance vocale est l'un des éléments clés dans l'accessibilité de l'échange entre l'humain et la machine. Des bibliothèques et des packages de reconnaissance vocale compatibles avec ROS sont développés par la communauté, parmi ces derniers, `Pocketsphinx`.

Conçu pour fonctionner dans des systèmes à ressources limitées et hors réseau internet, c'est une version plus légère de `Sphinx` utilisée dans des projets d'IoT et de systèmes embarqués.

Son implémentation dans le système est faite via le package `pocketsphinx_ros` en l'intégrant dans un subscriber au topic `/voice_commands`. Une configuration des fichiers de modèle et de dictionnaires est d'abord requise, en spécifiant dans l'en-tête les chemins pour :

- le modèle acoustique : fichier contenant les informations sur les phonèmes de la langue française.
- le modèle de langage : fichier contenant les informations de transition entre les mots.
- le dictionnaire phonétique : traduction phonétique de chaque mot.

L'extrait audio est alors transmis au subscriber `voice_subscriber` via le topic `voice_commands` (et les données vocales sont traduites en chaîne de caractères et peuvent ensuite être utilisées par le système pour toute tâche le nécessitant).

2.4.2 Synthèse vocale

Tout comme la reconnaissance vocale, créer un discours constitue un atout majeur dans la communication humain-machine. La synthèse vocale, pour mon projet, est possible via trois principaux packages représentés dans ce tableau :

Outil	Installation	Langues Supportées	Coût et Connexion
Festival ROS	Installation locale ROS + Festival	Plusieurs langues, mais limitées	Gratuit, fonctionne hors ligne
text_to_speech	Installation via ROS + API Microsoft Azure TTS	Large gamme de langues	Payant selon API, nécessite une connexion Internet
gtts_ros	Installation via ROS + API Google TTS	Langues disponibles via Google TTS	Frais selon API, nécessite une connexion Internet

TABLE 2.2 : Comparaison des outils de synthèse vocale compatibles avec ROS

Après comparaison, Festival ROS dispose des meilleurs arguments. Le meilleur atout reste la possibilité de l'utiliser en local sans avoir recours à une connexion internet. Malgré sa voix peu naturelle, il n'en reste pas moins très performant et facile à intégrer.

Son installation se fait via le package `festival` sous Linux, il est accompagné d'un pack de voix par défaut. Sur ROS, il faut installer le package `sound_play` utilisé pour lire les sons synthétisés par `festival`. Une fois le package installé et lancé, il suffit de publier une commande contenant les paramètres de lecture (le texte à synthétiser, le volume, le pack de voix) sur le topic par défaut `/robotsound`.

2.5 Dialogue ouvert

Dans le cadre de la robotique sociale, assurer un dialogue ouvert est nécessaire. Ce type de dialogue permet d'offrir une interaction beaucoup plus naturelle avec les utilisateurs, mais également plus captivante. Contrairement aux dialogues structurés où toutes les réponses sont des choix préétablis, l'ouverture de la conversation ici est un plus.

2.5.1 Contrôle de haut niveau via OpenAI

Afin d'assurer au mieux ce dialogue, le robot intègre le modèle de langage GPT d'OpenAi. Avec une configuration soignée et sa base de données sur le langage humain, il offre des réponses personnalisées et naturelles.

L'intégration de GPT dans le système se fait par étapes, de la manière suivante :

1. l'utilisateur parle au robot et l'audio est enregistré.
2. l'audio est repris par Pocketsphinx via le topic `/voice_commands` puis converti en chaîne de caractères.
3. la chaîne de caractère est transmise à GPT pour en extraire les intentions.
4. GPT génère une réponse en fonction de la chaîne.
5. la réponse est récupérée sur Festival qui lance l'audio correspondant via `sound_play`

Cette rapidité de traitement améliore la fluidité des interactions et renforce les traits naturels du robot.

2.5.2 Personnalité artificielle

Cette multitude de moyens de communication verbale (via la reconnaissance et la synthèse vocales) et non verbale (via les expressions faciales et les postures adoptées par le robot) sert à cimenter les fondations de diverses personnalités que le robot peut adopter.

La conception du robot, dans ses détails, fait qu'il est modulaire car chaque métrique est variable. Les expressions faciales peuvent aider à simuler un trait de caractère, d'une expression nerveuse (froncement de sourcils avec lèvres pressées) à une autre plus légère (coins des lèvres et joues levées). Les voix produites par la synthèse vocale s'adaptent également à la personnalité artificielle du robot (ton, rythme et inflexions de la voix).

Enfin, après l'intégration de GPT, ces personnalités ne se manifestent pas seulement par la voix ou les expressions faciales mais aussi par le naturel des réponses fournies. Ces réponses peuvent être adaptées en fonction du contexte donnée au robot.

Chapitre 3

Résultats

Dans cette partie, je présenterai les divers résultats obtenus lors de la conception du robot dans le cadre des différentes postures possibles et de l'implémentation des expressions faciales.

Les illustrations ci-dessous montrent les trois postures que le robot peut adopter :

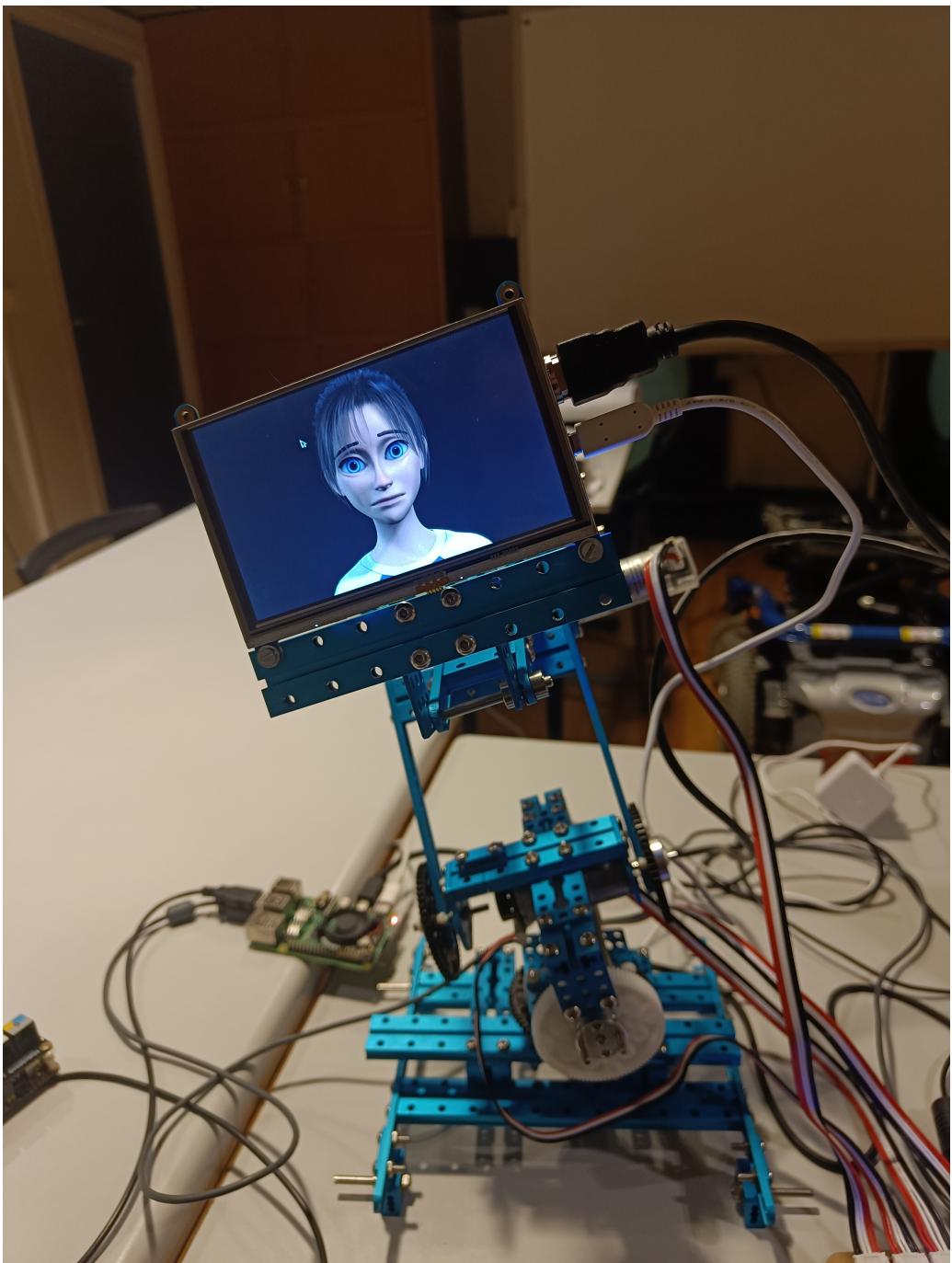


FIGURE 3.1 : Le robot est triste et se renferme sur lui-même

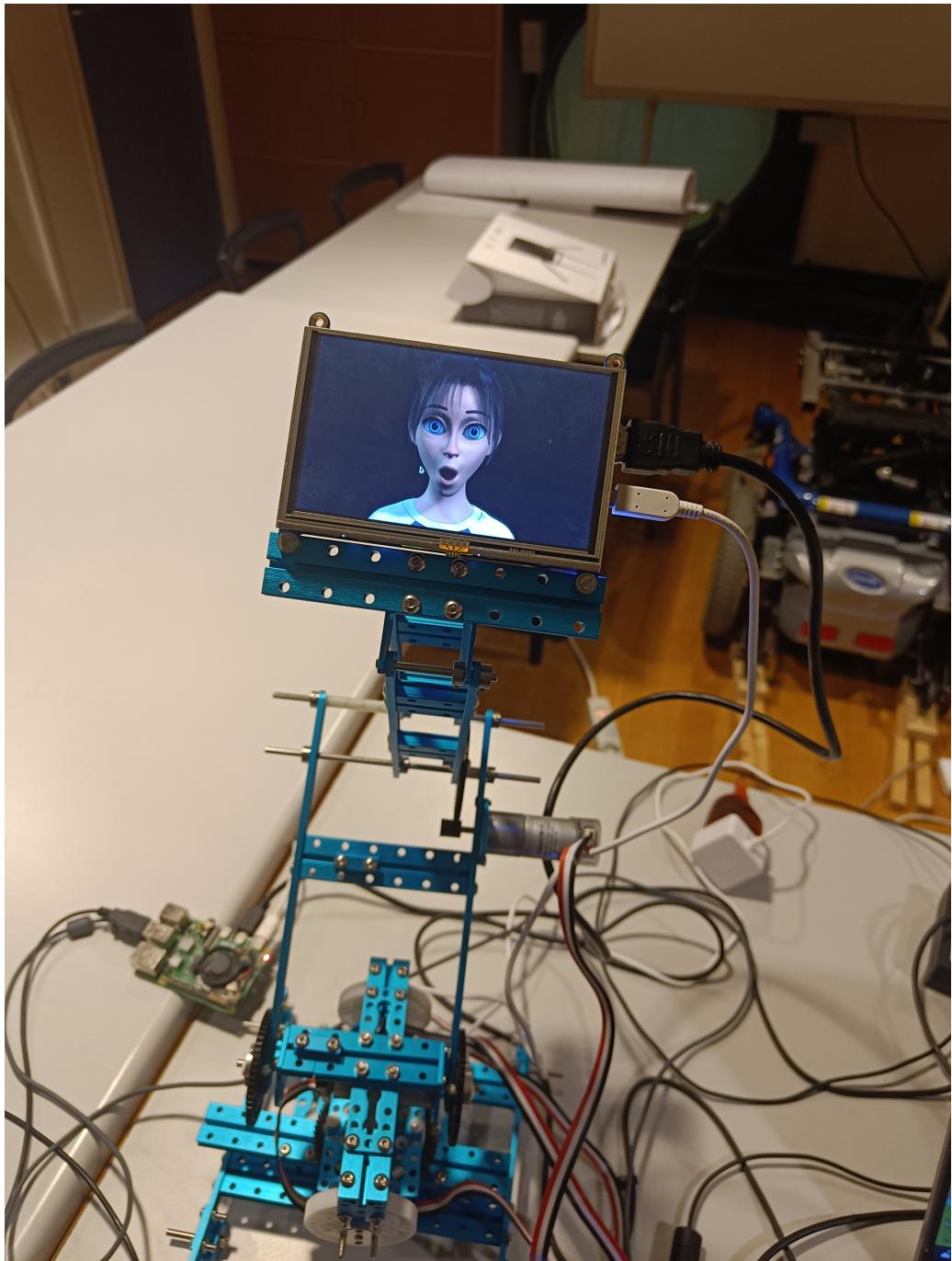


FIGURE 3.2 : Le robot regarde en l'air et est surpris par quelque chose

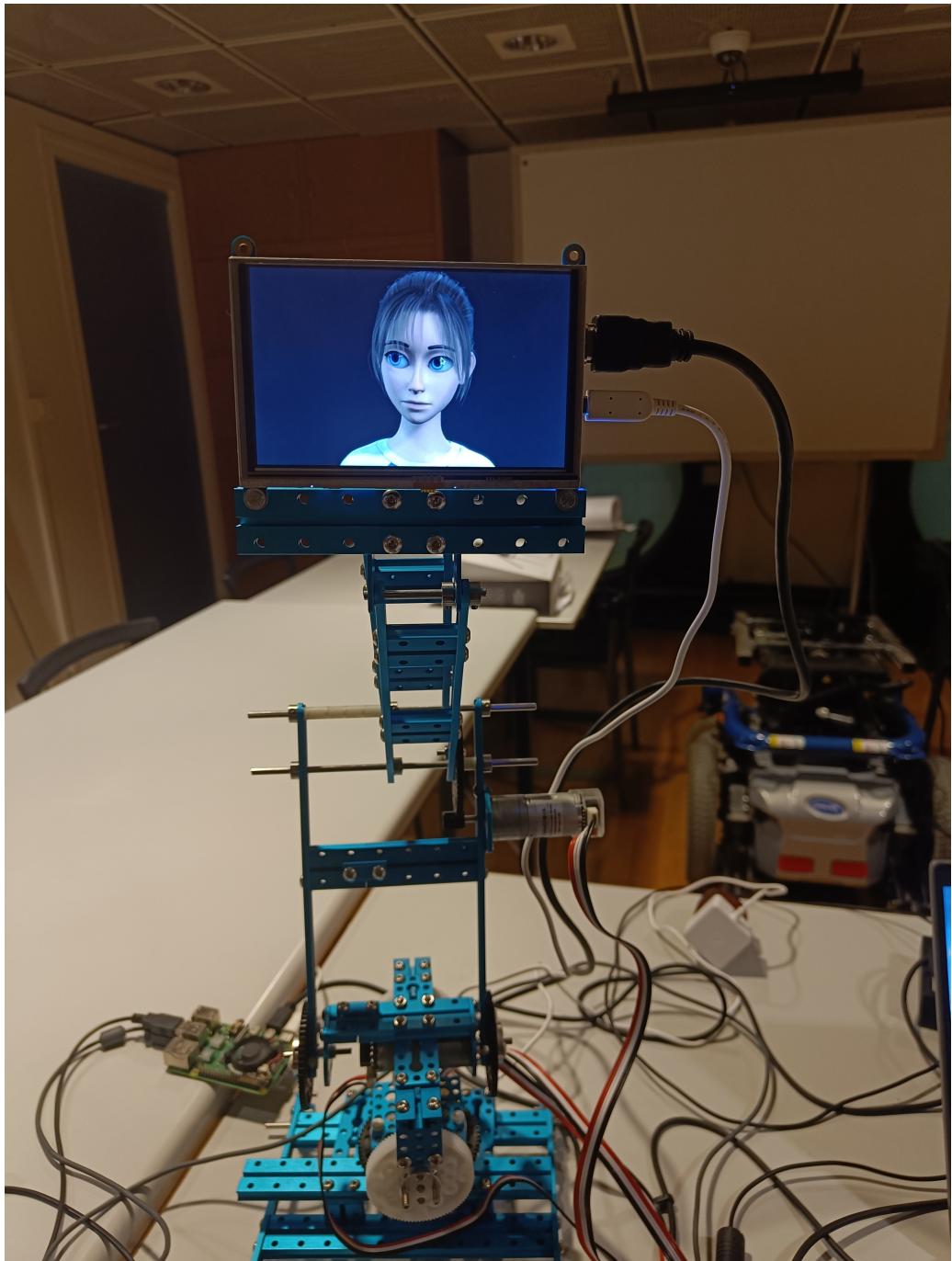


FIGURE 3.3 : Le robot ne fait rien de particulier, c'est son état neutre

Les illustrations suivantes sont des captures d'écran, chacune prise lors de la diffusion des vidéos créées via iClone. Je n'ai pas pu prendre des clichés de toutes les expressions faciales sur le robot.

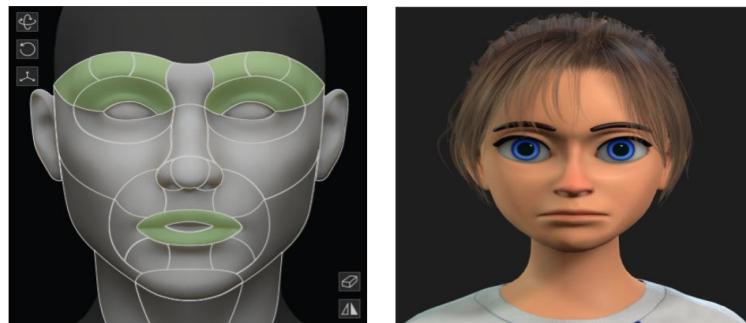


FIGURE 3.4 : Colère

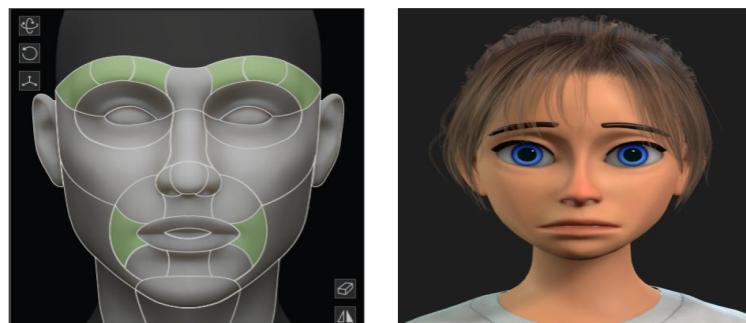


FIGURE 3.5 : Tristesse

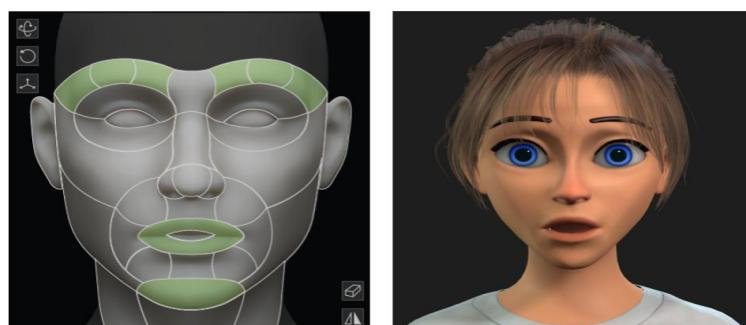


FIGURE 3.6 : Surprise



FIGURE 3.7 : Joie

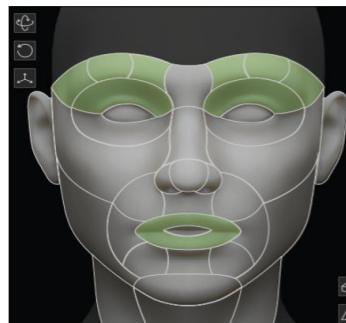


FIGURE 3.8 : Peur

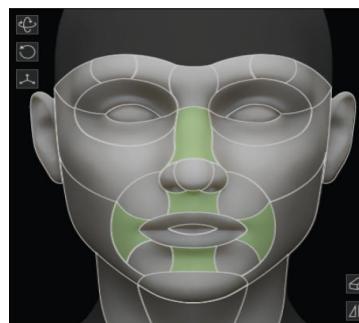


FIGURE 3.9 : Dégout



FIGURE 3.10 : Neutre

FIGURE 3.11 : Parole

Chapitre 4

Discussion

L'objectif du projet était de créer un robot autonome, à partir du kit MakeBlock Ultimave 2.0, capable d'intéragir et d'aider l'enfant atteint de TND de manière naturelle et fluide en intégrant la reconnaissance et la synthèse vocales, les expressions faciales et les postures physiques. Ces éléments offrent des moyens alternatifs à l'enfant atteint de TND pour améliorer leurs compétences en communication.

Pour la partie matérielle, seuls les éléments du kit fourni ont été utilisés (en dehors de l'écran et de la Raspberry Pi). Les angles d'inclinaison et de rotation des moteurs sont assez amples afin de permettre au robot d'effectuer des mouvements plus ou moins libres lui permettant d'adopter les postures nécessaires au contexte lors de son utilisation. La possibilité pour le robot d'effectuer ces mouvements est un plus, la communication non verbale (postures, gestes) et tout aussi essentielle que la communication verbale dans le bon déroulement des interactions sociales.

L'utilisation d'iClone pour l'édition des vidéos et leur retransmission en temps réel sur l'écran permet au robot d'exprimer des émotions de façon visuelle et ludique. Le robot, grâce à ces expressions peut devenir un outil pédagogique pour aider les enfants atteints de TND à reconnaître et comprendre les émotions. Aussi, les plugins de personnalisation et l'outil d'import d'objets 3D facilitent la diversification des personnalités artificielles (vêtements, cheveux, apparence physique) et les font correspondre à différents scénarios si besoin. Cette flexibilité permet d'adapter le robot aux besoins de chaque enfant.

La communication n'étant pas que visuelle, l'intégration de Pocketsphinx et Festival fait communiquer le robot de manière verbale avec l'enfant, les stimuli sonores offrent une nouvelle manière de s'exprimer, renforçant ainsi l'immersion et le contact établi. Ces deux outils réunis permettent à l'enfant d'échanger et de parler avec le robot, offrant ainsi la possibilité de créer des interactions plus ludiques et engageante.

Enfin, l'IA d'OpenAI vient consolider toutes ces fonctionnalités en les liant et en leur permettant d'être toutes automatisées. Et grâce à cette automatisation, il est possible de créer un environnement stable, autonome et prévisible, tous des aspects rassurants pour l'enfante. Ensuite, grâce à l'utilisation d'algorithmes de collecte ciblée de données, il serait alors possible grâce à ce système de mieux cerner les particularités liées au trouble de l'enfant et de personnaliser les traits d'apparence et de caractère du robot pour qu'il soit d'une meilleure aide pour lui.

Cependant, certaines limites subsistent. L'utilisation de vidéos pour l'expression du robot laisse place à des défauts tels que l'impossibilité de parfaire la synchronisation des expressions avec le dialogue. Par exemple, lors d'une conversation avec un rythme rapide et des changements de sujets,

il faudra du temps au robot entre la fin d'une vidéo et la lecture d'une autre. Ce temps, dit de latence, est dépendant des bibliothèques et outils utilisés dans le subscriber `video_subscriber`. Ce manque de fluidité des interactions pourrait gêner le bon déroulement de la thérapie et ainsi causer une perte de l'engagement de l'enfant.

Les problèmes de synchronisation ne touchent pas que la vidéo mais tout aussi bien l'audio que la gestuelle du robot. Synchroniser ces trois éléments ensemble implique des efforts tant en compétences informatiques qu'en ressources matérielles. Pour ce qui est de la boucle d'échange verbale (`Pocketsphinx <-> Festival`) les dictionnaires de langage et de phonétique peuvent différer d'une version à une autre. Des soucis de compatibilités sont souvent rencontrés lors d'utilisation de formules ou de tournures de langage inhabituelles risquant ainsi de dégrader la cohérence et la clarté des échanges entre le robot et l'enfant.

Je retrouve aussi deux autres limites, physiques cette fois. La première est causée par la différence de résolution d'écran d'un moniteur à un autre, certains écrans offriront une meilleure qualité d'image et d'autre un peu moindre. Cette différence impacte directement l'immersion et la crédibilité de l'échange. La seconde est la fiabilité de la structure dans un usage intensif, en effet, jusqu'à présent les tests d'usage ne concernaient que ceux effectués pour le développement de la plateforme, mais à plusieurs reprises j'ai dû recourir à des concessions car le robot nécessite l'utilisation de moteurs plus puissants capables de soutenir la structure en entier avec les moteurs, les cartes (`Arduino` et `Raspberry`) + batterie et l'écran.

Les expressions faciales reproduites, sur la base du FACS d'Ekman, pourraient être améliorées et déclenchées en temps réel par la personnalité artificielle du robot. Une structure et une mise en contexte plus précises lors de la configuration des personnalités du robot pourrait lui faire envoyer des scripts en Python permettant de bouger des parties ciblées du visage sous la base de la décomposition en AU. Cette charge computationnelle nécessite des ressources matérielles plus performantes (Un changement de la `Raspberry Pi` pour un micro-processeur dédié à l'IA par exemple).

Enfin, dans le but de respecter les mesures sanitaires en terme de technologies d'aide pour l'homme, le robot devra être testé, avec le soutien des équipes de l'Hôpital de la Pitié-Salpêtrière, afin de s'assurer de son fonctionnement et qu'il réponde bien aux besoins thérapeutiques des enfants atteints de TND. Ces tests devront de préférence être effectués avec la présence des parents, mieux capables de remarquer ses comportements.

Bibliographie

- [American Psychiatric Association, 2023] American Psychiatric Association, M. C. e. A. B. e. J. G. (2023). *DSM-5-TR Manuel diagnostique et statistique des troubles mentaux, texte révisé*. Elsevier Masson.
- [Anzalone et al., 2018] Anzalone, S., Jean, X., Boucenna, S., Billetti, L., Narzisi, A., Muratori, F., Cohen, D., and Chetouani, M. (2018). Quantifying patterns of joint attention during human-robot interactions : an application for autism spectrum disorder assessment. *Pattern Recognition Letters*, 118.
- [Anzalone et al., 2015] Anzalone, S. M., Boucenna, S., Ivaldi, S., and Chetouani, M. (2015). Evaluating the engagement with social robots. *International Journal of Social Robotics*, 7(4) :465–478.
- [Baccino et al., 2005] Baccino, T., Bellino, C., and Colombi, T. (2005). Mesure de l'utilisabilité des Interfaces. *Hermès, La Revue - Cognition, communication, politique*, pages 1–250.
- [Bertacchini et al., 2023] Bertacchini, F., Demarco, F., Scuro, C., Pantano, P., and Bilotta, E. (2023). A social robot connected with chatgpt to improve cognitive functioning in asd subjects. *Frontiers in Psychology*, 14.
- [Chen et al., 2021] Chen, M., Xiao, W., Hu, L., Ma, Y., Zhang, Y., and Tao, G. (2021). Cognitive wearable robotics for autism perception enhancement. *ACM Transactions on Internet Technology*, 21 :1–16.
- [Dautenhahn, 2021] Dautenhahn, K. (2021). Icra 2020 keynote : Kerstin dautenhahn – human-centred social robotics : Autonomy, trust and interaction challenges.
- [Diehl et al., 2012] Diehl, J. J., Schmitt, L. M., Villano, M., and Crowell, C. R. (2012). The clinical use of robots for individuals with autism spectrum disorders : A critical review. *Research in Autism Spectrum Disorders*, 6(1) :249–262.
- [Feil-Seifer and Matarić, 2008] Feil-Seifer, D. and Matarić, M. J. (2008). Toward socially assistive robotics for augmenting interventions for children with autism spectrum disorders. In *International Symposium on Experimental Robotics*.
- [Kim et al., 2012] Kim, E. S., Berkovits, L. D., Bernier, E. P., Leyzberg, D., Shic, F., Paul, R., and Scassellati, B. (2012). Social robots as embedded reinforcers of social behavior in children with autism. *Journal of Autism and Developmental Disorders*, 43(5) :1038–1049.
- [Lemaignan et al., 2022] Lemaignan, S., Newbutt, N., Rice, L., and Daly, J. (2022). “it’s important to think of pepper as a teaching aid or resource external to the classroom” : A social robot in a school for autistic children. *International Journal of Social Robotics*, 16(6) :1083–1104.

- [López-Sastre et al., 2021] López-Sastre, R. J., Baptista-Ríos, M., Acevedo-Rodríguez, F. J., Pacheco-da Costa, S., Maldonado-Bascón, S., and Lafuente-Arroyo, S. (2021). A low-cost assistive robot for children with neurodevelopmental disorders to aid in daily living activities. *International Journal of Environmental Research and Public Health*, 18(8) :3974.
- [Miller, 1998] Miller, D. P. (1998). *Assistive robotics : An overview*, page 126–136. Springer Berlin Heidelberg.
- [Miskam et al., 2013] Miskam, M. A., Hamid, M. A. C., Yussof, H., Shamsuddin, S., Malik, N. A., and Basir, S. N. (2013). Study on social interaction between children with autism and humanoid robot nao. *Applied Mechanics and Materials*, 393 :573–578.
- [Nasri et al., 2022] Nasri, N., López-Sastre, R. J., Pacheco-da Costa, S., Fernández-Munilla, I., Gutiérrez-Álvarez, C., Pousada-García, T., Acevedo-Rodríguez, F. J., and Maldonado-Bascón, S. (2022). Assistive robot with an ai-based application for the reinforcement of activities of daily living : Technical validation with users affected by neurodevelopmental disorders. *Applied Sciences*, 12(19) :9566.
- [Parajuli, 2024] Parajuli, A. (2024). Dual axis solar tracker arduino project using ldr and servo motors.
- [Pennisi et al., 2015] Pennisi, P., Tonacci, A., Tartarisco, G., Billeci, L., Ruta, L., Gangemi, S., and Pioggia, G. (2015). Autism and social robotics : A systematic review. *Autism Research*, 9(2) :165–183.
- [Puglisi et al., 2022] Puglisi, A., Caprì, T., Pignolo, L., Gismondo, S., Chilà, P., Minutoli, R., Marino, F., Failla, C., Arnao, A. A., Tartarisco, G., Cerasa, A., and Pioggia, G. (2022). Social humanoid robots for children with autism spectrum disorders : A review of modalities, indications, and pitfalls. *Children*, 9(7) :953.
- [Reis, 2023] Reis, J. & Delvenne, V. (2023). Neurodéveloppement et neurodiversité : quelles évaluations pluridisciplinaires ? *Revue Médicale de Bruxelles*, VOLUME 44 - N° 4 - Septembre 2023, 44(4).
- [Santos et al., 2023] Santos, L., Annunziata, S., Geminiani, A., Ivani, A., Giubergia, A., Garofalo, D., Caglio, A., Brazzoli, E., Lipari, R., Carrozza, M. C., Ambrosini, E., Olivieri, I., and Pedrocchi, A. (2023). Applications of robotics for autism spectrum disorder : a scoping review. *Review Journal of Autism and Developmental Disorders*.
- [Scassellati et al., 2012] Scassellati, B., Admoni, H., and Matarić, M. (2012). Robots for use in autism research. *Annual Review of Biomedical Engineering*, 14(1) :275–294.
- [Scassellati et al., 2018] Scassellati, B., Boccanfuso, L., Huang, C.-M., Mademtzi, M., Qin, M., Salomons, N., Ventola, P., and Shic, F. (2018). Improving social skills in children with asd using a long-term, in-home social robot. *Science Robotics*, 3(21).
- [Spatola, 2019] Spatola, N. (2019). L'interaction homme-robot, de l'anthropomorphisme à l'humanisation. *L'Année psychologique*, Vol. 119(4) :515–563.
- [Wainer et al., 2013] Wainer, J., Dautenhahn, K., Robins, B., and Amirabdollahian, F. (2013). A pilot study with a novel setup for collaborative play of the humanoid robot kaspar with children with autism. *International Journal of Social Robotics*, 6(1) :45–65.

Annexes

Annexe A

Code

Ci-dessous le code de motor_subscriber.py :

```
1 #!/usr/bin/env python3
2
3 import rospy
4 from std_msgs.msg import Int32MultiArray
5
6 #FONCTION APPELEE A CHAQUE RECEPTION DE DONNEES VIA "/motors_positions"
7 def head_positions_callback(message):
8     if len(message.data) >= 4:
9
10         #ENREGISTREMENT DES ANGLES DES MOTEURS
11         pos_motor1 = message.data[0]
12         pos_motor2 = message.data[1]
13         pos_motor3 = message.data[2]
14         pos_motor4 = message.data[3]
15
16         #AFFICHAGE DES VALEURS SUR LA CONSOLE
17         rospy.loginfo("Moteur 1: {}".format(pos_motor1))
18         rospy.loginfo("Moteur 2: {}".format(pos_motor2))
19         rospy.loginfo("Moteur 3: {}".format(pos_motor3))
20         rospy.loginfo("Moteur 4: {}".format(pos_motor4))
21
22         #DELAI POUR NE PAS SPAMMER LA CONSOLE
23         rospy.sleep(0.2)
24     else:
25         rospy.logwarn("Erreur: données insuffisantes.")
26
27 #FONCTION D'INITIALISATION DU NOEUD
28 def motor_subscriber():
29
30     #INITIALISATION DU NOEUD POUR ECOUTER LE TOPIC "/motors_positions"
31     rospy.init_node('motor_subscriber', anonymous=True)
32     rospy.Subscriber('motors_positions', Int32MultiArray, head_positions_
33     callback)
34     rospy.spin()
35
36 if __name__ == '__main__':
37     try:
38         motor_subscriber()
39     except rospy.ROSInterruptException:
40         pass
```

Ci-dessous, le code de `motor_publisher.py` :

```
1 #!/usr/bin/env python3
2
3 import rospy
4
5 from std_msgs.msg import Int32
6 from std_msgs.msg import Int32MultiArray
7 from std_msgs.msg import String
8
9 #DEFINITION DES TOPICS ET NOEUDS À UTILISER
10 nodeName='motors_publisher'
11 topicName='motors_commands'
12 videoTopicName='video_commands'
13
14 #INITIALISATION MOTEURS ET PUBLICATION VIA LE TOPIC "/motors_commands"
15 rospy.init_node(nodeName, anonymous=True)
16 publisher1=rospy.Publisher(topicName, Int32MultiArray, queue_size=1)
17
18 #INITIALISATION DU NOEUD DE DEBOGAGE POUR LA VIDEO
19 publisher2=rospy.Publisher(videoTopicName, String, queue_size=1)
20
21 #FREQUENCE DE PUBLICATION DES DONNEES (PAR DEFAUT)
22 ratePublisher=rospy.Rate(1)
23
24
25 while not rospy.is_shutdown():
26     try:
27
28         #SASIE DES VALEURS DES ANGLES/VITESSE
29         print("--MOTOR1--", end='\n')
30         array1_input = input("ANGLE\u00b7(degr\u00e9s)\u00b7[ESPACE]\u00b7VITESSE\u00b7(tours/min)\u00b7:\u00b7")
31         print("--MOTOR2--", end='\n')
32         array2_input = input("ANGLE\u00b7(degr\u00e9s)\u00b7[ESPACE]\u00b7VITESSE\u00b7(tours/min)\u00b7:\u00b7")
33         print("--MOTOR3--", end='\n')
34         array3_input = input("ANGLE\u00b7(degr\u00e9s)\u00b7[ESPACE]\u00b7VITESSE\u00b7(tours/min):\u00b7")
35         print("--MOTOR4--", end='\n')
36         array4_input = input("ANGLE\u00b7(degr\u00e9s)\u00b7[ESPACE]\u00b7VITESSE\u00b7(tours/min):\u00b7")
37         print("--VIDEO--", end='\n')
38
39         #MOYEN ALTERNATIF DE PUBLIER UNE VIDEO
40         video_input = input("Path\u00b7to\u00b7video\u00b7(~/path/to/video.avi)\u00b7:\u00b7")
41
42         #FORMATAGE DES DONNES SAISIES DANS 4 TABLEAUX
43         array1 = [int(x) for x in array1_input.split()]
44         array2 = [int(x) for x in array2_input.split()]
45         array3 = [int(x) for x in array3_input.split()]
46         array4 = [int(x) for x in array4_input.split()]
47
48         #LES TABLEAUX SONT REUNIS EN UN SEUL
49         multi_array = [array1, array2, array3, array4]
50
51         msg = Int32MultiArray()
52
53         for sublist in multi_array:
54             msg.data.extend(sublist)
55
56         #ENREGISTREMENT DU CHEMIN DE LA VIDEO DANS LA CHAÎNE "video_msg"
57         video_msg = String()
58         video_msg.data = video_input
```

```

59
60      #PUBLICATION DES ANGLES/VITESSES VIA "/motors_commands"
61      rospy.loginfo("Publishing: {}".format(msg.data))
62      publisher1.publish(msg)
63
64      #PUBLICATION DU CHEMIN DE LA VIDEO (DEBOGAGE)
65      rospy.loginfo("Publishing: {}".format(video_msg.data))
66      publisher2.publish(video_msg)
67
68  except ValueError:
69      print("Valeur erronnée.")
70  ratePublisher.sleep()

```

Ci-dessous, le code de video_subscriber.py :

```

1 #!/usr/bin/env python3
2
3 import rospy
4 from std_msgs.msg import String
5 import cv2
6 import os
7
8 class VideoPlayer:
9     def __init__(self):
10
11         #CHEMIN DE LA VIDEO PAR DEFAUT
12         self.default_video = "/home/lutin/facepal/src/facepal_pkg/src/videos/
13         idle.mp4"
14         self.current_video = self.default_video
15         self.video_window_name = 'facepal_face'
16         self.new_video_requested = False
17
18         #INITIALISATION DU NOEUD ET ABONNEMENT AU TOPIC "/video_commands"
19         rospy.init_node('video_subscriber', anonymous=True)
20         rospy.Subscriber('video_commands', String, self.video_callback)
21
22     def video_callback(self, msg):
23
24         #ENREGISTREMENT DU CHEMIN DE LA NOUVELLE VIDEO
25         new_video_path = msg.data
26
27         #MISE A JOUR DE LA VIDEO ACTUELLE PAR LA NOUVELLE
28         if new_video_path != self.current_video:
29             self.current_video = new_video_path
30             self.new_video_requested = True
31             rospy.loginfo("Changement de vidéo: {}".format(self.current_video))
32
33
34     def play_video(self, video_path):
35
36         #SI LA VIDEO N'EXISTE PAS
37         if not os.path.isfile(video_path):
38             rospy.logerr("Le fichier vidéo n'existe pas: {}".format(video_path))
39             return
40
41         vid = cv2.VideoCapture(video_path)
42
43         #SI LA VIDEO PEUT ÊTRE LUE

```

```

44     if not vid.isOpened():
45         rospy.logerr("Erreur lors de l'ouverture du fichier:{}".
46                     format(video
47                     _path))
48         return
49
50     #MISE DE LA VIDEO EN PLEIN ECRAN/SUPPRESSION DE L'INTERFACE D'OPTIONS
51     cv2.namedWindow(self.video_window_name, cv2.WND_PROP_FULLSCREEN)
52     cv2.setWindowProperty(self.video_window_name, cv2.WND_PROP_FULLSCREEN,
53                           cv2.WINDOW_FULLSCREEN)
54
55     while vid.isOpened():
56         ret, frame = vid.read()
57         if not ret or self.new_video_requested:
58             break
59
60     #REDIMENSION DE LA VIDEO POUR CORRESPONDRE A L'ECRAN UTILISE
61     frame_resized = cv2.resize(frame, (800,480))
62
63     #ARRÊT DE LA VIDEO QUAND ON CLIQUE SUR "Q"
64     cv2.imshow(self.video_window_name, frame_resized)
65     if cv2.waitKey(30) & 0xFF == ord('q'):
66         rospy.signal_shutdown("User pressed 'q'")
67         break
68
69     vid.release()
70
71     if self.new_video_requested:
72         self.new_video_requested = False
73
74     #REMISE DE LA VIDEO PAR DEFAUT A LA FIN DE LA LECTURE DE LA NOUVELLE
75     if video_path != self.default_video:
76         self.current_video = self.default_video
77
78     #FONCTION PRINCIPALE POUR LA LECTURE DE LA VIDEO
79     def run(self):
80         cv2.namedWindow(self.video_window_name, cv2.WINDOW_NORMAL)
81         while not rospy.is_shutdown():
82             rospy.loginfo("Vidéo en cours:{}".
83                           format(self.current_video))
84             self.play_video(self.current_video)
85
86 if __name__ == '__main__':
87     try:
88
89         #INITIALISATION ET AFFICHAGE DE LA VIDEO
90         video_player = VideoPlayer()
91         video_player.run()
92
93     except rospy.ROSInterruptException:
94
95         pass

```

Ci-dessous, le code facepal_arduino sur la MegaPi :

```

1 #include <MeMegaPi.h>
2 #include <ros.h>
3 #include <std_msgs/Int32.h>
4 #include <std_msgs/Float32.h>

```

```

5 #include <std_msgs/Int32MultiArray.h>
6
7 //INITIALISATION DES 4 MOTEURS
8 MeEncoderOnBoard Encoder_1(SLOT1);
9 MeEncoderOnBoard Encoder_2(SLOT2);
10 MeEncoderOnBoard Encoder_3(SLOT3);
11 MeEncoderOnBoard Encoder_4(SLOT4);
12
13 //INITIALISATION DE NOEUD ROSSERIAL
14 ros::NodeHandle nh;
15
16 //LES VARIABLES DE DONNEES POUR LES ANGLES
17 ///"commands" reçoit les angles
18 ///"positions" envoie les angles
19 std_msgs::Int32MultiArray commands;
20 std_msgs::Int32MultiArray positions;
21
22 //DEFINITION DU TOPIC SUR LEQUEL ROSSERIAL PUBLIE LES ANGLES
23 ros::Publisher pub("motors_positions", \&positions);
24
25 //DEFINITION DU TOPIC DEPUIS LEQUEL ROSSERIAL RECOIT LES ANGLES
26 ros::Subscriber<std_msgs::Int32MultiArray> sub("motors_commands", \&callBackFunction);
27
28 //FONCTIONS PAR DEFAUT POUR INTERROMPRE LES MOTEURS A LA FIN DU MOUVEMENT (1/4)
29 void isr_process_encoder1(void)
30 {
31     if(digitalRead(Encoder_1.getPortB()) == 0)
32     {
33         Encoder_1.pulsePosMinus();
34     }
35     else
36     {
37         Encoder_1.pulsePosPlus();;
38     }
39 }
40
41//(2/4)
42 void isr_process_encoder2(void)
43 {
44     if(digitalRead(Encoder_2.getPortB()) == 0)
45     {
46         Encoder_2.pulsePosMinus();
47     }
48     else
49     {
50         Encoder_2.pulsePosPlus();;
51     }
52 }
53
54//(3/4)
55 void isr_process_encoder3(void)
56 {
57     if(digitalRead(Encoder_3.getPortB()) == 0)
58     {
59         Encoder_3.pulsePosMinus();
60     }
61     else
62     {
63         Encoder_3.pulsePosPlus();;
64     }

```

```

65 }
66
67 // (4/4)
68 void isr_process_encoder4(void)
69 {
70     if(digitalRead(Encoder_4.getPortB()) == 0)
71     {
72         Encoder_4.pulsePosMinus();
73     }
74     else
75     {
76         Encoder_4.pulsePosPlus();;
77     }
78 }
79
80 // LA FONCTION APPELEE A CHAQUE RECEPTION DE DONNEES VIA ROSSERIAL
81 void callBackFunction(const std_msgs::Int32MultiArray &inputMessage){
82
83     // ATTACHEMENT DES FONCTIONS D'INTERRUPTIONS SUR LES 4 MOTEURS
84     attachInterrupt(Encoder_1.getIntNum(), isr_process_encoder1, RISING);
85     attachInterrupt(Encoder_2.getIntNum(), isr_process_encoder2, RISING);
86     attachInterrupt(Encoder_3.getIntNum(), isr_process_encoder3, RISING);
87     attachInterrupt(Encoder_4.getIntNum(), isr_process_encoder4, RISING);
88
89     // MISE DE "commands" A LA MÊME TAILLE QUE "inputMessage" (MESSAGE RECU)
90     commands.data_length = inputMessage.data_length;
91     commands.data = (int32_t*)realloc(commands.data, commands.data_length * sizeof(int32_t));
92
93     // STOCKAGE DES VALEURS DU TABLEAU RECU VIA ROSSERIAL DANS UNE VARIABLE "commands"
94     for (uint16_t i = 0; i < commands.data_length; i++) {
95         commands.data[i] = inputMessage.data[i];
96     }
97
98     // MOUVEMENT DES ROBOTS EN FONCTION DES DONNEES RECUES
99     Encoder_1.move(commands.data[0], commands.data[1]);
100    Encoder_2.move(commands.data[2], commands.data[3]);
101    Encoder_3.move(commands.data[4], commands.data[5]);
102    Encoder_4.move(commands.data[6], commands.data[7]);
103 }
104
105 void setup()
106 {
107     // INITIALISATION DU NOEUD ROSSERIAL
108     nh.initNode();
109     positions.data_length = 3;
110
111     positions.data = (int32_t*)malloc(positions.data_length * sizeof(int32_t));
112
113     // DEFINITION DU PWM DES MOTEURS (PUISANCE MOYENNE DELIVREE)
114     TCCR1A = _BV(WGM10);
115     TCCR1B = _BV(CS11) | _BV(WGM12);
116
117     TCCR2A = _BV(WGM21) | _BV(WGM20);
118     TCCR2B = _BV(CS21);
119
120     // INITIALISATION DES PARAMETRES DES MOTEURS (PAR DEFAUT DEPUIS MBLOCK) (1/4)
121     Encoder_1.setPulse(7);
122     Encoder_1.setRatio(26.9);
123     Encoder_1.setPosPid(1.8,0,1.2);
124 }
```

```

125 Encoder_1.setSpeedPid(0.18,0,0);
126 // (2/4)
127 Encoder_2.setPulse(7);
128 Encoder_2.setRatio(26.9);
129 Encoder_2.setPosPid(1.8,0,1.2);
130 Encoder_2.setSpeedPid(0.18,0,0);
131 // (3/4)
132 Encoder_3.setPulse(7);
133 Encoder_3.setRatio(26.9);
134 Encoder_3.setPosPid(1.8,0,1.2);
135 Encoder_3.setSpeedPid(0.18,0,0);
136 // (4/4)
137 Encoder_4.setPulse(7);
138 Encoder_4.setRatio(26.9);
139 Encoder_4.setPosPid(1.8,0,1.2);
140 Encoder_4.setSpeedPid(0.18,0,0);
141
142 // ACTIVATION DU PUBLISHER
143 nh.advertise(pub);
144
145 // ABONNEMENT AU TOPIC "/motors_commands"
146 nh.subscribe(sub);
147 }
148
149 void loop(){
150
151 // ICI :
152 // ".loop()" SERT A MAINTENIR LE MOTEUR EN MOUVEMENT JUSQU'A L'ANGLE VOULU
153 // ".updateSpeed" SERT A RAFRACHIR LA VITESSE DU MOTEUR
154 // ".updateCurPos" SERT A RAFRAICHER LA POSITION DU MOTEUR (1/4)
155 Encoder_1.loop();
156 Encoder_1.updateSpeed();
157 Encoder_1.updateCurPos();
158 // (2/4)
159 Encoder_2.loop();
160 Encoder_2.updateSpeed();
161 Encoder_2.updateCurPos();
162 // (3/4)
163 Encoder_3.loop();
164 Encoder_3.updateSpeed();
165 Encoder_3.updateCurPos();
166 // (4/4)
167 Encoder_4.loop();
168 Encoder_4.updateSpeed();
169 Encoder_4.updateCurPos();
170
171 // ENREGISTREMENT DES ANGLES MIS A JOUR DANS LE TABLEAU "positions"
172 positions.data[0] = Encoder_1.getCurPos();
173 positions.data[1] = Encoder_2.getCurPos();
174 positions.data[2] = Encoder_3.getCurPos();
175 positions.data[3] = Encoder_4.getCurPos();
176
177 // PUBLICATION DU TABLEAU
178 pub.publish(\&positions);
179
180 // MISE EN BOUCLE DU NOEUD
181 nh.spinOnce();
182
183 // DELAI POUR NE PAS SATURER LA MEMOIRE
184 // SINON PROBLEMES D'ALLOCATION DYNAMIQUE

```

```
185     delay(1);  
186 }  
187 }
```