Projet : "Un robot et des gâteaux"

BLANC Thomas, BODET Martial, DENOU Julien, DUVAL Ludvig

Responsables du projet : Adrien Boussicault et Pierre Lacroix

Master 1 Bioinformatique

06 mars 2018

Table des matières

Introduction

Le sujet "Un robot et des gâteaux" est proposé dans le cadre de l'Unité d'Enseignement "Projet de Programmation". Il fait partie du projet OpenHandicap[1], qui cherche à promouvoir des solutions adaptées, libres de droit et peu coûteuses, au problème d'accessibilité que rencontrent les personnes handicapées. Le projet OpenHandicap est porté par des membres du LaBRI[2] (Laboratoire Bordelais de Recherche en Informatique).

Il a été fixé pour objectif de proposer une solution d'accès à des outils robotisés et autonomes pour une personne lourdement handicapée, pour laquelle le simple fait de piquer un gâteau avec une fourchette et se nourrir est quelque chose d'impossible. Cette personne se servira d'une application mobile ou d'une télécommande. Il doit donc lui être permis de pouvoir émettre des ordres à un bras robotisé afin que celui-ci vienne lui présenter un gâteau pour se nourrir. La solution implémentée devra être libre, accessible, ergonomique et pensée pour être abordable .

1 Contexte

1.1 Analyse du matériel existant

Il a été fournit un bras articulé disposant de 6 moteurs contrôlés par ordinateur, un connecteur USB et un système d'alimentation. Le début du code nécessaire pour les rotation des moteurs est déjà préexistant malgré plusieurs réglages supplémentaires a y appliquer. Il a été mis à disposition une Rasberry Pi modèle 3B, ainsi que des moteurs supplémentaires et différents câblages.

1.2 État de l'art et documentation

1.2.1 Moteurs Dynamixel

L'utilisation de servomoteurs Dynamixel AX-12 [3] est prévue afin de déplacer le robot. Le prototype est déjà équipé de tels moteurs. Ceux-ci seront donc conservés et utilisés dans le cadre de ce projet.

1.2.2 Aimants

Les aimants peuvent être contrôlés via courant [Insérer doc des aimants, et méthode pour donner l'ordre de les alimenter en courant.]

1.2.3 Raspberry Pi

L'utilisation d'un micro-ordinateur Raspberry Pi est prévue pour faire tourner l'algorithme gouvernant les décisions du robot. Il sera intégré au robot et pourra permettre de lancer le programme automatiquement lors du démarrage de la Raspberry [4]

1.2.4 Bibliothèque GPIO

La bibliothèque python *RPi.GPIO* a été développée pour contrôler les ports GPIO présents sur la Raspberry pi. Les commandes de rotations des moteurs pourront être transmise via ces ports. [5] [6]

Une utilisation de la PWM (Pulse Width Modulation), supportée par la bibliothèque *GPIO*, sera également nécessaire pour commander les servomoteurs a partir des ports de la Raspberry pi.[7]

1.2.5 Bibliothèque *Pypot*

La Bibliothèque *Pypot*, développé par l'équipe INRIA Flowers pour controler facilement les moteurs Dynamixels, sera utilisée dans le cadre du projet pour commander nos moteurs.[8]

Le module *Herborist* de *Pypot* est un outil utilisé pour détecter et configurer les moteurs utilisés avec la bibliothèque. [9]

1.2.6 Communication via la bibliothèque *MQTT*

Le protocole de messagerie MQTT (MQ Telemetry Transport), sera implémenté en utilisant la librairie *MQTT* pour le langage python.[10] [11]

1.2.7 Utilisation de marqueurs *ArUco* via *OpenCV*

La bibliothèque graphique *OpenCV* (Open Computer Vision) sera utilisée pour le traitement d'images et vidéo dans le cadre de la détection des marqueurs ArUco. [12]

1.2.8 Calculs mathématiques

Afin de calculer les déplacements du robot il nous faut faire appel à des fonctions mathématiques. Les fonctions utilisées seront basées sur des calculs vectoriels et matriciels. Dans l'algorithme, il sera utilisé de façon non exhaustive les produits scalaires, les produits vectoriels et les matrices de rotation et tout autre type de fonction utile.

2 Cas d'utilisation

2.1 Description du prototype

Le prototype est actuellement composé de six servomoteurs disposés de part et d'autre d'une pièce centrale sur laquelle sera fixée la Rasberry Pi. Chaque moteur a un axe de rotation bien défini. L'ossature du bras est définie comme il suit :

— Un aimant, alimenté en énergie est fixé à l'extrémité
— Un moteur Dynamixel AX-12A est fixé via une pièce à cet aimant. Ce moteur tourne selon l'axe Z.
— Un moteur Dynamixel AX-12A est fixé via une pièce à ce moteur. Ce moteur tourne selon l'axe Y.
— Un moteur Dynamixel AX-12A est fixé via une pièce à ce moteur. Ce moteur tourne selon l'axe Y.
— Un bloc central contenant les connectiques et l'unité de calcul Raspberry Pi.
— Un moteur Dynamixel AX-12A est fixé via ce bloc central. Ce moteur tourne selon l'axe Y.
— Un moteur Dynamixel AX-12A est fixé via une pièce à ce moteur. Ce moteur tourne selon l'axe X.
— Un moteur Dynamixel AX-12A est fixé via une pièce à ce moteur. Ce moteur tourne selon l'axe Z.
— Un aimant alimenté en énergie est fixé à l'extrémité.

Cette disposition lui permet une meilleure mobilité dans l'espace. C'est un bras à 6 degrés de liberté.

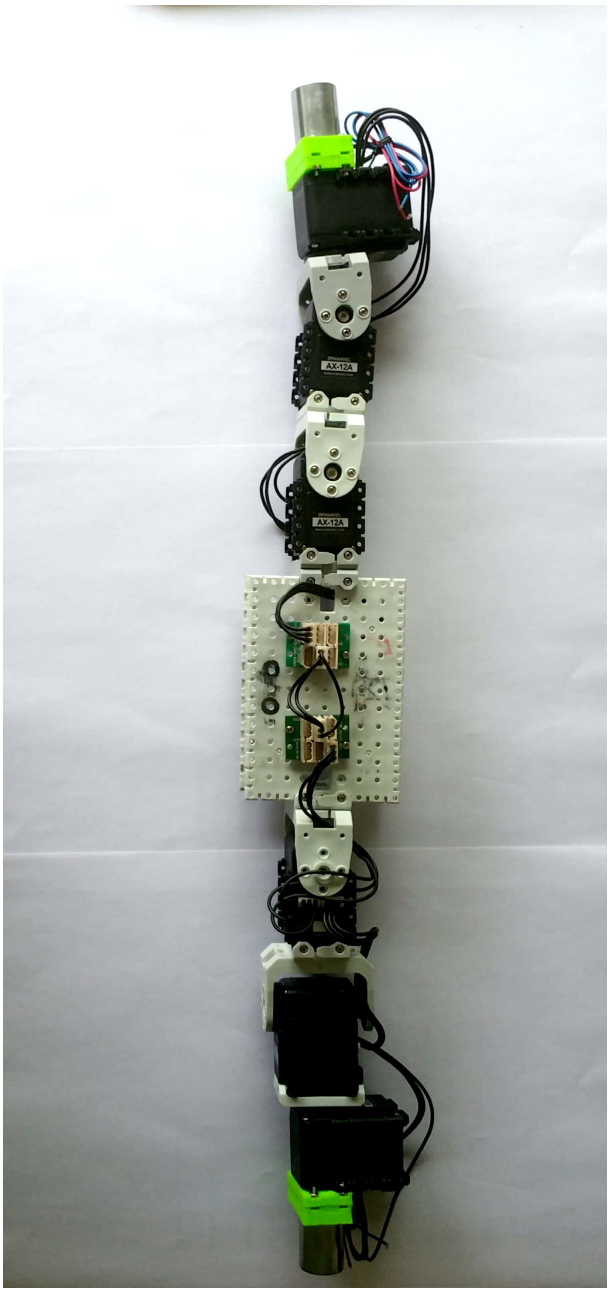

FIGURE 1 – Schéma conceptuel de la scène.

2.1.1 Initialisation du robot

2.2 Description de la scène

Dans l'exemple qui va suivre, une scène fixée a été définie afin de correspondre au cas d'utilisation. Dans cette scène sont fixés les éléments suivants : Au centre de la scène se trouve une table avec une boîte à gâteaux et une personne est attablée face à la boîte. La personne possède un dispositif pour communiquer. Un bras robotisé se trouve quelque part dans la pièce et peut recevoir des communications. Un chemin de plaques métalliques relie le bras robotisé où qu'il soit à la table et la boîte. Un outil se trouve positionné auprès de la boîte. La position de l'utilisateur est considérée comme une zone fixée auprès de la boîte. La position de l'outil est considérée comme une zone fixée au niveau de la boîte également.

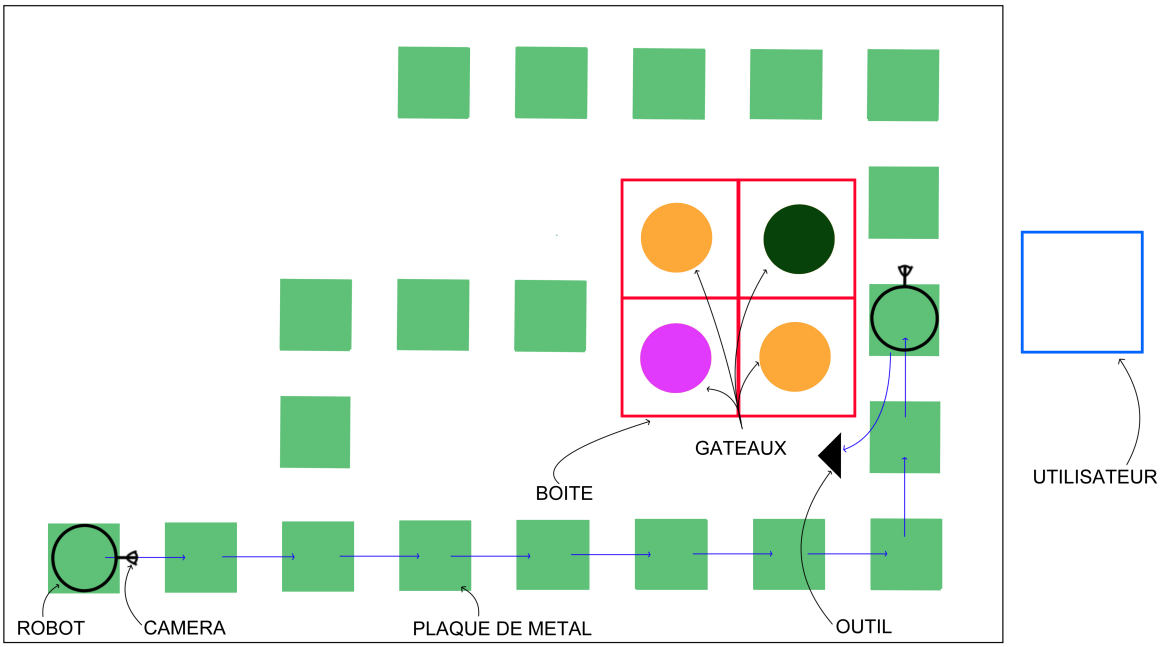

FIGURE 2 – Schéma conceptuel de la scène.

2.2.1 Initialisation du robot

L'utilisateur est donc une personne a mobilité réduite. Cette personne a envie de manger un gâteau, elle doit donc activer le robot de manière simple par un système de télécommande. De préférence, ces systèmes se situeront soit sur la boîte à gâteaux, soit sur un boîtier de type télécommande en possession de la personne. Ce boîtier contiendra un système permettant la communication entre la personne et le bras. (via un système de communications $MQTT$). Ce système devra également être capable de transmettre le choix du gâteau voulu. Le robot se met alors en marche et analyse l'environnement a l'aide de la camera située au plafond ou directement sur lui afin de déterminer l'emplacement de la boite. Si la camera ne se trouve pas sur le robot directement elle devra pouvoir transmettre ses informations.

2.2.2 Repérage de la pièce

Le robot repère la boîte, puis le chemin constitué de plaques métalliques de sa position à la boîte en utilisant éventuellement des marqueurs spécifiques.

2.2.3 Déplacement du robot

Le robot se déplace sur des plaques en métal, disposées de manière équidistantes entre elles de façon à ce que le bras puisse toutes les atteindre en passant de l'une à l'autre. Le bras utilisera donc une combinaison de rotations pour atteindre les plaques et se déplacer. Des aimants fixés sur chaque extrémité du bras, lui permettent de se fixer a ces plaques.

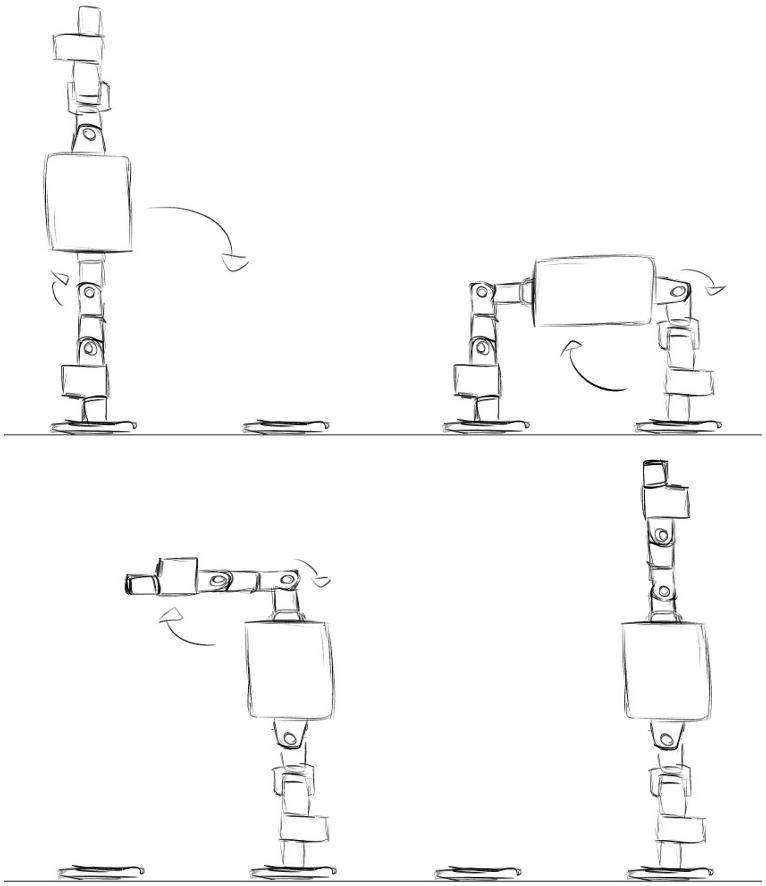

FIGURE 3 – Schéma conceptuel du mode de déplacement.

2.2.4 Fixation à la boîte

Le robot arrive devant la boite, après déplacement. Il lui faut donc se fixer sur une plaque elle même fixée sur la boite. De préférence cette plaque fixée à la boîte sera face à l'utilisateur, ce qui lui permettra par la suite au robot de présenter la part de gâteau sans se déplacer.

2.2.5 Saisie de l'outil

Le robot est donc fixé a la boite a l'aide d'un aimant et d'une plaque en métal. Il doit maintenant attraper une fourchette à l'aide de son second aimant. Une fois a proximité de la fourchette, l'aimant s'active permettant de la maintenir. Le robot ayant attrapé la fourchette se redresse et se positionne face aux gâteaux.

2.2.6 Repérer le gâteau

Le robot doit identifier le gâteau voulu par l'utilisateur. Cela peut se faire via un système de marqueurs spécifiques.

2.2.7 Prendre le gâteau

Après avoir récupérer l'outil et identifier le gâteau, le bras s'inclinera pour "piquer" le gâteau à l'aide de la fourchette. Après avoir piqué le gâteau, le bras se redresse, toujours avec la part de gâteau sur la fourchette.

2.2.8 Présenter le gâteau

Par mouvement de rotation, le robot se retourne en direction de l'utilisateur de façon à ce que le gâteau lui soit accessible. Ce dernier n'aura plus qu'à déguster le gâteau.

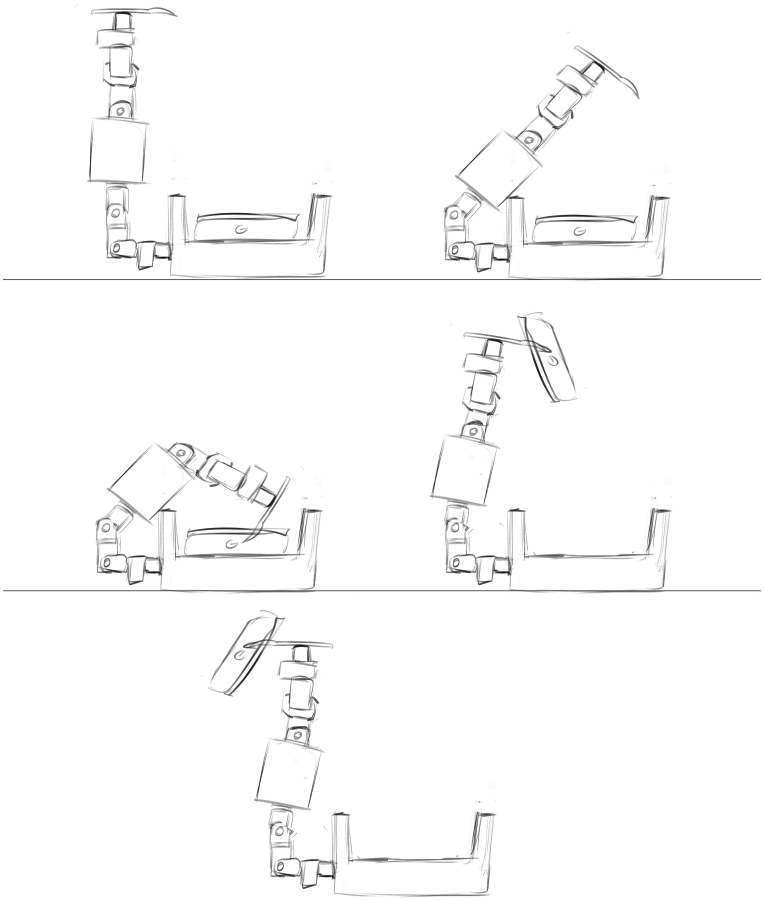

FIGURE 4 – Schéma conceptuel de saisie et présentation du gâteau.

2.3 Description des étapes.

Afin de cadrer les différents éléments de la scène précédemment décrite, celle-ci a été découpée en étapes distinctes allant de la plus réalisable à la version la plus complexe.

2.3.1 Étape numéro 1 :

Le bras est directement fixé a la boîte. Il n'aura donc pas à se déplacer jusqu'à celle-ci. Le robot devra attraper l'outil. Pour cela il est prévu de lui faire effectuer un balayage au dessus de la boîte fixe (dont les dimension et la position sont connues) afin de récupérer l'outil également fixe à l'aide uniquement des interactions entre aimants (celui du robot, celui de l'outil). Puis il devra piquer le gâteau aussi fixé et le présenter a la personne dont la position est connue. Dans ce cas, aucun outil de détection de la scène est mis en place, tous les éléments seront connus et invariables.

2.3.2 Étape numéro 2 :

Dans ce cas, une caméra a été ajoutée a la boîte. De préférence celle-ci sera surélevée par rapport à la scène et reliée directement au bras robotique. Cela permettra donc de pouvoir repérer l'outil a l'aide de marqueur *ArUco*, et de localiser différents gâteaux. Dans cette disposition, le bras et la boîte sont toujours fixes, cependant l'outil et les gâteaux pourront être mobiles à l'intérieur de la boîte et devront être identifiés par la caméra. La manœuvre pour piquer un gâteau, une fois identifié comme celui voulu, et le présenter à l'utilisateur reste inchangée tout comme la position de l'utilisateur.

2.3.3 Étape numéro 3 :

Pour cette phase, il est prévu d'intégrer le déplacement du bras robot d'un emplacement fixe et connu sur la table jusqu'à la boîte dont la position est toujours connue. Un seul chemin est considéré ici comme existant. La camera est toujours considérée fixée au niveau de la boîte à gâteaux. Un système de communication de

type *MQTT* devra être mis en place, afin de communiquer les informations envoyée par la caméra au robot. La caméra devra être capable d'identifier un nouveau type de marqueurs celui des plaques métalliques servant au mouvement. L'utilisateur pourra choisir à distance (via une télécommande) le gâteau de son choix. Le bras devra donc finalement se déplacer, se situer au niveau de la boîte et accéder à la demande de la personne. Ici, la position de l'utilisateur reste la même et les gâteaux, tout comme l'outil se trouvent toujours dans la boîte.

2.3.4 Étape numéro 4 :

Cette étape représente le cas idéal. Pour celle-ci la caméra serait embarquée sur le bras robotisé. L'utilisateur et la boîte pourraient donc être déplacés (ensemble) autour de la table. Plusieurs chemins seraient disponibles pour le mouvement. Le robot devrait trouver le plus court. A nouveau, l'utilisateur à une position fixe par rapport à la boîte et les éléments comme les gâteaux et l'outil se trouvent à l'intérieur de la boîte.

3 Besoins Fonctionnels :

La priorité de chaque besoin est évaluée selon cinq indicateurs :
- Priorité Obligatoire : nous nous engageons à réaliser ce besoin.
- Priorité Haute : nous réaliserons ce besoin si les tâches obligatoires sont toutes réalisées.
- Priorité Moyenne : nous réaliserons ce besoin si les taches de rang supérieures sont toutes réalisées.
- Priorité Basse : ces besoins fonctionnels pourront être réalisés avec un budget ou des délais plus importants.
- Non Réalisable : ces besoins ne seront pas réalisés, ils seront cependant identifiés et décrits mais ne rentrent pas de le cadre immédiat du projet.

3.1 Besoins fonctionnels liés au déplacement du bras :

Calculer la cinématique inverse des rotations successives requises pour que le bras se déplace jusqu'à un point donné. Pour cela une extrémité est positionnée a un emplacement fixe et connu. l'autre extrémité se déplace alors de sa position initiale connue à une nouvelle position et/ou orientation connue également.
Priorité Obligatoire.
Test : Faire bouger le robot dans différentes positions.

Attraper un outil à l'aide d'une fonction de déplacement. C'est à dire déplacer le bras jusqu'à l'outil fixe et utiliser les interactions entre aimants pour accrocher l'outil.
Priorité Obligatoire.
Test : Utiliser différents emplacements fixes de l'outil (distances différentes, calculs d'angles différents).

Piquer un gâteau avec l'outil et le présenter a l'utilisateur. Implémenter un système de sécurité déclenchant l'arrêt du bras dans une position fixe et un signal sonore prévenant l'utilisateur.
Priorité Obligatoire.
Test : Ne pas mettre le gâteau, utiliser différents types de gâteaux.

Implémenter une routine permettant de récupérer un outil. C'est à dire détecter la position de l'outil, ainsi que son orientation et sa présence. Puis attraper l'outil dans le bon sens.
Priorité Haute.
Test : Ne pas mettre l'outil, le retourner.

Implémenter une routine de déplacement standard Pour que le bras se déplace et se remette ensuite en position de repos. La position de repos du robot est décrite en section 2.1 et illustré par le premier schéma de la figure 1. Pour cela, il faut déterminer les valeurs à prendre pour les angles et modifier les valeurs courantes des angles avec les nouvelles valeurs.
Priorité Haute
Test :Faire bouger le robot entre deux plaques métalliques et vérifier sa position finale.

Capacité du bras d'estimer si un déplacement est possible. C'est à dire déterminer si les valeurs d'angles prises par les moteurs sont autorisées.
Priorité Moyenne.
Test : Indiquer des valeurs interdites.

Faire le calcul d'une série de déplacement pour trouver le plus court chemin pour aller jusqu'au point d'arrivée en utilisant les plaques métalliques. C'est à dire développer un graphe de déplacement et y positionner les plaques métalliques comme sommets. Puis trouver la suite d'arcs ou arêtes la plus courte allant de la position initiale connue jusqu'à la position finale connue également. (Algo de Dijikstra, A etoile).
Priorité Basse.
Test : Prévoir plusieurs chemins possibles.

Capacité du bras d'estimer si des obstacles sont présents sur le chemin. C'est à dire déterminer s'il n'y a pas d'obstacles et si c'est le cas choisir l'action a effectuer.(Passer au dessus, contourner ?) [a fixer] et par quel moyens se fait cette détection ? (vision, efforts sur les moteurs dues à la collision ?)
Priorité Non Réalisable.
Test : Placer des obstacles.

Pouvoir se déplacer sur les murs et le plafond. C'est a dire effectuer les transitions de type sol <=> murs ou murs <=> plafonds. La transition sol <=> murs pourra également servir lors du mouvement de fixation à la

boîte.
Priorité Non Réalisable..
Test : Mise en situation.

3.2 Besoins fonctionnels liés a la détection des objets

Détecter les objets de l'environnement via marqueurs *ArUco* et les positionner dans l'espace. C'est à dire détecter un marqueur de type *ArUco* en utilisant la bibliothèque *OpenCV*. Déterminer l'objet représenté par le marqueur. Déterminer l'orientation et la position de l'objet dans l'espace.
Priorité Haute.
Test : Utiliser différents marqueurs (plaques métalliques, outils, boîte), inverser les sens des marqueurs, disposer les marqueurs a des distances différentes.

Détecter l'emplacement de rechargement et rangement du robot en autonomie. Il a été considéré de façon la plus simple qu'une personne tierce rechargera la batterie régulièrement.
Il a tout de même été pensé d'installer un bloc de rechargement encerclé de quatre plaques métalliques. Celles-ci permettrait au robot dès le démarrage de pouvoir changer d'orientation. Il faudrait également penser au type de batterie, une batterie inductible ne nécessitant pas de branchements serait la meilleure solution pour un robot totalement autonome. Le robot devra alors connaître la position du bloc de rechargement et devra être capable de s'y déplacer depuis sa position courante à l'arrêt d'utilisation.
Priorité Non Réalisable.
Test : Vérifier l'autonomie de la batterie, le nombre d'aller retours pouvant être effectués.

3.3 Besoins fonctionnels liés à la communication.

Prévoir un système de communication entre une caméra fixe et le bras robotique. Utiliser pour cela la librairie *MQTT*
Priorité Haute.
Test : Tester une communication entre deux Rasberry Pi ou entre un ordinateur et une Rasberry Pi.

Pouvoir choisir via un système de communication un gâteau parmi plusieurs dans une boite a plusieurs compartiments. Utiliser pour cela les marqueurs *ArUco*
Priorité Haute.
Test : La détection des différents marqueurs et leurs identification peut être testées sur mobiles de façon indépendante.

Pouvoir communiquer avec le bras pour lui donner l'ordre de servir un gâteau particulier (choisi par l'utilisateur). Utiliser pour cela la librairie *MQTT*.
Priorité Moyenne.
Test : Mise en situation.

4 Besoins non fonctionnels

4.1 Besoins non fonctionnels lié au système d'exploitation.

Il a été choisi de conserver l'environnement Python pour le logiciel. Le logiciel devra être Libre.
Priorité Obligatoire.

4.2 Besoins non fonctionnels liés a l'optimisation du robot

Le temps de calcul du déplacement devra être optimisé pour une bonne rapidité de mouvements. En effet une demande de l'utilisateur doit pouvoir s'effectuer dans des délais raisonnables. C'est à dire inférieurs à quinze minutes si la mise en place de la scène le permet. (C'est a dire que l'utilisateur ne se trouve dans un manoir au troisième étage et le robot au sous-sol.)
Priorité Moyenne.
Test : Essayer plusieurs algorithmes et les comparer entre eux.

La vitesse de rotation des moteurs devra être réglée pour ne pas faire de mouvement trop lents ou brusques. Afin d'avoir un robot stabilisé et sécurisé. La force du robot ne doit pas excéder le nombre de joules autorisés par la législation dans le cadre d'une arme. Elle doit donc être inférieur à 20 joules.
Priorité Moyenne.

4.3 Besoins non fonctionnels liés a l'utilisateur

Le système de commande du bras doit être accessible, ergonomique et simple d'utilisation, afin de convenir a des personnes handicapées ou a des personnes âgées.
Priorité Moyenne.
Test : Demander retour à des personnes concernées.

4.4 Besoins non fonctionnels financiers :

Le système ciblant des personnes handicapées, beaucoup d'entre elles n'ont que pour unique revenu l'AAH (Allocation aux Adultes Handicapés) dont le montant est de 810,89€.[13]Il est choisi de réduire ainsi le coût au maximum pour permettre à la solution de fonctionner selon les contraintes budgétaires du client.
En plus du bras robotisé, l'utilisateur devra être muni d'une boîte contenant la nourriture, d'outils comme la fourchette, de plaques de métal fixées dans son habitat.
Afin de parfaire le projet une estimation du coût du prototype pourra être effectuée. **Priorité Moyenne.**
Test : Estimer le coût de différents de prototypes et le comparer à celui utilisé.

Références

[1] Open handicap. https://open-handicap.labri.fr/. Consulté : 06-03-2018.

[2] Site du labri. http://www.labri.fr/. Consulté : 06-03-2018.

[3] Référence du servomoteur dynamixel. https://www.generationrobots.com/media/Dynamixel-AX-12-user-manual.pdf. Consulté : 06-03-2018.

[4] Auto exécution des programmes sur un raspberry pi. https://www.dexterindustries.com/howto/auto-run-python-programs-on-the-raspberry-pi/. Consulté : 06-03-2018.

[5] Bibliotheque gpio. https://pypi.python.org/pypi/RPi.GPIO. Consulté : 06-03-2018.

[6] Tutoriel pour l'utilisation de la bibliotheque gpio. http://deusyss.developpez.com/tutoriels/RaspberryPi/PythonEtLeGpio/. Consulté : 06-03-2018.

[7] Tutoriel pour le pilotage d'un servomoteur via les pins gpio de la raspberry pi. https://www.toptechboy.com/raspberry-pi/raspberry-pi-lesson-28-controlling-a-servo-on-raspberry-pi-with-python/. Consulté : 06-03-2018.

[8] Bibliotheque pypot. https://poppy-project.github.io/pypot/. Consulté : 06-03-2018.

[9] Tutoriel d'installation du logiciel herborist. https://github.com/poppy-project/poppy-humanoid/blob/master/hardware/doc/fr/adressage_dynamixel.md. Consulté : 06-03-2018.

[10] Bibliotheque mqtt pour python. https://pypi.python.org/pypi/paho-mqtt/1.2. Consulté : 06-03-2018.

[11] tutoriel pour la communiquation mqtt. http://www.ev3dev.org/docs/tutorials/sending-and-receiving-messages-with-mqtt/. Consulté : 06-03-2018.

[12] tutoriel pour la communiquation mqtt. https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html. Consulté : 06-03-2018.

[13] Source du service public. https://www.service-public.fr/particuliers/vosdroits/F12242. Consulté : 06-03-2018.