REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département D'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état en Automatique

Réalisation d'un bras manipulateur pour des applications de pick & place à base de machine vision.

Réalisé par :

Ibrahim HADDAD

Salah Eddine ABID

Présenté et soutenue Publiquement le : 13/07/2021

Composition Du Jury :

Président : M.S BOUCHERIT	Grade : Professeur, ENP ALGER
Examinateur : O.STIHI	Grade : Enseignant chercheur, ENP ALGER
Encadreur : M.TADJINE	Grade : Professeur, ENP ALGER
Co-Encadreur : M.CHAKIR	Grade : MCA,ENP ALGER

ENP 2021

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département D'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état en Automatique

Réalisation d'un bras manipulateur pour des applications de pick & place à base de machine vision.

Réalisé par :

Ibrahim HADDAD

Salah Eddine ABID

Présenté et soutenue Publiquement le : 13/07/2021

Composition Du Jury :

Président : M.S BOUCHERIT	Grade : Professeur, ENP ALGER
Examinateur : O.STIHI	Grade : Enseignant chercheur, ENP ALGER
Encadreur : M.TADJINE	Grade : Professeur, ENP ALGER
Co-Encadreur : M.CHAKIR	Grade : MCA,ENP ALGER

ENP 2021

الملخص:

تشرح هذه المذكرة الخطوات التي قمنا بها من أجل صنع ذراع آلية للقيام بمهام حمل و وضع و هذا بالاعتماد على تقنيات رؤية الآلة. التحكم بالذراع الآلية تم بواسطة معالج Arduino Mega 2560و تحت نظام تشغيل الروبوتات .ROS تم استخدام كاميرا من أجل التقاط صور للأشياء المراد حملها حيث تم برمجة الذراع الآلية على أن تقوم بحمل هذه الأشياء و وضعها في أماكن محددة مسبقا بناء على نوع الشيء المُلتَقط بواسطة الكاميرا . معالجة الصور المُلتقَطة و تصنيف الأشياء المصورة يتم بواسطة الحاسوب الذي يُرسل الأوامر للمعالج Arduino Mega 2560 و الذي بدوره يُحوّلها الى المحركات المسؤولة على تدوير كل مفاصل الذراع الآلية .

الكلمات المفتاحية: ROS ، رؤية الآلة ، Arduino ، Ramps ، URDF، Moveit ! ،Rviz، Gazebo ، Tensorflow API، OpenCV.

Abstract :

This work presents the steps taken in order to construct a robotic arm capable of doing different tasks of picking and placing different objects, using machine vision. The control of this robotic arm was achieved through the Arduino Mega 2560 processing board, under The Robotics Operating System (ROS). A webcam was used to take feeds of the objects that we wish to pick, then based on the analysis of these pictures, the robotic arm was programmed to move these objects to predefined locations based on the nature of the object and its type. The task of processing the pictures and classifying them was done through a computer (Laptop), which is responsible for sending the necessary commands to the Arduino Mega 2560 board, this board then in its turn converts these commands to each stepper motor allowing us to turn each robotic joint accordingly.

Keywords : ROS, Machine vision, Arduino, Ramps, URDF, MoveIt !, Rviz, Gazebo, TensorFlow API, OpenCv.

Résumé :

Ce travail présente les étapes suivies pour construire un bras de robot capable d'effectuer des différentes tâches de Pick & Place, pour une variété d'objets, utilisant la vision des machines. La commande du bras était achevée par la carte Arduino Mega 2560, sous le MiddleWare ROS (Robotics Operating System). Une webcam est utilisée pour prendre des feeds des objets qu'on souhaite manipuler. Ces derniers seront analysés par le CPU du PC pour classifier les objets captés par la webcam. En fonction de ces données de classification, le ROS envoie des commandes à la carte Arduino Mega 2560, qui à son tour les transferts à chaque moteur pas à pas, pour pouvoir déplacer le bras à des positions prédéfinies en fonction des classes des objets en question.

Mots Clés : ROS, Machine vision, Arduino, Ramps, URDF, MoveIt !, Rviz, Gazebo, TensorFlow API, OpenCv.

Remerciements

Nous remercions **Dieu**, le tout puissant de nous avoir donné la patience et la force durant toutes ces années d'étude. Les travaux présentés dans ce mémoire ont été effectués au sein de L'entreprise CELL et du Laboratoire de Commande des Processus (LCP) de l'Ecole Nationale Polytechnique.

Ce travail que nous présentons a été effectué sous la direction de Monsieur M. Tadjine, professeur à l'Ecole Nationale Polytechnique, et Monsieur M.Chakir, enseignant chercheur à l'Ecole Nationale Polytechnique. Nous les remercions également pour leur orientation pédagogique dans l'élaboration de ce mémoire.

Nous tenons à remercier Monsieur Ketfi Cherif Fares, Directeur de l'entreprise « CELL », pour avoir nous donner de la chance pour travailler sur ce projet et de nous assurer tous les moyens pour le compléter.

Enfin, nous remercions vivement tous qui nous a aidé avec le matériel nécessaire pour réaliser ce projet. Qu'ils trouvent ici l'expression de notre profonde gratitude.

Dédicace :

A mes **très chers parents**, pour tous leurs sacrifices et leur soutien tout au long de mes études. Que **Dieu** les protège !

A ma sœur Imane, mon frère Aboubaker,

A toute la famille HADDAD et la famille Benyahia,

A mes amis et ma deuxième famille :
Yasser,Abderrahim,Rachid,Bilal,Mohcine,Ilyas,Dhiae Eddine, Abd El-Mouiz ,Tayebe,El-qalaa,Salah Eddine , toute personne qui travaille pour la renaissance de la nation des musulmans et à tous ceux que j'aime,

Je dédie ce travail.

Ibrahim HADDAD

Dédicace :

A celle et celui qui ont toujours garni mon chemin avec force et lumière, mes très chers parents.

A mes chers frères et mes chères sœurs,

A mes amies : Moncef ALLEL, Sofiane AOUF, Oussama TELLIA, Omar Aîssani, Djebbar Rabah, Amine Lalouche,

A toute personne qui m'a aidé à franchir un horizon dans ma vie,

Je dédie ce travail.

Salah Eddine ABID.

Table des matières :

Liste des Figures

Liste des Tableaux

Liste des Symboles

Introduction Générale : 14

1 Rappelle sur les bases de la robotique et modélisation du bras de robot..... 17

1.1 Introduction : 17

1.2 Rappels et Généralités : 17

1.2.1 Les manipulateurs en série : 17

1.2.2 Le nombre de degrés de liberté : 17

1.2.3 Un effecteur ou un organe terminal : 17

1.2.4 L'espace de travail : 18

1.2.5 La cinématique : 19

1.2.6 Transformations homogènes : 19

1.2.7 Le modèle géométrique direct : 21

1.2.8 Le modèle géométrique inverse : 21

1.2.9 Paramètres Denavit–Hartenberg : 21

1.3 Application sur le bras robotique BCN-3D-Moveo : 24

1.3.1 Matrice Denavit–Hartenberg : 25

1.3.2 Modèle géométrique direct : 25

1.3.3 Modèle géométrique inverse : 26

1.4 Conclusion : 30

2 Dimensionnement du robot : 32

2.1 Introduction : 32

2.2 Partie électrique : 32

2.3 Les cartes électroniques : 38

2.4 Partie mécanique : 43

2.5 Conclusion : 53

3 L'ensemble des logiciels utilisés et leurs fonctionnements : 55

3.1 Linux/GNU : 55

3.2 Ubuntu : 55

3.3 ROS : 56

3.4 URDF : 58

3.5 SolidWorks : 59
3.6 ROS Visualizer RViz : 63
3.7 TensorFlow 68
3.8 OpenCV : 69
3.9 Gazebo : 73
3.10 Conclusion : 74
4 La commande du robot, la détection des objets et l'exécution des trajectoires prédéfinies : 76
4.1 Introduction : 76
4.2 La commande du robot : 76
4.3 Le processus d'entrainement des modèles avec L'API de TensorFlow : 85
4.4 Etiquetage des images : 88
4.5 Génération des données d'entraînement : 89
4.6 Création des étiquettes et configuration de l'entrainement : 90
4.7 Lancement de l'entrainement : 90
4.8 Pick and Place spécifique à l'objet utilisant le bras BCN-3D Moveo : 91
4.9 Conclusion : 92
5 Résultats, Avantages et perspectives : 94
5.1 Introduction : 94
5.2 Résultats : 94
5.3 Les Avantages du bras de robot : 99
5.4 Conclusion générale et Perspectives : 99
Bibliographie : 101

Liste des figures :

Figure 1: des bras robotiques effectuant des différentes tâches..... 17
Figure 2: l'espace de travail d'un bras robotique industriel à 6ddl..... 18
Figure 3: des repères liés aux manipulateur et différents objets dans son environnement [2]. 19
Figure 4: Translation et rotation d'un repère B par rapport à un repère A [1]..... 20
Figure 5: Relation entre le modèle géométrique direct et inverse [1]..... 21
Figure 6: les paramètres de Denavit-Hartenberg..... 22
Figure 7: le bras robotique BCN-3D-Moveo..... 24
Figure 8: Représentation selon la convention DH [6]. 24
Figure 9: les types et les localisations des moteurs sur le bras robotique [8]. 32
Figure 10: Nema 17 SM42HT47-1684..... 33
Figure 11: Dimensions et câblage du Nema 17 SM42HT47-1684..... 34
Figure 12: Nema 23 SM57HT76-3004A..... 34
Figure 13: Nema 17 SM42HT33-1334A..... 35
Figure 14: Dimensions et câblage du Nema 17 SM42HT33-1334A. 36
Figure 15: Nema 23 SM57HT112-3004A..... 36
Figure 16: Dimensions et câblage du Nema 23 SM57HT112-3004A..... 37
Figure 17:Servo moteur 55g 7300-SERVO213..... 37
Figure 18: Driver TB6560 connecté avec un moteur pas à pas [10]. 38
Figure 19: le Ramps v1.4. 39
Figure 20: les composants du Ramps v1.4..... 40
Figure 21: La carte Arduino 2560. 41
Figure 22: Schéma global de la partie électronique [13]..... 42
Figure 23: Articulation 1 [15]..... 43
Figure 24: la base rotative [15]..... 43
Figure 25: La plaque rotative [15]..... 44
Figure 26: La forme finale de la base [15]. 44
Figure 27: Le moteur pas à pas (Nema 17) qui assure le mouvement de l'articulation 1 [15]. 45
Figure 28: la base de l'articulation 2 [15]. 45
Figure 29: L'emplacement des deux moteurs Nema 23 [15]. 46
Figure 30: Les deux pièces d'engrenages de l'articulation 2 [15]..... 46
Figure 31: Le tendeur qui permet le réglage de la tension des courroies [15]. 47
Figure 32: L'assemblage de l'articulation 2 [15]. 47
Figure 33: La base de l'articulation 3 [15]. 48
Figure 34: La forme finale de l'articulation 3 [15]..... 49
Figure 35: L'ajout du Moteur Nema 17 et le coupleur à la pièce [15]..... 49
Figure 36: Montage de l'articulation 4 [15]..... 50
Figure 37: La base de l'articulation 5 [15]. 51
Figure 38: L'engrenage de l'articulation 5 [15]..... 51

Figure 39: Montage de l'articulation 5 [15]. 52
Figure 40: L'assemblage du gripper et sa fixation sur l'articulation 5 [15]. 52
Figure 41: Assemblage des Articulations 4 et 5 + le gripper [15]. 53
Figure 42: La forme finale du bras de robot après assemblage [15]. 53
Figure 43: Les logs de quelques distributions de Linux [20]. 55
Figure 44: Terminale sous Ubuntu. 56
Figure 45: Les distributions de ROS [21]. 57
Figure 46: Quelques robot fonctionnant sur ROS : Pepper (a), REEM-C (b), Turtlebot (c), Robonaut (d), et Universal Robots (e) [21]. 57
Figure 47: Modélisation du bras Moveo par SolidWorks pour la génération du fichier URDF. 60
Figure 48: Définition des Liens du robot selon leurs ordres par l'outil SW2URDF de SolidWorks. 61
.Figure 49: Spécification des paramètres supplémentaires du fichier URDF pour chaque lien du bras Moveo. 62
Figure 50: Résultat de la génération du fichier URDF par l'outil SW2URDF. 62
Figure 51: Le Logo de RViz. 63
Figure 52: MoveIt ! Setup Assistant. 64
Figure 53: Le modèle du bras Moveo dans MoveIt ! 64
Figure 54: génération de la matrice de collision. 65
Figure 55: Définition des planning groups. 65
Figure 56: définition des positions pour le bras Moveo. 66
Figure 57: L'ajout des régulateurs pour chaque articulation. 66
Figure 58: Génération des Packages MoveIt ! 67
Figure 59: Manipulation du robot dans RViz. 67
Figure 60: Logo du TensorFlow. 68
Figure 61: Logo du OpenCV. 69
Figure 62: Une matrice de pixels. 70
Figure 63: Représentation d'une image par une matrice de pixels. 70
Figure 64: Différentes Types de Seuillage. 71
Figure 65: Différentes Types de Contours. 71
Figure 66: Multiplication d'une image par un filtre pour la détection des objets. 71
Figure 67: Exemple du résultat d'un filtre de lissage. 72
Figure 68: Résultat de l'érosion et la dilatation d'une image. 72
Figure 69: Application du filtre de l'ouverture pour la suppression du bruit. 73
Figure 70: Application du filtre de la fermeture pour la suppression du bruit. 73
Figure 71: Logo du Gazebo. 73
Figure 72: Simulation de l'application pick & place avec le bras Moveo dans le simulateur Gazebo. 74
Figure 73: L'espace de travail « catkin_ws ». 76
Figure 74: Le dossier « robot ». 77
Figure 75: Le dossier « robot_description ». 77
Figure 76: Le dossier « robot_moveit ». 78
Figure 77: Le dossier « robot_moveit_arduino ». 78

Figure 78: Le dossier « robot_moveit_config ». 79
Figure 79: Le dossier « object_detector_app». 80
Figure 80: Le dossier « object_detection»..... 80
Figure 81: Les scripts « object_detection» et « object_detection_multithreading »..... 81
Figure 82: Le résultat de l'éxecution. 82
Figure 83: Le résultat de l'éxecution. 83
Figure 84: Les nœuds actifs. 85
Figure 85: Anaconda [35], CUDA [36] et cuDNN [37]. 86
Figure 86: le répertoire d'installation de TensorFlow. 87
Figure 87: Exemple d'utilisation de Labe1Lmg. 89
Figure 88: Lancement de l'entrainement. 90
Figure 89: Graphe de la perte. 91
Figure 90 : les pièces imprimés par l'imprimante 3D. 94
Figure 91: la construction de chaque articulation individuelle. 95
Figure 92: l'assemblage mécanique de toutes les articulations ensemble. 95
Figure 93: Câblage du Circuit de commande. 96
***Figure 94 : La commande en temps réel du bras de robot avec Rviz et Moveit !*..... 97**
Figure 95: Quelques manipulations de Pick & Place spécifiques. 98
Figure 96: le schéma qui résume le processus. 99

Liste des tableaux :

Tableau 1 : Matrice des paramètres DH pour le manipulateur BCN-3D Moveo. 25
Tableau 2 : les moteurs utilisés et leurs spécifications. 33
Tableau 3 : Le courant d'alimentation du moteur. 38
Tableau 4 : Le mode d'excitation. 39
Tableau 5 : Le courant de recirculation. 39

Liste des symbols :

$\vec{r}_p^A$: le vecteur de position du point p dans le repère A.

$\vec{r}_p^B$: le vecteur de position du point p dans le repère B.

${}_B^A R$: la matrice de rotation du repère B vers le repère A.

$\vec{r}_{B0\to A0}^A$: le vecteur de translation de l'origine B_0 vers l'origine A_0.

α_{i-1}: angle de twist entre les axes z des liens i-1 et i.

d_{i-1}: distance de translation entre les axes z des liaisons i-1 et i.

r_i: distance de translation entre les axe x-des liens i-1 et i.

θ_i: angle de rotation entre les axes x des liens i-1 et i.

${}_j^{j-1}T$: matrice de transformation homogène du repère j-1 vers le repère j.

Cj : le cosinus de l'angle Theta j.

Sj : le sinus de l'angle Theta j.

Cij : le cosinus de l'angle (Theta j + Theta i).

Sij : le sinus de l'angle (Theta j + Theta i).

I : le courant en Ampère.

SW : le switch du driver.

Introduction générale :

Les bras manipulateurs sont parmi les outils les plus puissants présents au niveau de la fabrication industrielle. Ces bras apportent l'avantage d'être très précis et très robustes dans leurs tâches. Ils différent selon leurs tailles, nombres d'articulations et les tâches à effectuer, ceci les rend une pièce centrale au niveau des usines et des grands sites de fabrication.

Souvent, il est nécessaire de réaliser des tâches qui sont en relation avec la nature des objets à manipuler, ceci peut être difficile à implémenter par les méthodes classiques dans le domaine de la robotique, car elles sont susceptibles à des erreurs qui peuvent engendrer des conséquences terribles. C'est pour ça qu'on s'est intéressés au niveau de ce travail à l'implémentation d'une plateforme à base de Machine vision pour garantir le bon fonctionnement de ces robots. Ça fait partie du domaine de la robotique moderne et l'intelligence artificielle, donc ça nous pousse à essayer de développer cette branche de la robotique et à tenter de la rendre vastement utilisée.

Afin de pouvoir effectuer des tâches basées sur la machine vision, un capteur à base de vision doit être introduit dans le processus de fonctionnement des robots. Ces capteurs peuvent être une webcam en 2D, un scanner Laser, une caméra qui construit les images en 3D (Kinect), etc. les données qui arrivent depuis ces capteurs seront analysées et traitées afin de pouvoir produire les commandes nécessaires aux robots. Ces capteurs sont des outils très utiles, et ils permettent d'introduire un nouvel aspect dans le monde de la robotique, leurs implémentations avec les outils et les API de l'intelligence artificielle les rends encore très puissants et leurs permet la réalisation de plusieurs tâches qui étaient une fois considérées impossible.

Les API de l'intelligence artificielle sont des programmes qui permet la classification des objets à partir des données visuelles (offertes par les capteurs), en implémentant des algorithmes d'entrainement différents. La rapidité de ces processus d'entrainement dépend de la puissance du calculateur utilisé, un calculateur puissant pourra effectuer la classification très rapidement tandis qu'un calculateur peu puissant peut prendre beaucoup de temps pour réaliser le même processus.

Cette classification sera ensuite utilisée pour commander les robots en leurs spécifiant des ordres précis qui dépendent de la nature des objets détectés (les classes des objets).

Dans le cadre de cette mémoire, nous allons implémenter ces principes pour réaliser la commande visuelle du bras manipulateur BCN-3D Moveo.

Cette mémoire est organisée comme suit :

Le premier chapitre est consacré à la citation des bases de la robotique et la modélisation du bras de robot et la génération de ses modèles géométriques directe et inverse.

Le deuxième chapitre aborde le dimensionnement du bras de robot. Dans ce chapitre nous détaillons les spécifications des composants et éléments mécaniques et électriques nécessaires pour la réalisation de ce bras de robot.

Dans le troisième chapitre, nous présentons les logiciels et les outils informatiques que nous avons utilisé pour réaliser la commande de ce bras manipulateur ainsi que la tâche de l'intelligence artificielle. Nous avons spécifié pourquoi le choix de ces logiciels et outils informatiques et leurs avantages.

Dans le quatrième chapitre nous avons détaillé les étapes que nous avons suivi pour faire fonctionner le bras manipulateur et la tache de détection des objets et leur classification. Nous avons détaillé comment entrainer un modèle à base de neurones artificiels et comment créer un classificateur d'objets avec L'API de TensorFlow. Nous avons aussi détaillé les commandes linux que nous avons utilisé sous ROS pour exécuter les programmes et les scripts nécessaire pour la commande en temps réel du bras manipulateur.

Dans le dernier chapitre, nous avons présenté les résultats que nous avons aboutis et nous avons proposé des idées qui peuvent servir à l'amélioration de ce travail.

Chapitre 1

Rappelle sur les bases de la robotique et modélisation du bras de robot.

1 Rappelle sur les bases de la robotique et modélisation du bras de robot.

1.1 Introduction :

Dans ce chapitre on cite les bases de la robotique et on aborde la modélisation du bras de robot et la génération de ses modèles géométriques directe et inverse.

1.2 Rappels et Généralités :

1.2.1 Les manipulateurs en série :

Sont des robots composés d'un assemblage de bras reliés par des articulations (une chaîne), et ces types de robots sont les plus courants dans l'industrie. On les appelle souvent des bras robotiques à cause de leurs formes qui rassemble à un bras.

Figure 1: des bras robotiques effectuant des différentes tâches.

1.2.2 Le nombre de degrés de liberté :

(DDL) ou Degrees of Freedom (DOF), d'un corps rigide ou d'un système mécanique est le nombre de paramètres ou de coordonnées indépendants qui définissent entièrement sa configuration dans l'espace [2] et [3].

1.2.3 Un effecteur ou un organe terminal :

Est le dispositif situé à l'extrémité d'un bras robotique, conçu pour interagir avec l'environnement et effectuer des tâches spécifiques. La nature exacte de cet appareil dépend de l'application du robot. L'effecteur terminal pourrait être

une pince, une torche de soudage, un électroaimant ou un autre appareil [2] et [3].

1.2.4 L'espace de travail :

d'un bras robotique est défini comme l'ensemble des points qui peuvent être atteints par son effecteur terminal, il s'agit de l'espace 3D dans lequel fonctionne le robot [1] et [2].

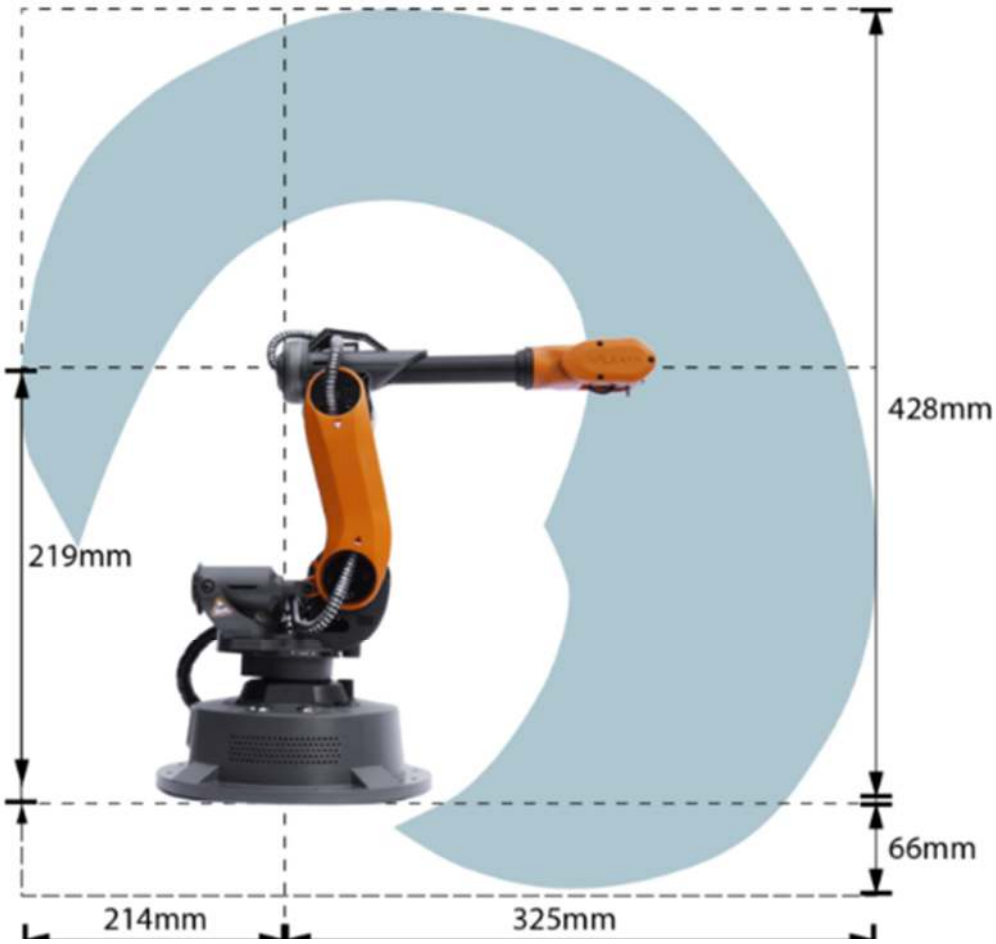


Figure 2: l'espace de travail d'un bras robotique industriel à 6ddl.

L'étude de la robotique est étroitement liée avec la position des objets dans l'espace 3D à tous moments. Ces objets peuvent être les chaines qui forment les robots ou les outils que ces robots manipulent dans leurs environnements. Ces objets peuvent être décrits par deux attributs : la position et l'orientation.

Afin de décrire la position et l'orientation d'un objet dans l'espace, nous allons attacher un repère à cet objet. Nous allons ensuite décrire la position et l'orientation de ce repère par rapport à un repère de référence généralement lié à une base fixe [2].

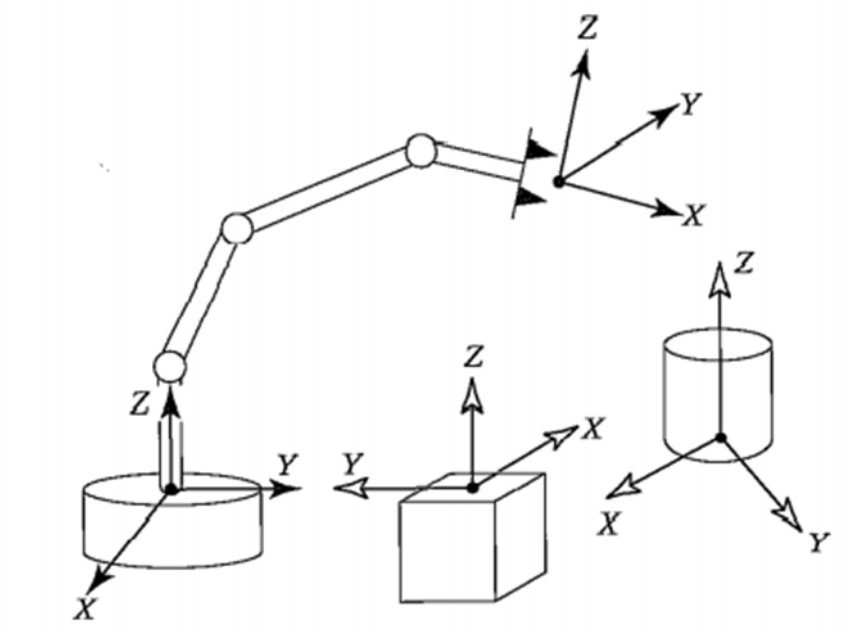


Figure 3: des repères liés aux manipulateur et différents objets dans son environnement [2].

1.2.5 La cinématique :

La cinématique est la science du mouvement qui traite le mouvement sans considération des forces qui le causent. Cette science étudie la position, la vitesse, l'accélération, et toutes les dérivées d'ordre supérieur des variables de position.

Les manipulateurs se composent de liens (des chaines) rigides, qui sont reliés par des articulations qui permettent le mouvement relatif des liens voisins. Ces articulations sont généralement instrumentées avec des capteurs de position, qui permettent à la position relative des liaisons voisines d'être mesuré. Il existe deux types d'articulations : rotoides et prismatiques.

Pour les articulations rotoides, les mouvements sont des rotations (angles), et dans le cas des articulations prismatiques, les mouvements sont des translations (déplacements linéaires) [1], [2] et [5].

1.2.6 Transformations homogènes :

Dans le cas où un repère de référence est à la fois en rotation et en translation par rapport à un autre repère de référence, une matrice de transformation homogène est utilisée pour décrire cette transformation [1].

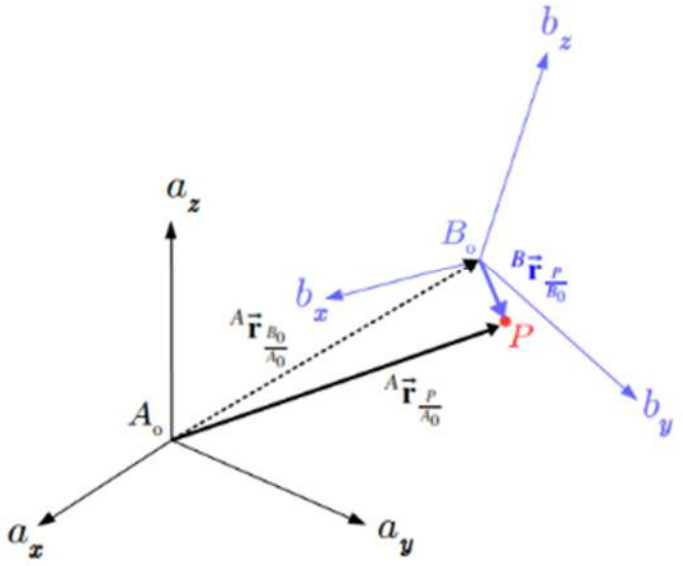


Figure 4: Translation et rotation d'un repère B par rapport à un repère A [1].

Dans la figure précédente, le point P est exprimé par rapport au repère B et l'objectif est de l'exprimer par rapport au repère A. Pour ce faire, il faudrait projeter le repère B sur le repère A, tout d'abord en le tournant pour qu'il ait la même orientation que le repère A, ensuite en le translatant pour que les deux centres A_0 et B_0 soient alignées [1].

La relation matricielle suivante montre la relation entre les trois vecteurs représentés dans la figure précédente :

$$\begin{bmatrix} r_{Ax} \\ r_{Ay} \\ r_{Az} \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{BAx} \\ r_{21} & r_{22} & r_{23} & r_{BAy} \\ r_{31} & r_{32} & r_{33} & r_{BAz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{\boldsymbol{B}x} \\ r_{By} \\ r_{Bz} \\ 1 \end{bmatrix}$$

Ou sous une forme plus compacte :

$$\begin{bmatrix} \vec{r}_p^{\,A} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^A_B R & \vec{r}^{\,A}_{B0\to A0} \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{r}_p^{\,B} \\ 1 \end{bmatrix}$$

Avec :

$\vec{r}_p^{\,A}$: le vecteur de position du point p dans le repère A.

$\vec{r}_p^{\,B}$: le vecteur de position du point p dans le repère B.

${}^A_B R$: la matrice de rotation du repère B vers le repère A.

$\vec{r}^{\,A}_{B0\to A0}$: le vecteur de translation de l'origine B_0 vers l'origine A_0.

1.2.7 Le modèle géométrique direct :

Un problème très basique dans l'étude des manipulateurs robotiques est le modèle géométrique direct (forward kinematics). Il s'agit du calcul de la position et l'orientation de l'effecteur terminal du robot [1] et [2].

Plus précisément, étant donné un ensemble d'angles d'articulation, le but du modèle géométrique direct est de calculer la position et l'orientation du repère de l'outil (organe terminal) par rapport au repère de la base fixe (origine) [1] et [2].

Ceci peut être réalisé par une composition de transformations homogènes qui transforment le repère de base au repère de l'organe terminal, en prenant comme entrée les angles des articulations. Les coordonnées de l'effecteur peuvent ensuite être extraites de la matrice de transformation résultante [1] et [2].

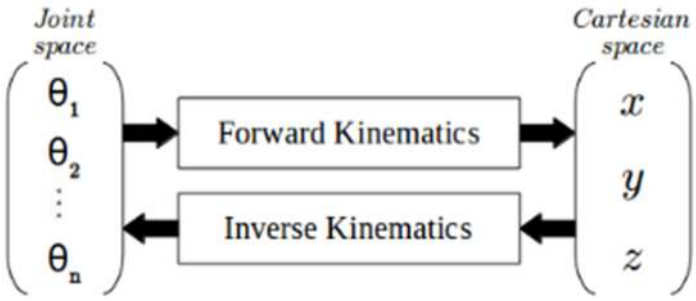


Figure 5: Relation entre le modèle géométrique direct et inverse [1].

1.2.8 Le modèle géométrique inverse :

Le modèle géométrique inverse est le processus inverse dans lequel la position de l'organe terminal est connue et les angles d'articulation qui entraîneraient cette position doivent être déterminés [1], [2] et [3].

Ce problème n'est pas aussi simple que celui de la détermination du modèle géométrique directe car les équations cinématiques ne sont pas linéaires, donc leurs solutions ne sont pas toujours faciles à déterminer et peuvent être parfois inexistantes. Il se pose aussi le problème de l'unicité des solutions.

L'absence de solutions signifie que le manipulateur ne peut pas atteindre la position et l'orientation souhaitées car elle se trouve en dehors de son espace de travail.

1.2.9 Paramètres Denavit–Hartenberg :

Avant que les transformations homogènes entre les liaisons adjacentes puissent être calculées, les repères de coordonnées des liaisons sur lesquelles les transformations sont appliquées doivent être définis. Les paramètres Denavit–Hartenberg (DH) sont quatre paramètres décrivant les rotations et les translations entre les liaisons adjacentes. La définition de ces paramètres constitue une convention d'affectation de référentiels de

coordonnées aux liens d'un robot manipulateur. La figure suivante montre la convention dite modifiée (ou selon Khalil) des paramètres DH [1], [2] et [5].

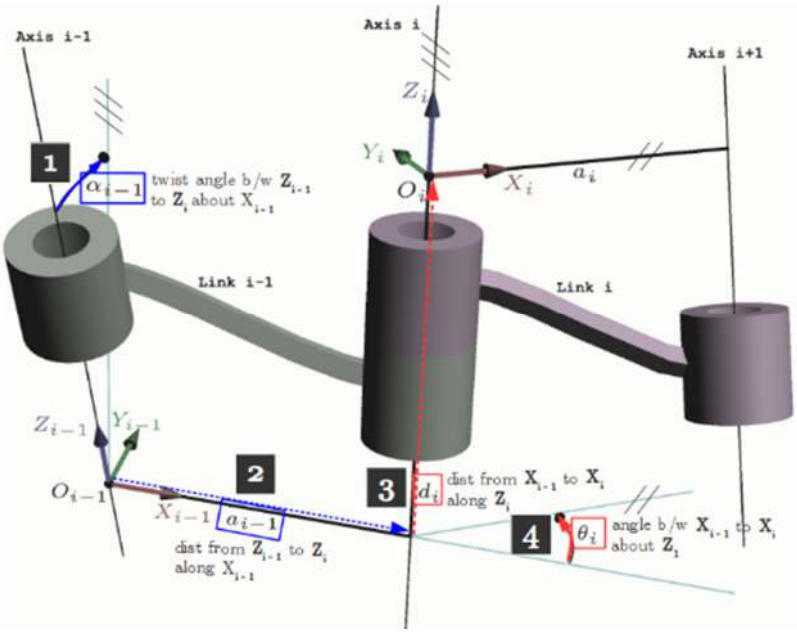


Figure 6: les paramètres de Denavit-Hartenberg.

Les paramètres sont définis comme suit [1], [2] et [5]:

α_{i-1}: angle de twist entre les axes z des liens i-1 et i (mesuré autour de x_{i-1} selon la règle de la main droite).

d_{i-1}: distance de translation entre les axes z des liaisons i-1 et i (mesurée selon x_{i-1}).

r_i: distance de translation entre les axe x-des liens i-1 et i (mesurée selon z_i).

θ_i: angle de rotation entre les axes x des liens i-1 et i (mesuré autour de zi selon la règle de la main droite).

- L'origine d'un repère i est définie par l'intersection de x_i et z_i.
- Les axes x définissent les normales communes entre zi-1 et z_i.

Pour calculer la position de l'effecteur terminal par rapport à un repère de référence de base, les transformations entre les liens adjacents sont composées comme suit [1]:

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T \ldots {}^{N-1}_{N}T $$

Où le repère de base est noté 0 et le repère de l'effecteur terminale est noté N. Ainsi, ${}^{0}_{N}T$ définit la transformation homogène qui projette le repère N sur le

repère 0. Plus précisément, une seule transformation entre les liens i-1 et i est donnée par [5] :

$$^{i-1}_{i}T = Rx_{(\boldsymbol{\alpha j})}Dx_{(\mathbf{dj})}Rz_{(\boldsymbol{\theta j})}Dz_{(\mathbf{rj})}$$

$$^{j-1}_{j}T = \begin{bmatrix} C\mathbf{j} & -Sj & 0 & dj \\ C(\alpha j)Sj & C(\alpha j)Cj & -S(\alpha j) & -rjS(\alpha j) \\ S(\alpha j)Sj & S(\alpha j)Cj & C(\alpha j) & rjC(\alpha j) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Cette transformation est composée de deux rotations R de grandeurs α et θ, et de deux déplacements D de grandeurs d et r.

Le processus d'affectation des paramètres pour les chaînes cinématiques ouvertes à n degrés de liberté (les articulations) est résumé comme suit [1], [2] et [5] :

- Attribuer toutes les articulations : {1, 2..., n}.
- Attribuer tous les repères : {0,1, 2..., n}, avec le repère de base étant donné le numéro 0.
- Tracer des lignes à travers toutes les articulations, définissant les axes des articulations.
- Attribuer l'axe Z de chaque repère pour qu'il pointe le long de son axe d'articulation.
- Identifier la normale commune entre chaque deux axes Z_{i-1} et Z_i.
- Pour i de 1 à n-1, attribuer l'axe x_i à :
    - Pour les axes obliques, le long de la normale entre Z_i et Z_{i+1} et pointant de i à i+1.
    - Pour les axes en intercession, normal au plan contenant Z_i et Z_{i+1}.
    - Pour les axes parallèles ou coïncidents, l'affectation est arbitraire.
- Pour le repère de base, il faut toujours choisir le repère {0} pour qu'il coïncide avec le repère {1} lorsque la première variable d'articulation (θ_1 ou d_1) est égale à zéro.
- Pour le repère de l'effecteur terminal, si l'articulation n est rotoide, choisir x_n dans la direction de x_{n-1} lorsque $\theta_n = 0$ et l'origine du repère {n} tel que $d_n = 0$.

1.3 Application sur le bras robotique BCN-3D-Moveo :

Le bras robotique Moveo a 5 degrés de liberté, La figure suivante montre la configuration des axes pour le modèle géométrique direct selon la convention de DH.

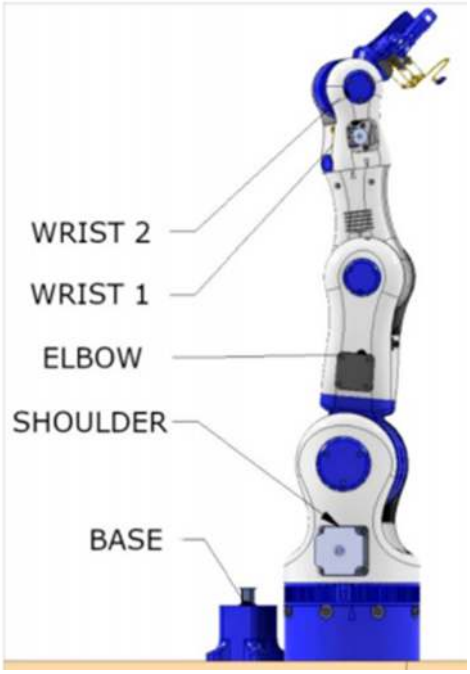


Figure 7: le bras robotique BCN-3D-Moveo.

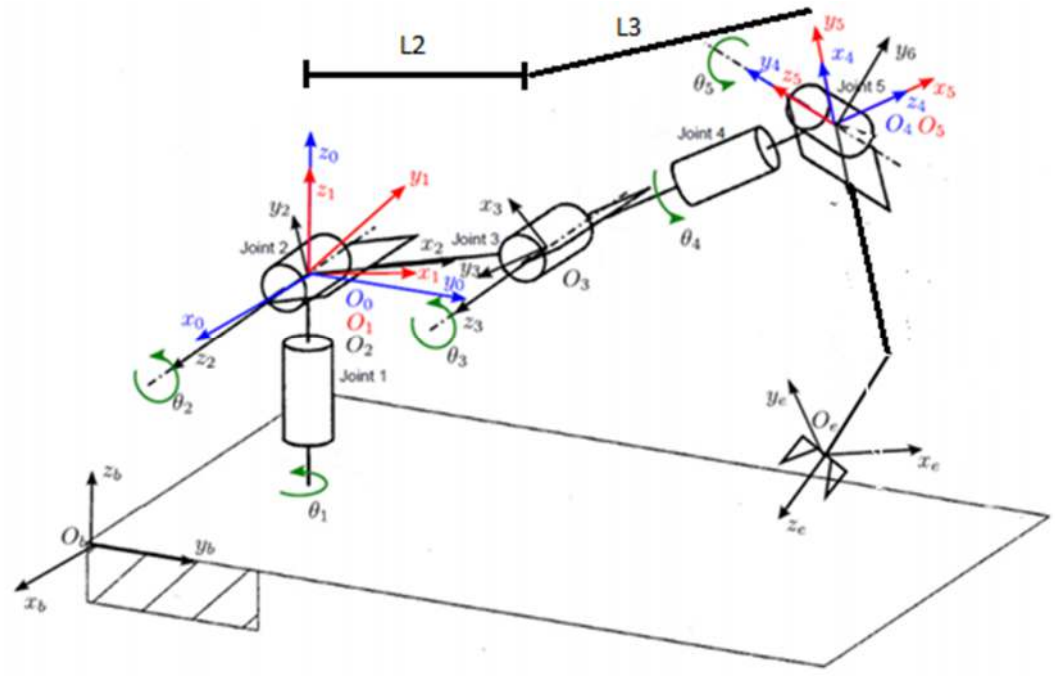


Figure 8: Représentation selon la convention DH [6].

1.3.1 Matrice Denavit–Hartenberg :

Chaque transformation ${}^{i-1}_{i}T$ est représenté comme un produit de quatre transformations :

$$ {}^{i-1}_{i}T = Rx_{(\boldsymbol{\alpha j})} Dx_{(\mathbf{dj})} Rz_{(\boldsymbol{\theta j})} Dz_{(\mathbf{rj})} $$

La table de Denavit-Hartenberg modifiée est donnée par :

j	**αj**	**dj**	**θj**	**rj**
1	**0**	**0**	**Θ1**	**0**
2	$\frac{\pi}{2}$	**0**	**Θ2**	**0**
3	**0**	**L2 = 221.5**	**Θ3**	**0**
4	$\frac{\pi}{2}$	**0**	**Θ4**	**L3 = 224.5**
5	$-\frac{\pi}{2}$	**0**	**Θ5**	**0**
Organe Terminal	**0**	**LT = 98**	**0**	**0**

Tableau 1 : Matrice des paramètres DH pour le manipulateur BCN-3D Moveo.

1.3.2 Modèle géométrique direct :

En utilisant les valeurs du tableau précédent de la matrice DH, on obtient les matrices suivantes de transformation homogène pour chaque axe de coordonnées :

$$ {}^{0}_{1}T = \begin{bmatrix} C\mathbf{1} & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}T = \begin{bmatrix} C\mathbf{2} & -S2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S2 & C2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}T = \begin{bmatrix} C\mathbf{3} & -S3 & 0 & L2 \\ S3 & C3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} C\mathbf{4} & -S4 & 0 & 0 \\ 0 & 0 & -1 & -L3 \\ S4 & C4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} C5 & -S5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S5 & -C5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Le modèle géométrique directe du robot s'exprime par le produit des matrices de transformation homogène dans l'ordre suivant :

$$ {}^{0}_{5}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T $$

Où, le résultat du produit précédent est une matrice unique qui regroupe chacune des rotations et translations des articulations, de sorte que l'effecteur terminal se trouve à une position spécifique dans l'espace par rapport à la base. Le calcul de ce produit nous donne le résultat suivant :

$$ {}^{0}_{5}T = T = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Avec :

$$ T_{11} = C5\big(S1\,S4 - C4\,(C1\,S2\,S3 - C1\,C2\,C3)\big) - S5(C1\,C2\,S3 + C1\,C3\,S2) $$

$$ T_{12} = -S5\,\big(S1\,S4 - C4\,(C1\,S2\,S3 - C1\,C2\,C3)\big) - C5(C1\,C2\,S3 + C1\,C3\,S2) $$

$$ T_{13} = C4\,S1 + S4\,(C1\,S2\,S3 - C1\,C2\,C3) $$

$$ T_{14} = L3\,(C1\,C2\,S3 + C1\,C3\,S2) + L2\,C1C2 $$

$$ T_{21} = -C5\,\big(C1\,S4 + C4\,(S1\,S2\,S3 - C2\,C3\,S1)\big) - S5(C2\,S1\,S3 + C3\,S1\,S2) $$

$$ T_{22} = S5\,\big(C1\,S4 + C4\,(S1\,S2\,S3 - C2\,C3\,S1)\big) - C5(C2\,S1\,S3 + C3\,S1\,S2) $$

$$ T_{23} = S4\,(S1\,S2\,S3 - C2\,C3\,S1) - C1C4 $$

$$ T_{24} = L3\,(C2\,S1\,S3 + C3\,S1\,S2) + L2\,C2S1 $$

$$ T_{31} = S5\,(C2\,C3 - S2\,S3) + C4\,C5(C2\,S3 + C3\,S2) $$

$$ T_{32} = C5\,(C2\,C3 - S2\,S3) - C4\,S5(C2\,S3 + C3\,S2) $$

$$ T_{33} = -S4\,(C2\,S3 + C3\,S2) $$

$$ T_{34} = L2\,S2 - L3\,(C2\,C3 - S2\,S3) $$

1.3.3 Modèle géométrique inverse :

Lorsque on choisit de générer des mouvements dans l'espace opérationnel d'une position $\mathbf{X_1}$ (X_1, Y_1, Z_1, Rx_1, Ry_1, Rz_1) à une position $\mathbf{X_2}$ (X_2, Y_2, Z_2, Rx_2, Ry_2, Rz_2), la commande du robot requière de connaitre les coordonnées

articulaires **q** correspondants à ces positions. Il y a donc une nécessité d'inverser le modèle géométrique direct.

Pour trouver le modèle géométrique du bras Moveo, on va implémenter la méthode dite de Paul proposée par Richard P. Paul en 1980 [7] :

L'inversion du modèle géométrique se fera successivement suivant ces différentes équations :

$$U_0 = {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_pT$$

$${}^1_0TU_0 = {}^1_0T\,{}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_pT = {}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_pT$$

.

.

.

$${}^5_4T\,{}^4_3T\,{}^3_2T\,{}^2_1T\,{}^1_0TU_0 = {}^5_pT$$

La première itération revient à résoudre cette équation :

$${}^1_0TU_0 = {}^1_0T\,{}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_pT = {}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_pT$$

Elle devrait nous permettre d'exprimer q_1 en fonction des termes de U_0 et des paramètres du robot.

$${}^0_1T = \begin{bmatrix} C\mathbf{1} & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Longrightarrow {}^1_0T = \begin{bmatrix} C\mathbf{1} & S1 & 0 & 0 \\ -S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matrice U_0 est de la forme suivante :

$$\mathrm{U0} = \begin{bmatrix} Sx & Nx & Ax & Px \\ Sy & Ny & Ay & Py \\ Sz & Nz & Az & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Le deuxième terme de l'équation correspond à :

$${}^1_pT = \begin{bmatrix} C_{23}C_4C_5 + S_{23}S_5 & C_{23}S_4 & C_{23}C_4S_5 - S_{23}C_5 & C_{23}C_4S_5r_6 - S_{23}(C_5r_6 - r_4) + C_2d_3 \\ -S_4C_5 & C_4 & -S_4S_5 & -S_4S_5r_6 \\ S_{23}C_4C_5 - C_{23}S_5 & S_{23}S_4 & S_{23}C_4S_5 + C_{23}C_5 & S_{23}C_4S_5r_6 + C_{23}(C_5r_6 - r_4) + S_2d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Nous pouvons donc réécrire cette équation :

$$\begin{bmatrix} C\mathbf{1} & S1 & 0 & 0 \\ -S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx & Nx & Ax & Px \\ Sy & Ny & Ay & Py \\ Sz & Nz & Az & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_{23}C_4C_5 + S_{23}S_5 & C_{23}S_4 & C_{23}C_4S_5 - S_{23}C_5 & C_{23}C_4S_5r_6 - S_{23}(C_5r_6 - r_4) + C_2d_3 \\ -S_4C_5 & C_4 & -S_4S_5 & -S_4S_5r_6 \\ S_{23}C_4C_5 - C_{23}S_5 & S_{23}S_4 & S_{23}C_4S_5 + C_{23}C_5 & S_{23}C_4S_5r_6 + C_{23}(C_5r_6 - r_4) + S_2d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En simplifiant le premier terme, on obtient l'équation suivante :

$$\begin{bmatrix} C_1Sx + S_1Sy & C_1Nx + S_1Ny & C_1Ax + S_1Ay & C_1Px + S_1Py \\ -S_1Sx + C_1Sy & -S_1Nx + C_1Ny & -S_1Ax + C_1Ay & -S_1Px + C_1Py \\ Sz & Nz & Az & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} C_{23}C_4C_5 + S_{23}S_5 & C_{23}S_4 & C_{23}C_4S_5 - S_{23}C_5 & C_{23}C_4S_5r_6 - S_{23}(C_5r_6 - r_4) + C_2d_3 \\ -S_4C_5 & C_4 & -S_4S_5 & -S_4S_5r_6 \\ S_{23}C_4C_5 - C_{23}S_5 & S_{23}S_4 & S_{23}C_4S_5 + C_{23}C_5 & S_{23}C_4S_5r_6 + C_{23}(C_5r_6 - r_4) + S_2d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Par identification, on peut extraire le système suivant :

$$\begin{cases} -S_1Ax + C_1Ay = -S_4S_5r_6 \\ -S_1Px + C_1Py = -S_4S_5r_6 \end{cases} \rightarrow \{-S_1Px + C_1Py = (-S_1Ax + C_1Ay)r_6\}$$

$$S_1(Axr_6 - Px) + C_1(Py - Ayr_6) = 0$$

⇨ Deux solutions possibles $\begin{cases} \theta 1 = Atan2(-Py + Ayr_6, Axr_6 - Px) \\ ou \\ \theta 1' = \theta 1 + 180° \end{cases}$

Lors de la deuxième itération, on va résoudre cette équation :

$${}^2_1T\,{}^1_0TU_0 = {}^2_pT$$

Posons ${}^1_0TU_0 = \begin{bmatrix} A & B & C & G \\ D & E & F & H \\ Sz & Nz & Az & I \\ 0 & 0 & 0 & 1 \end{bmatrix} =$

$$\begin{bmatrix} C_1Sx + S_1Sy & C_1Nx + S_1Ny & C_1Ax + S_1Ay & C_1Px + S_1Py \\ -S_1Sx + C_1Sy & -S_1Nx + C_1Ny & -S_1Ax + C_1Ay & -S_1Px + C_1Py \\ Sz & Nz & Az & Pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C\mathbf{2} & 0 & S2 & 0 \\ -S2 & 0 & C2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A & B & C & G \\ D & E & F & H \\ Sz & Nz & Az & I \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} C_3C_4C_5 + S_3S_5 & C_3S_4 & C_3C_4S_5 - S_3C_5 & C_3C_4S_5r_6 - S_3(C_5r_6 - r_4) + d_3 \\ S_3C_4C_5 - C_3S_5 & S_3S_4 & S_3C_4S_5 + C_3C_5 & S_3C_4S_5r_6 + C_3(C_5r_6 - r_4) \\ S_4C_5 & -C_4 & S_4S_5 & S_4S_5r_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C_2A + S_2Sz & C_2B + S_2Nz & C_2C + S_2Az & C_2G + S_2I \\ -S_2A + C_2Sz & -S_2B + C_2Nz & -S_2C + C_2Az & -S_2G + C_2I \\ -D & -E & -F & -H \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} C_3C_4C_5 + S_3S_5 & C_3S_4 & C_3C_4S_5 - S_3C_5 & C_3C_4S_5r_6 - S_3(C_5r_6 - r_4) + d_3 \\ S_3C_4C_5 - C_3S_5 & S_3S_4 & S_3C_4S_5 + C_3C_5 & S_3C_4S_5r_6 + C_3(C_5r_6 - r_4) \\ S_4C_5 & -C_4 & S_4S_5 & S_4S_5r_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On peut extraire deux systèmes :

$$\begin{cases} C_2C + S_2Az = C_3C_4S_5 - S_3C_5 \\ C_2G + S_2I = C_3C_4S_5r_6 - S_3(C_5r_6 - r_4) + d_3 \\ -S_2C + C_2Az = S_3C_4S_5 + C_3C_5 \\ -S_2G + C_2I = S_3C_4S_5r_6 + C_3(C_5r_6 - r_4) \end{cases}$$

➔

$$\begin{cases} C_2G + S_2I = (C_2C + S_2Az)r_6 + S_3r_4 + d_3 \\ -S_2G + C_2I = (-S_2C + C_2Az)r_6 - C_3r_4 \end{cases}$$

➔

$$\begin{cases} S_3r_4 = C_2(G - Cr_6) + S_2(I - Azr_6) - d_3 \\ C_3r_4 = S_2(G - Cr_6) + C_2(Azr_6 - I) \end{cases}$$

$\theta 2$ est résolu par le système suivant : $B_1 \sin(q_2) + B_2 \cos(q_2) = B_3$

$$avec \begin{cases} B_1 = -2(I - Azr_6)d_3 \\ B_2 = -2(G - Cr_6)d_3 \\ B_3 = r_4^2 - (G - Cr_6)^2 - (I - Azr_6)^2 - {d_3}^2 \end{cases}$$

➔ $\theta 2 = Atan2(\sin(q_2), \cos(q_2))$

$$avec \begin{cases} \sin(q_2) = \dfrac{B_1B_3 + \varepsilon B_2\sqrt{B_1^2 + B_2^2 - {B_3}^2}}{B_1^2 + B_2^2} \\ \cos(q_2) = \dfrac{B_2B_3 - \varepsilon B_1\sqrt{B_1^2 + B_2^2 - B_3^2}}{B_1^2 + B_2^2} \\ et\ \varepsilon = \pm 1 \end{cases}$$

Puis $\theta 3$ est résolu par un système de la forme suivante :

$$\begin{cases} S_3r_4 = C_2(G - Cr_6) + S_2(I - Azr_6) - d_3 \\ C_3r_4 = S_2(G - Cr_6) + C_2(Azr_6 - I) \end{cases}$$

➔ $\theta 3 = Atan2\left(\frac{C_2(G - Cr_6) + S_2(I - Azr_6) - d_3}{r_4}, \frac{S_2(G - Cr_6) + C_2(Azr_6 - I)}{r_4}\right)$ avec r_4 non nul.

La troisième itération consiste à résoudre cette équation :

$${}^3_2T\,{}^2_1T\,{}^1_0T\,U_0 = {}^3_pT$$

Posons ${}^2_1T\,{}^1_0T\,U_0 = \begin{bmatrix} M & N & O & P \\ Q & R & S & T \\ -D & -E & -F & -H \\ 0 & 0 & 0 & 1 \end{bmatrix} =$

$$\begin{bmatrix} C_2A + S_2Sz & C_2B + S_2Nz & C_2C + S_2Az & C_2G + S_2I \\ -S_2A + C_2Sz & -S_2B + C_2Nz & -S_2C + C_2Az & -S_2G + C_2I \\ -D & -E & -F & -H \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{C3} & S3 & 0 & -C3d3 \\ -S3 & C3 & 0 & S3d3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} M & N & O & P \\ Q & R & S & T \\ -D & -E & -F & -H \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_4C_5 & S_4 & C_4S_5 & C_4S_5r_6 \\ -S_5 & 0 & C_5 & S_5r_6 - r_4 \\ S_4C_5 & -C_4 & S_4S_5 & S_4S_5r_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C_3M + S_3Q & C_3N + S_3R & C_3O + S_3S & C_3P + S_3T - C_3d_3 \\ -S_3M + C_3Q & -S_3N + C_3R & -S_3O + C_3S & -S_3P + C_3T + s_3d_3 \\ -D & -E & -F & -H \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} C_4C_5 & S_4 & C_4S_5 & C_4S_5r_6 \\ -S_5 & 0 & C_5 & S_5r_6 - r_4 \\ S_4C_5 & -C_4 & S_4S_5 & S_4S_5r_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On peut extraire de ce système :

$$\begin{cases} S_4 = C_3N + S_3R \\ -C_4 = -E \\ -S_5 = -S_3M + C_3Q \\ C_5 = -S_3O + C_3S \end{cases}$$

On peut calculer $\theta 4$ à partir du système suivant :

$$\begin{cases} S_4 = C_3N + S_3R \\ -C_4 = -E \end{cases}$$

➔ $\theta 4 = Atan2(C_3N + S_3R, \mathrm{E})$

Et on peut calculer $\theta 5$ à partir du système suivant :

$$\begin{cases} -S_5 = -S_3M + C_3Q \\ C_5 = -S_3O + C_3S \end{cases}$$

➔ $\theta 5 = Atan2(S_3M - C_3Q, -S_3O + C_3S)$

Les cinq angles sont complètement calculés, donc ça sert à rien de calculer plus d'itérations.

1.4 Conclusion :

Dans ce chapitre on a cité les bases de la robotique et on a abordé la modélisation du bras de robot BCN-3D Moveo et la génération de ses modèles géométriques directe et inverse.

Chapitre 2

Dimensionnement du robot.

2 Dimensionnement du robot :

2.1 Introduction :

Dans ce chapitre nous allons détailler la construction du bras de robot dans deux parties : la partie électrique et la partie mécanique. Dans la partie électronique, nous allons détailler les spécifications des moteurs et cartes électroniques (Arduino, Ramps, driver TB6566) qu'on a utilisé. Dans la partie mécanique on détaille les parties du bras de robot , leurs formes , tailles , la matière d'impression 3D , …etc.

2.2 Partie électrique :

2.2.1 Les Moteurs :

Le bras robotique Moveo BCN3D est doté de 6 emplacements moteurs prédéfinis. C'est un bras robotique avec 5 degrés de liberté. Chaque articulation est bougée via un système de poulies-courroies qui permettent un meilleur contrôle sur la position du robot.

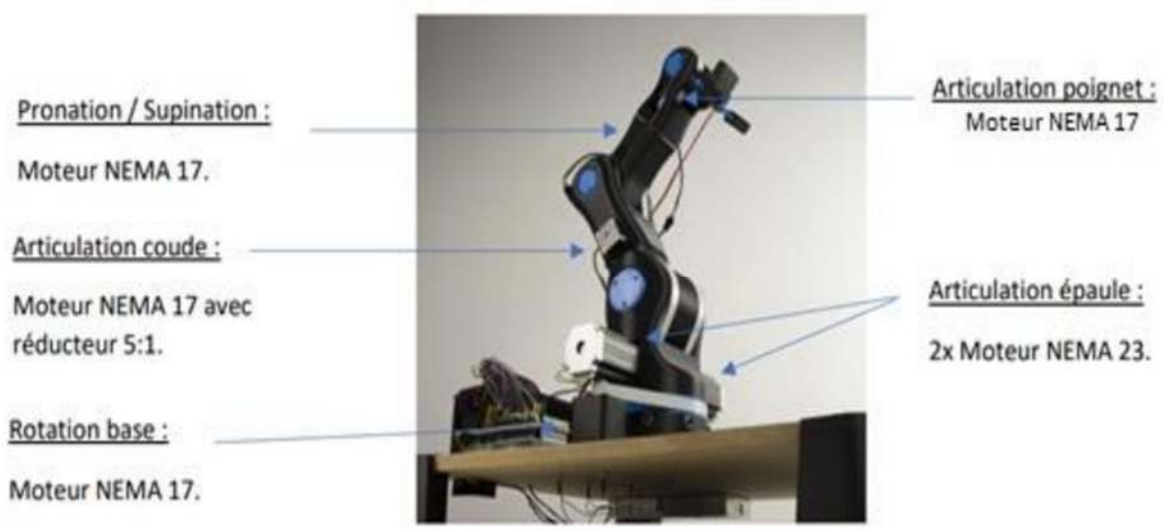


Figure 9: les types et les localisations des moteurs sur le bras robotique [8].

Voici un tableau récapitulatif sur tous les moteurs utilisés :

Moteur	Référence	Ampérage (A)	Torque (Kg.cm)
Nema 17	**SM42HT47-1684**	**1.6**	**3.6**
Nema 23	**SM57HT76-3004A**	**3**	**20.39**
Nema 17	**SM42HT33-1334A**	**1.33**	**2.2**

Nema 23	**SM57HT112-3004A**	**3**	**28**
Servo moteur	**7300-SERVO213**	**0.1**	**13**

Tableau 2 : les moteurs utilisés et leurs spécifications.

Maintenant on va détailler les caractéristiques de chaque moteur :

2.2.2 Nema 17 SM42HT47-1684 :

Figure 10: Nema 17 SM42HT47-1684.

Ce moteur pas à pas bipolaire hybride a un angle de pas de 1,8° (200 pas / tour). Chaque phase consomme 1.6 A à 2.8 V, ce qui permet un couple de maintien de 3,6 kg-cm.

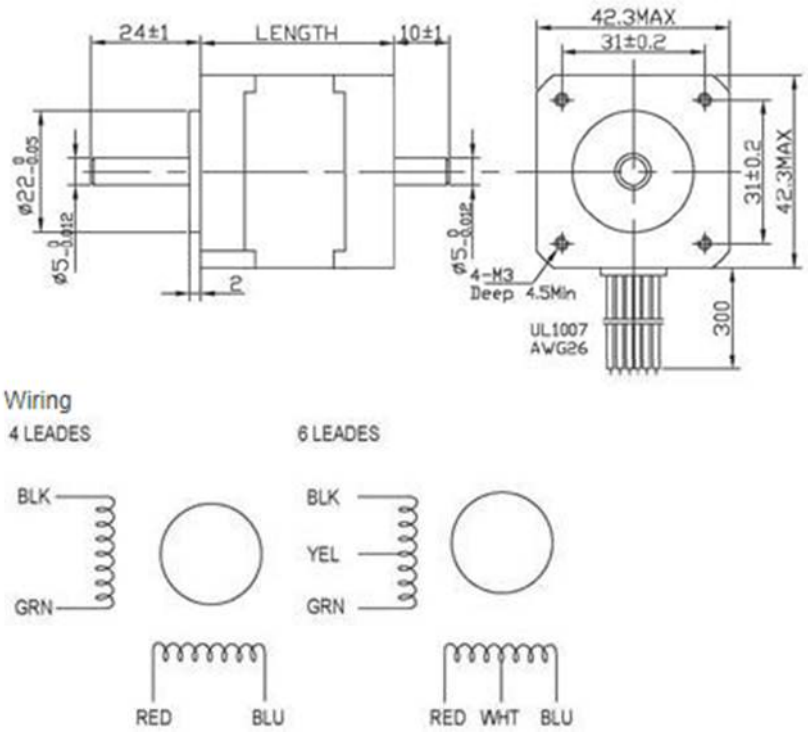


Figure 11: Dimensions et câblage du Nema 17 SM42HT47-1684.

2.2.3 Nema 23 SM57HT76-3004A :

Figure 12: Nema 23 SM57HT76-3004A.

Ce moteur nema 23 de taille 57x57x76mm avec un couple de maintien : 19,27 kg.cm (267 oz.cm - 1,89N.m) et un nombre de pas : 200 (pas angulaire de 1,8°).Il consomme un courant d'intensité 3A par phase.C'est une bonne solution pour les applications avec un espace limité mais nécessitant une faible vitesse et / ou un couple élevé et c'est le cas pour l'articulation 3 du robot.

2.2.4 Nema 17 SM42HT33-1334A :

Figure 13: Nema 17 SM42HT33-1334A.

Ce moteur pas à pas bipolaire hybride a un angle de pas de 1,8 ° (200 pas / tour). Chaque phase consomme 1,33A A à 2,8 V, ce qui permet un couple de maintien de 2,2 kg-cm (30 oz-in).

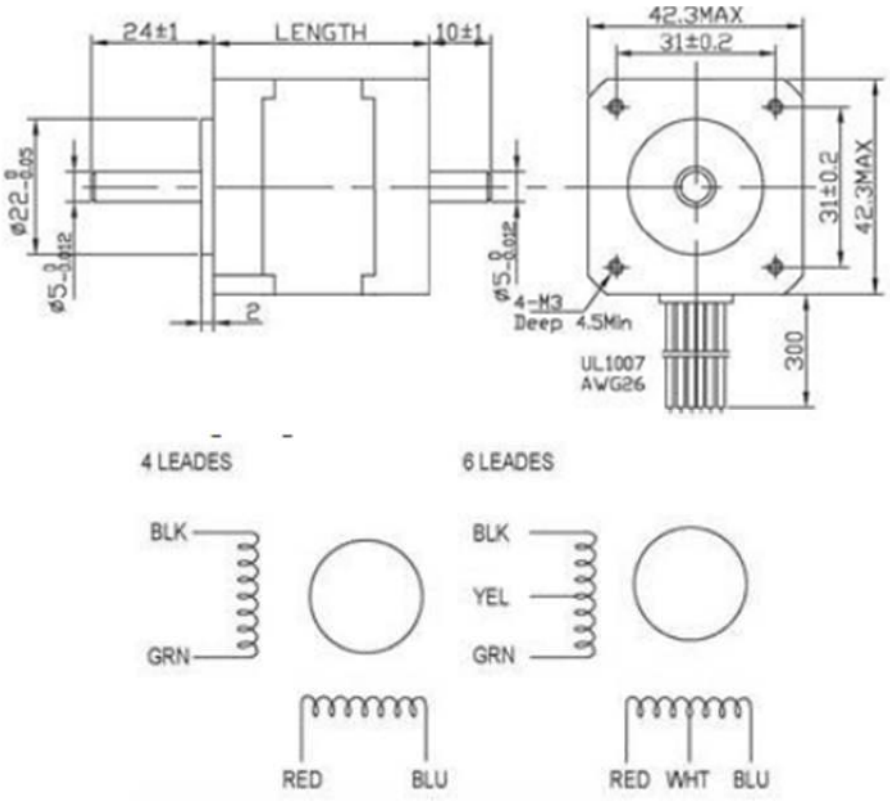


Figure 14: Dimensions et câblage du Nema 17 SM42HT33-1334A.

2.2.5 Nema 23 SM57HT112-3004A 112 mm :

Figure 15: Nema 23 SM57HT112-3004A.

Ce moteur pas à pas Nema 23 avec un corps de 112mm et un courant nominal de 3A à 4.8V ce qui permet un couple de maintien de 28 kg.cm. On a besoin de deux moteurs de ce type. Ce sont les gros moteurs qui doivent supporter presque tout le poids du bras. Il est important qu'ils aient beaucoup de couple de maintien.

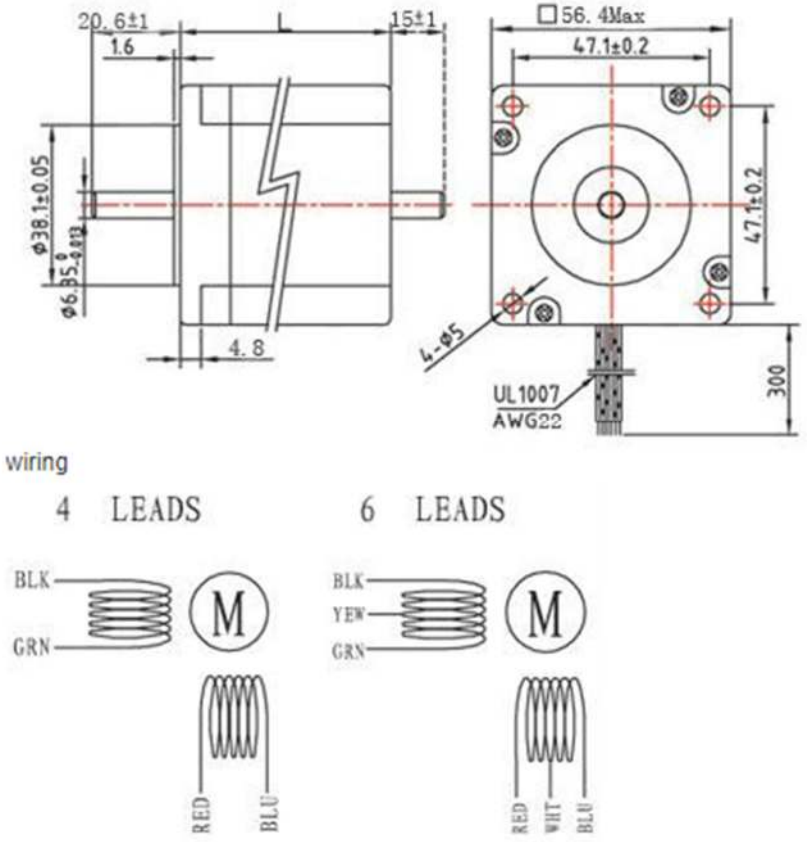


Figure 16: Dimensions et câblage du Nema 23 SM57HT112-3004A.

2.2.6 Servo moteur 55g 7300-SERVO213 :

Figure 17:Servo moteur 55g 7300-SERVO213.

Ce servomoteur est de type RC avec 180 degrés, un couple de maintien de 13 Kg.cm et un engrenage métallique. Il est alimenté par un courant de 100 mA. On a l'utilisé pour commander le pince (gripper).

2.3 Les cartes électroniques :

2.3.1 Drivers TB6560 :

Chacun de ces moteurs est contrôlé par un driver de moteur TB6560. Ce dernier sert à garder une commande de courant à injecter dans le moteur afin d'avoir un couple de maintien au niveau de l'articulation. Le robot garde donc sa position même si les moteurs ne sont pas utilisés.

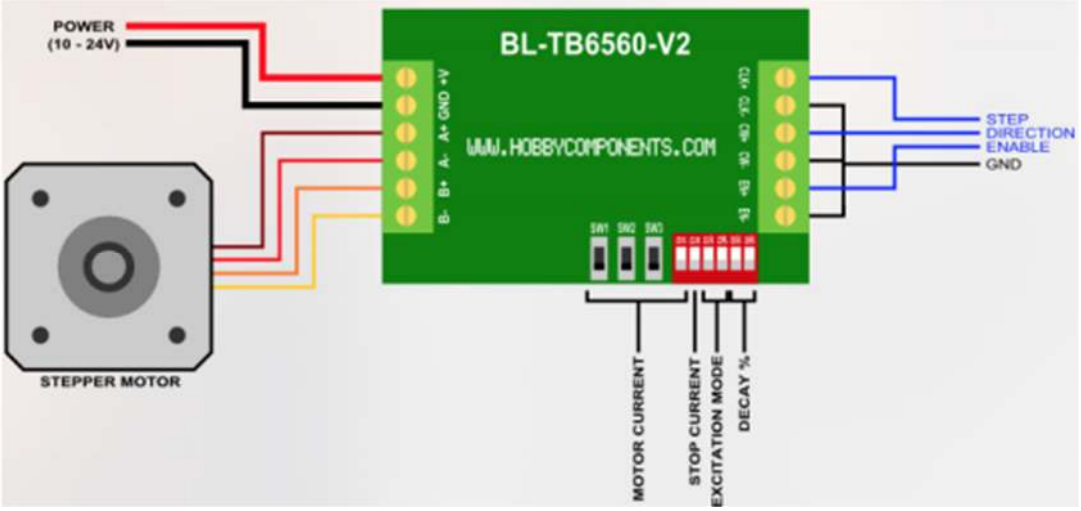


Figure 18: Driver TB6560 connecté avec un moteur pas à pas [10].

La tension d'alimentation du driver est faite via un transformateur 24V se servant du réseau. Le driver est quant à lui configuré pour des pas de 1/16', ce qui correspond à la segmentation de pas du moteur pas-à-pas.

On doit d'abord configurer les switches du driver selon le courant d'alimentation du moteur, le mode d'excitation, le courant d'arrêt et la décroissance de courant (le courant de recirculation). Pour cela on configurer 6 contacts (switches) selon les tableaux suivant :

Current Settings							
I(A)	**0.5**	**1**	**1.5**	**1.8**	**2**	**2.5**	**3**
SW1	**0**	**0**	**0**	**1**	**1**	**1**	**1**
SW2	**1**	**0**	**1**	**0**	**1**	**0**	**1**
SW3	**0**	**1**	**1**	**0**	**0**	**1**	**1**

Tableau 3 : Le courant d'alimentation du moteur.

Excitation Mode Setting				
SW4	**1**	**2**	**8**	**16**

S3/M2	**1**	**1**	**0**	**0**
S4/M1	**1**	**0**	**0**	**1**

Tableau 4 : Le mode d'excitation.

Decay Mode Setting				
SW4	**0%**	**25%**	**50%**	**100%**
S5/DY2	**1**	**1**	**0**	**0**
S6/DY1	**1**	**0**	**1**	**0**

Tableau 5 : Le courant de recirculation.

2.3.2 Le Ramps v1.4 :

Les 6 cartes (drivers) TB6560 sont quant à elle commandées grâce à une carte Ramps V1.4. Son intérêt est qu'elle permet d'interfacer différents périphériques qu'on utilise souvent, comme des moteurs pas à pas, des dispositifs de puissances ou des contacts ou des servomoteurs. La carte est alimentée en 12V et est montée sur la carte Arduino Mega 2560 [11].

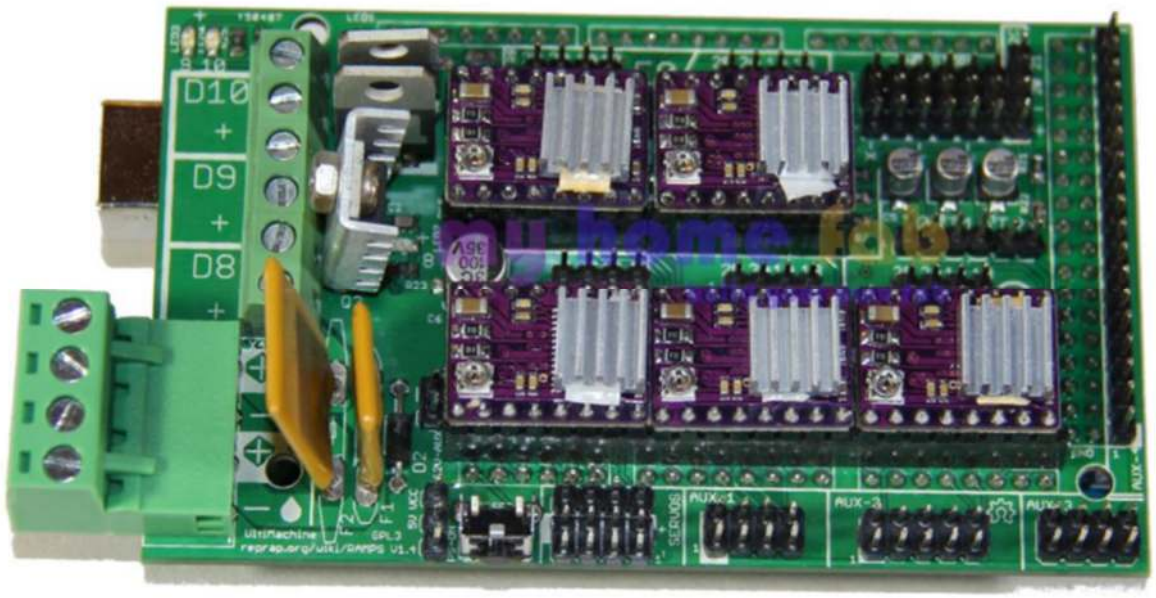


Figure 19: le Ramps v1.4.

Pour la carte Ramps V1.4, les résistances et les condensateurs sont montés en(SMD) afin de pouvoir contenir plus de composants passifs. Ceci rajoute des étapes d'assemblage supplémentaires, mais le fait que cette carte conserve des dimensions assez grandes rend la tâche beaucoup moins ardue.

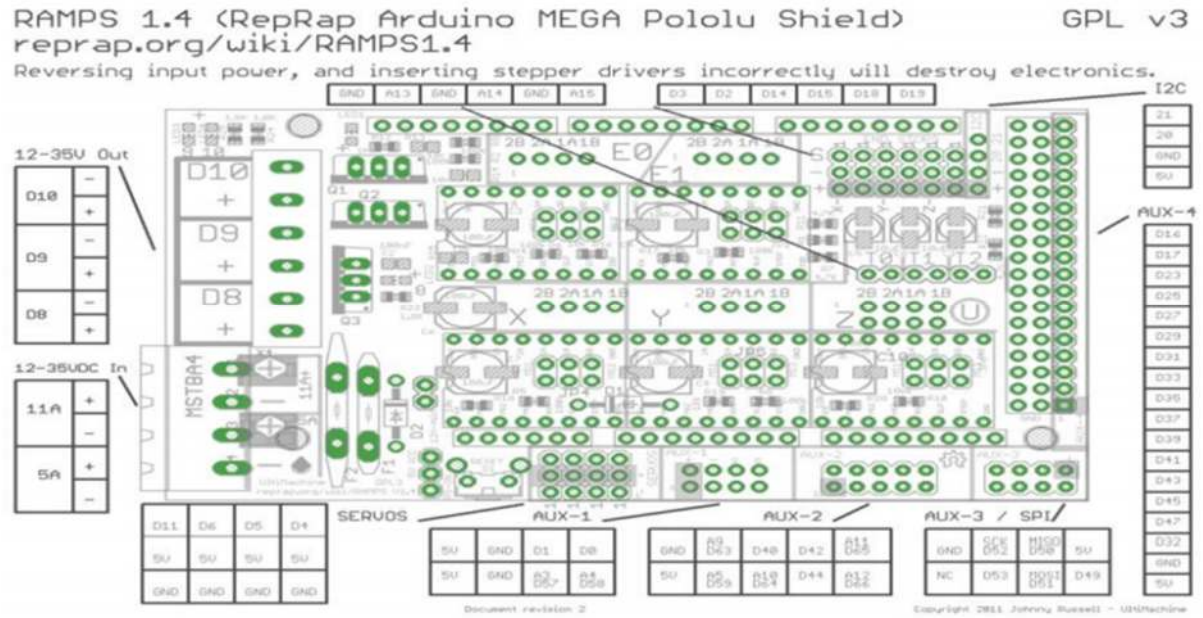


Figure 20: les composants du Ramps v1.4.

2.3.3 La carte Arduino Mega 2560 :

On utilise une carte Arduino Mega 2560 pour commander les moteurs en même temps. L'Arduino donne des ordres qui seront traduits en signal PWM par Le Ramps. Ce signal sera compréhensible par les moteurs [12].

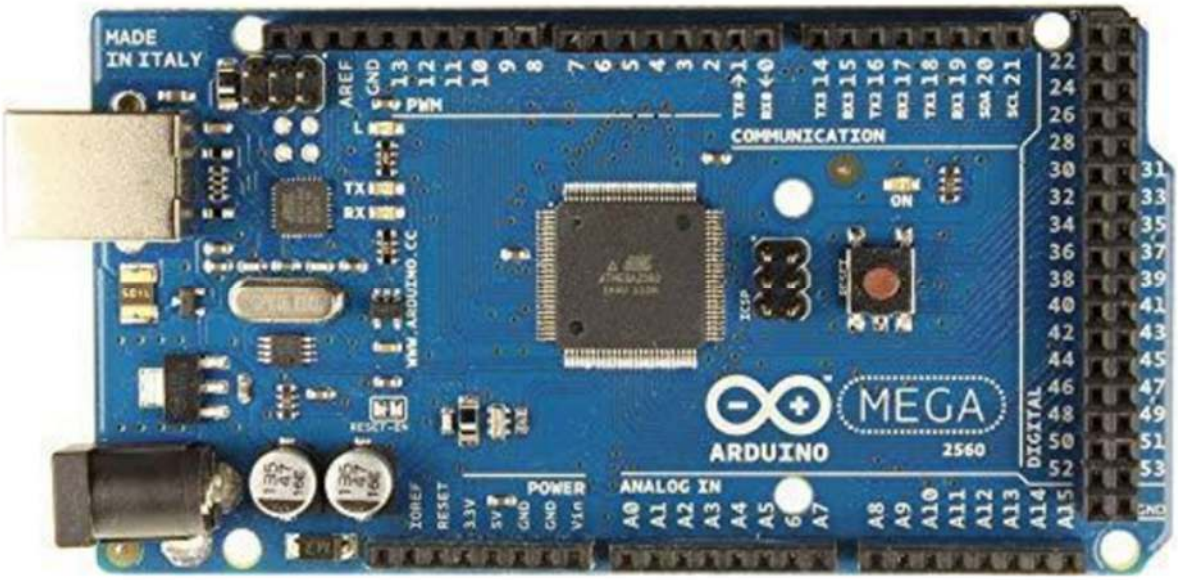


Figure 21: La carte Arduino 2560.

On peut résumer la partie électronique et toutes les liaisons entre les différents composants par le schéma suivant [13] :

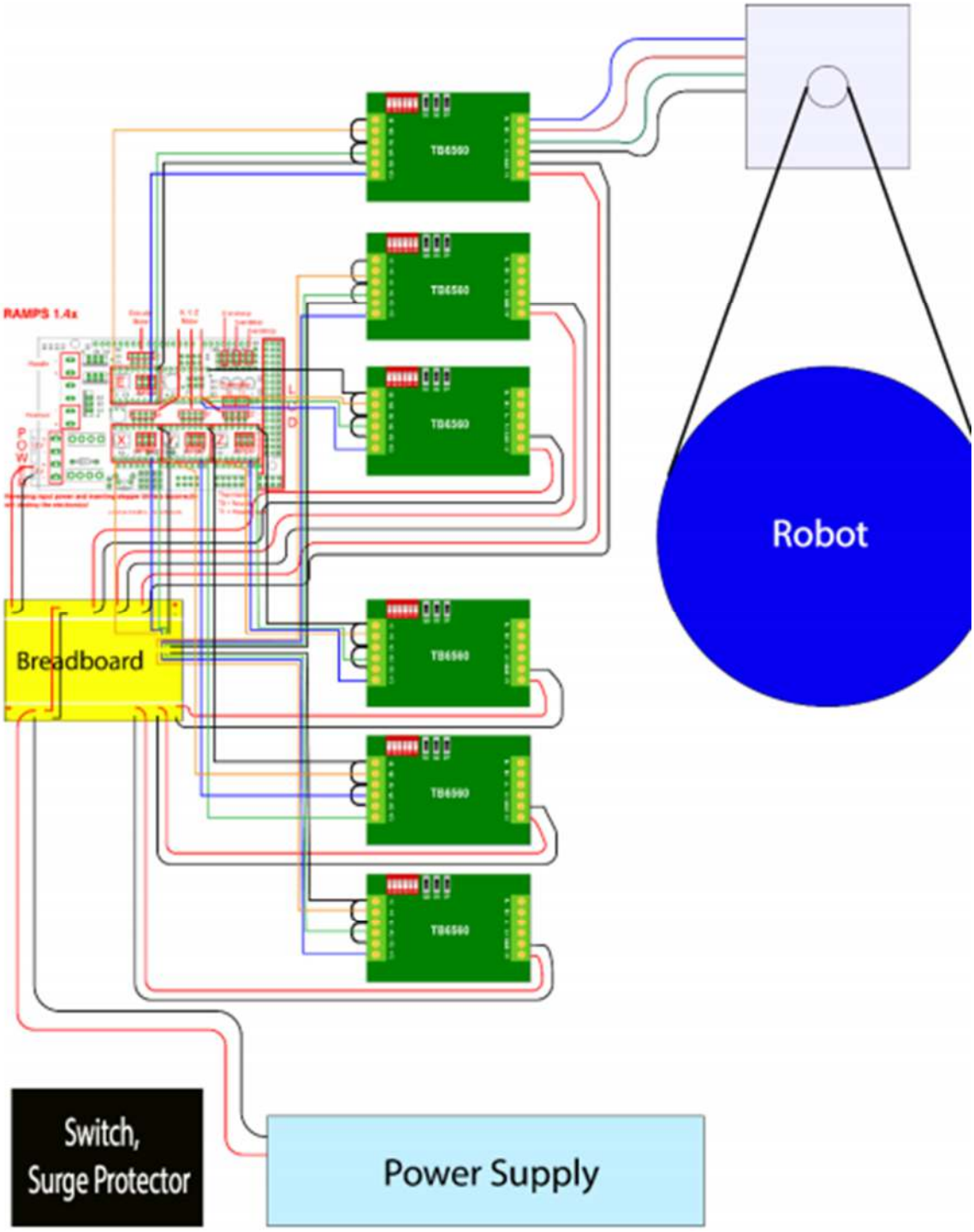


Figure 22: Schéma global de la partie électronique [13].

2.4 Partie mécanique :

2.4.1 La construction du corps du bras :

Toute la structure du bras du robot est fabriquée à partir de pièces imprimées en 3D [14]. Certaines pièces sont assez volumineuses et leur impression prendra un certain temps. La plupart des parties du bras du robot se connectent ensemble à l'aide de vis mécaniques. Le manuel du robot nous aide un peu pour la construction [15].

2.4.2 Articulation 1 : la base :

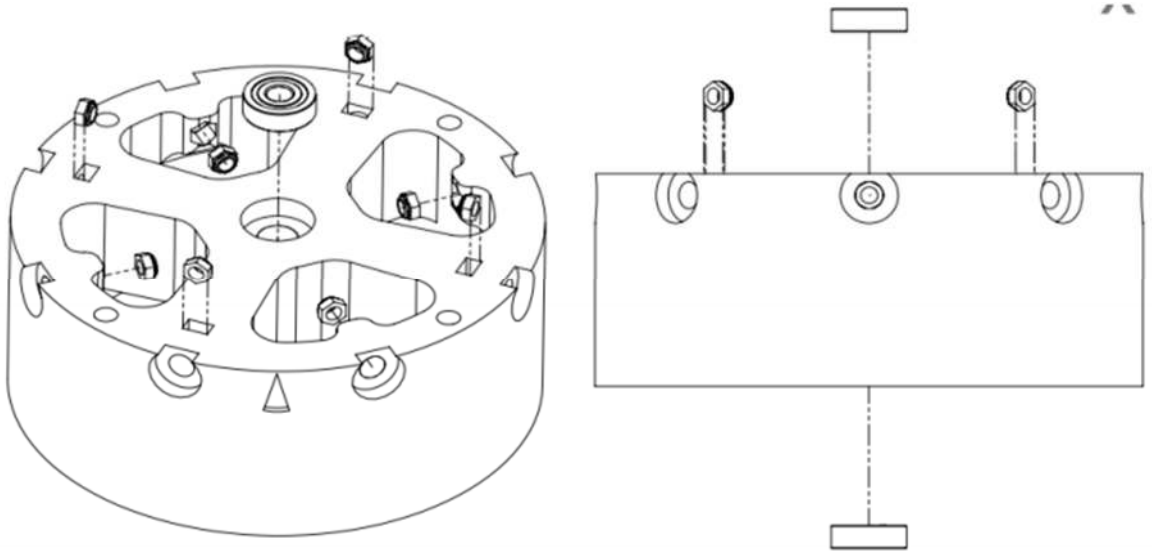


Figure 23: Articulation 1 [15].

Pour fournir un mouvement à faible friction, nous avons installé deux roulements à billes dans la base rotative. Plus tard, un arbre reliera la plaque rotative à la base rotative et ces deux roulements lui permettront de tourner librement. Les roulements à billes sont conçus pour être ajustés par pression dans la base rotative. Le bras du robot tourne au-dessus de la base rotative. Dans cette étape, nous allons attacher des roulements à billes à la base rotative qui permettent au bras de tourner en douceur. Les roulements à billes dépassent légèrement au-dessus du haut de la base rotative.

Figure 24: la base rotative [15].

La partie plaque rotative finira par se fixer de manière rigide à la base du bras du robot lui-même. Afin de former la connexion, nous installons un total de six écrous M4 dans la partie supérieure de la plaque rotative et ensuite on insère six vis M4 de longueur 45mm et un vis M8 de longueur 65mm.

Figure 25: La plaque rotative [15].

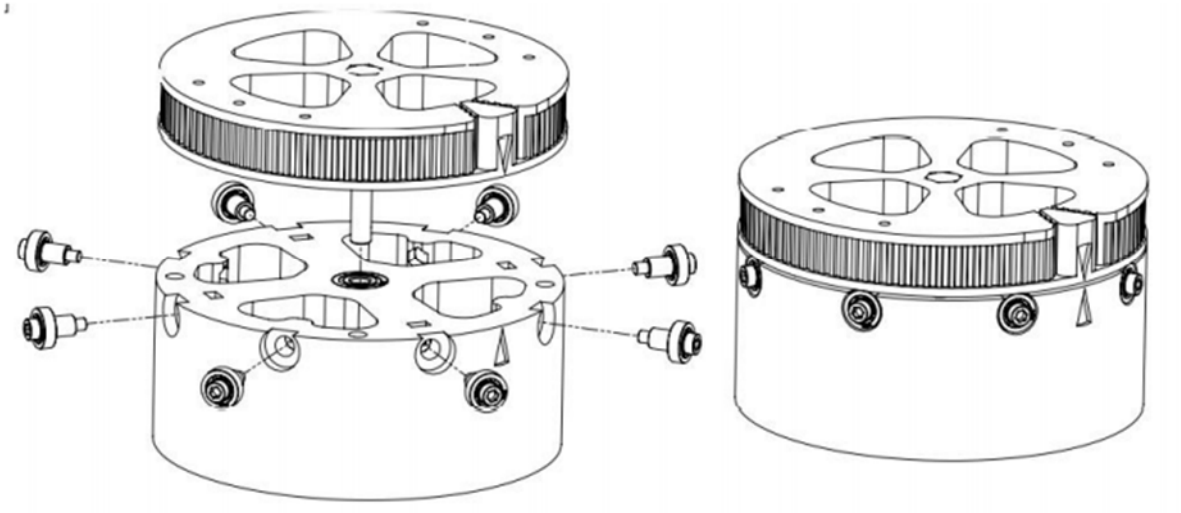


Figure 26: La forme finale de la base [15].

Pour faire tourner la base, on utilise un moteur Nema 17 et on place une pulley sur l'arbre du moteur. Le moteur sera fixé sur la base du bois :

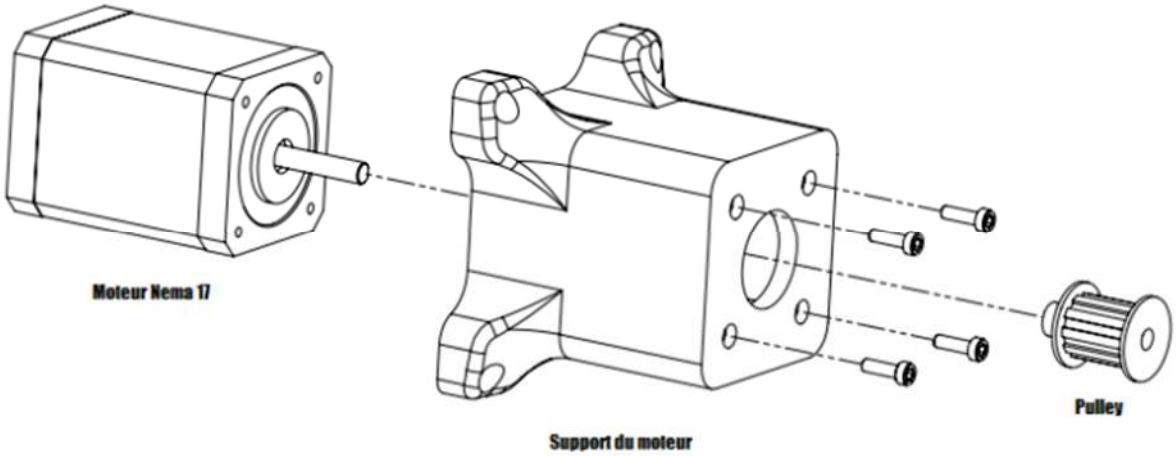


Figure 27: Le moteur pas à pas (Nema 17) qui assure le mouvement de l'articulation 1 [15].

L'utilité de la poulie est de relier le moteur avec la base (la plaque rotative) avec une corroie T5 de longueur 62cm.

2.4.3 Articulation 2 : l'épaule (Shoulder) :

Cette articulation est très importante car elle doit supporter tout le reste du bras, et c'est pour cela qu'elle est formée par une base qui est la pièce la plus grande dans le bras :

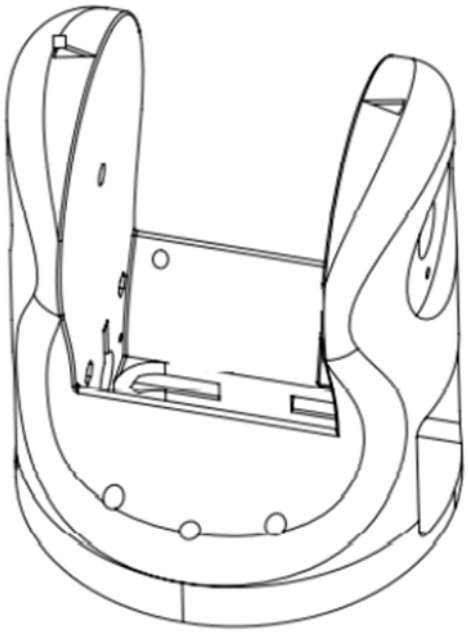


Figure 28: la base de l'articulation 2 [15].

Après ça on place les deux moteurs Nema 23 taille 112mm :

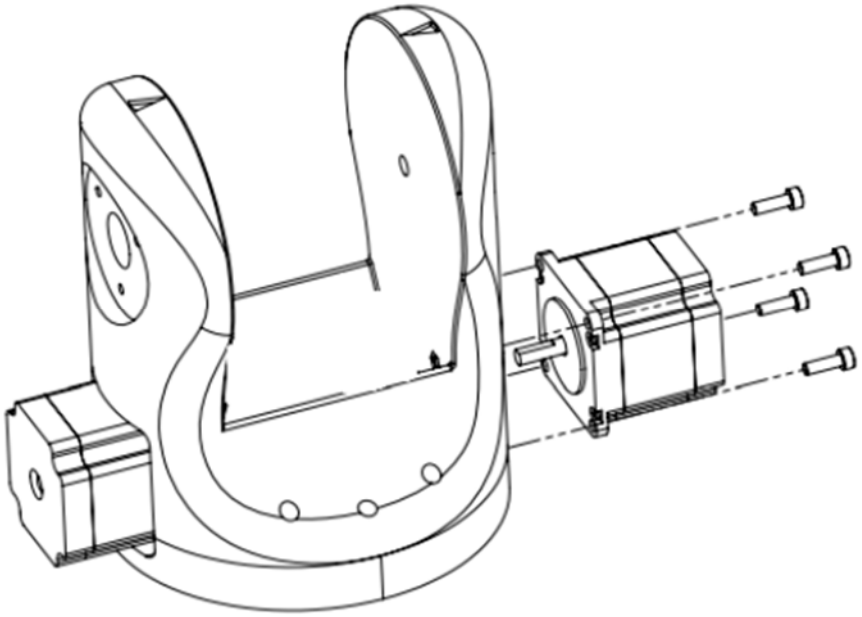


Figure 29: L'emplacement des deux moteurs Nema 23 [15].

On ajoute deux pièces d'engrenage qui seront relier au moteur avec deux courroies de type T5 de longueur 36cm chacune :

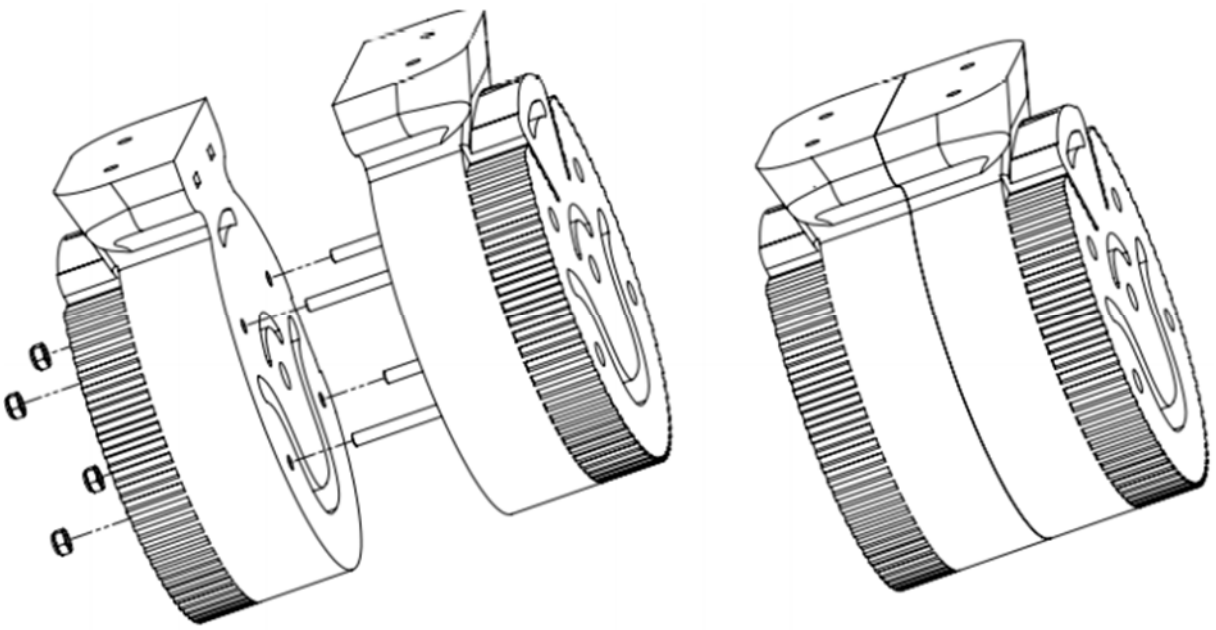


Figure 30: Les deux pièces d'engrenages de l'articulation 2 [15].

On règle la tension des courroies à l'aide d'un tendeur qui est un mécanisme à roulements :

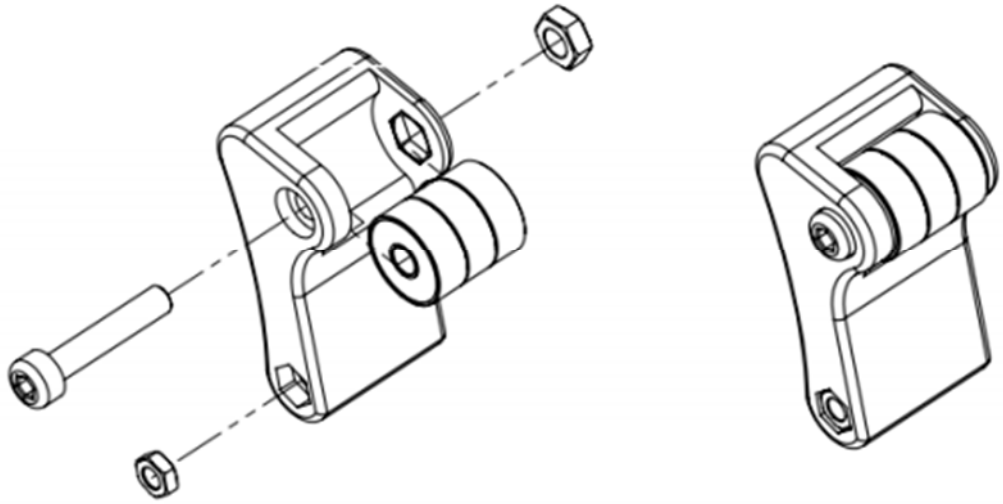


Figure 31: Le tendeur qui permet le réglage de la tension des courroies [15].

On fixe deux tendeurs (un pour chaque courroie), et on place les deux engrenages sur la base de l'articulation et on les fixe avec une barre lisse de longueur 14cm :

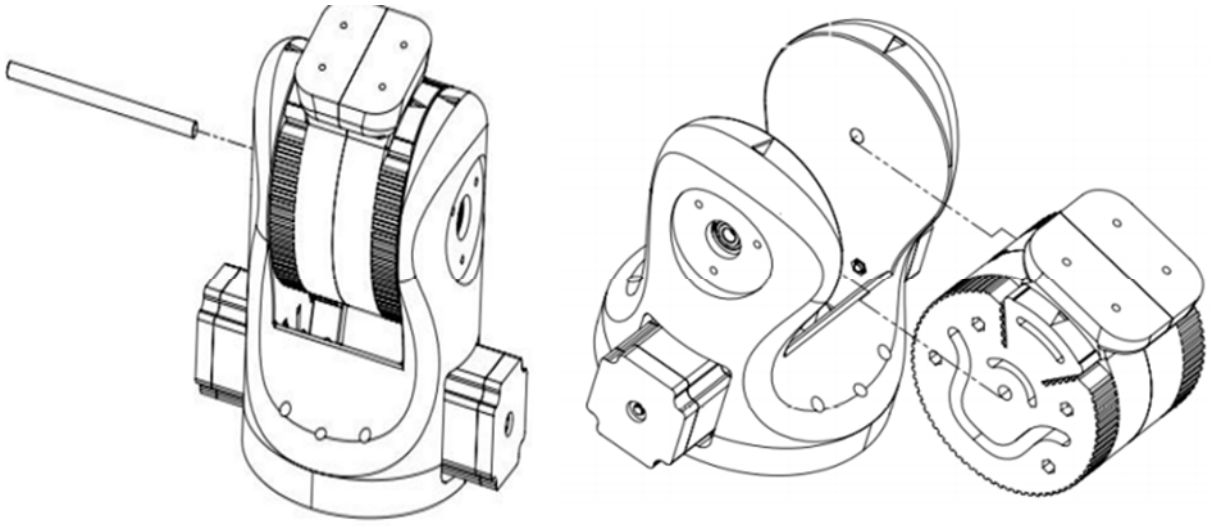


Figure 32: L'assemblage de l'articulation 2 [15].

2.4.4 Articulation 3 : le coude (Elbow) :

On commence par fixant la base de cette articulation sur l'articulation précédente :

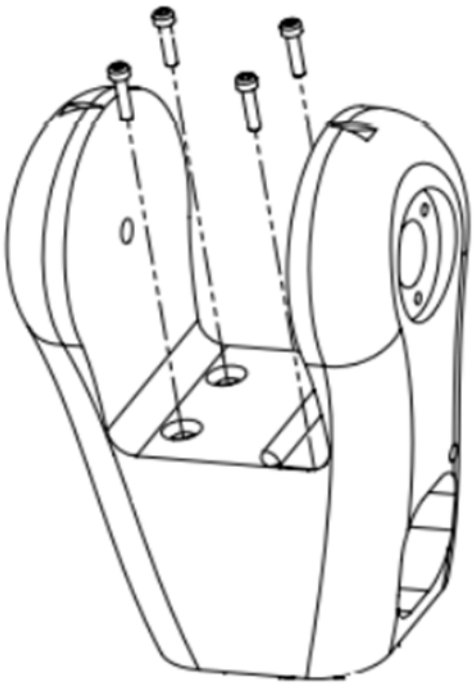


Figure 33: La base de l'articulation 3 [15].

Le mécanisme pour construire l'articulation est le même, on place la pièce d'engrenage et le moteur Nema 17 avec réducteur mécanique de rapport 5:1 et on ajoute une poulie sur l'arbre de ce moteur. Ensuite on monte une corroie T5 de longueur 42cm. La corroie sera ajustée par un tendeur.

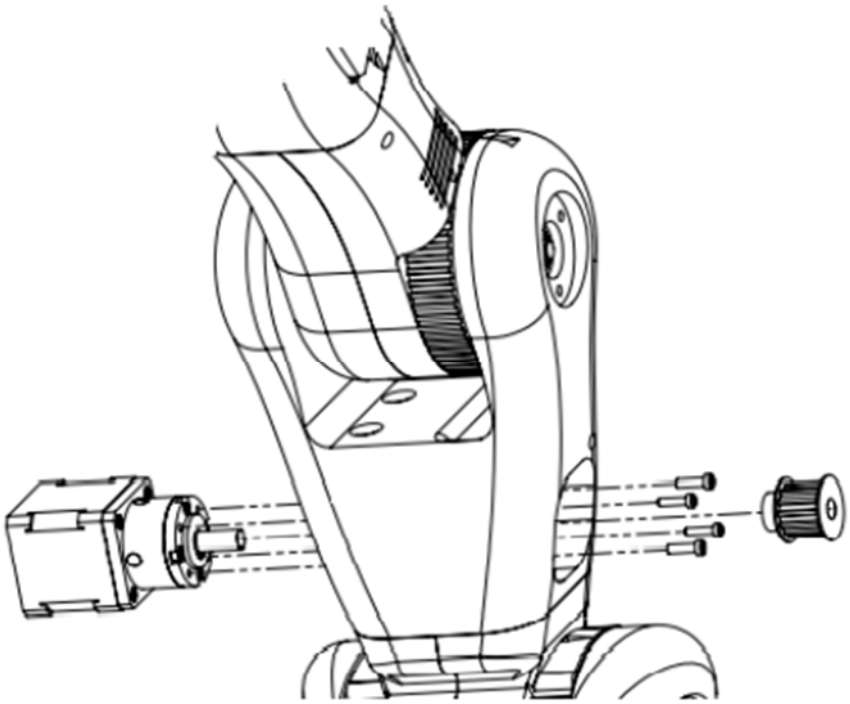


Figure 34: La forme finale de l'articulation 3 [15].

2.4.5 Articulation 4 : le poignet1 (Wrist1) :

On commence par plaçant un moteur Nema 17 dans la pièce de base de cette articulation, et on lui ajoute un coupleur qui permet de coupler l'arbre du moteur avec une barre de taille 38mm.

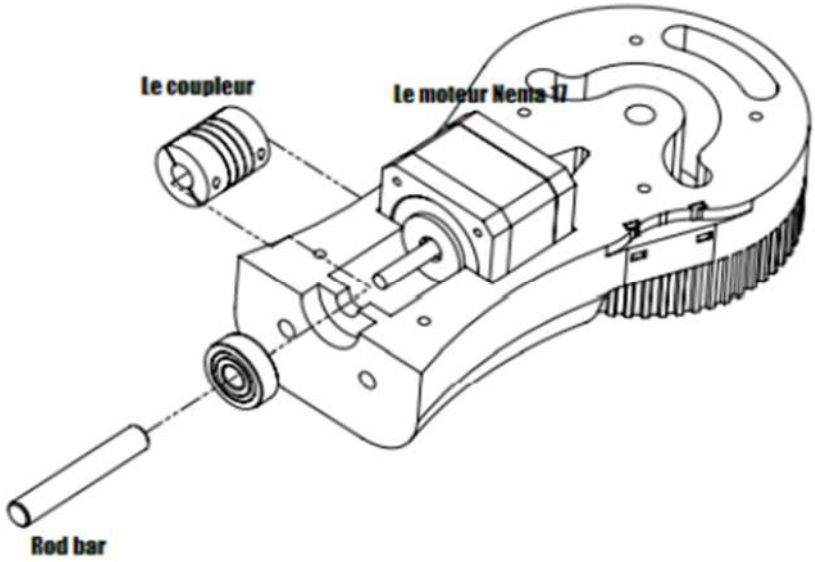


Figure 35: L'ajout du Moteur Nema 17 et le coupleur à la pièce [15].

Après ça, on ajoute la partie supérieure et on les met ensemble à l'aide des vis et des écrous :

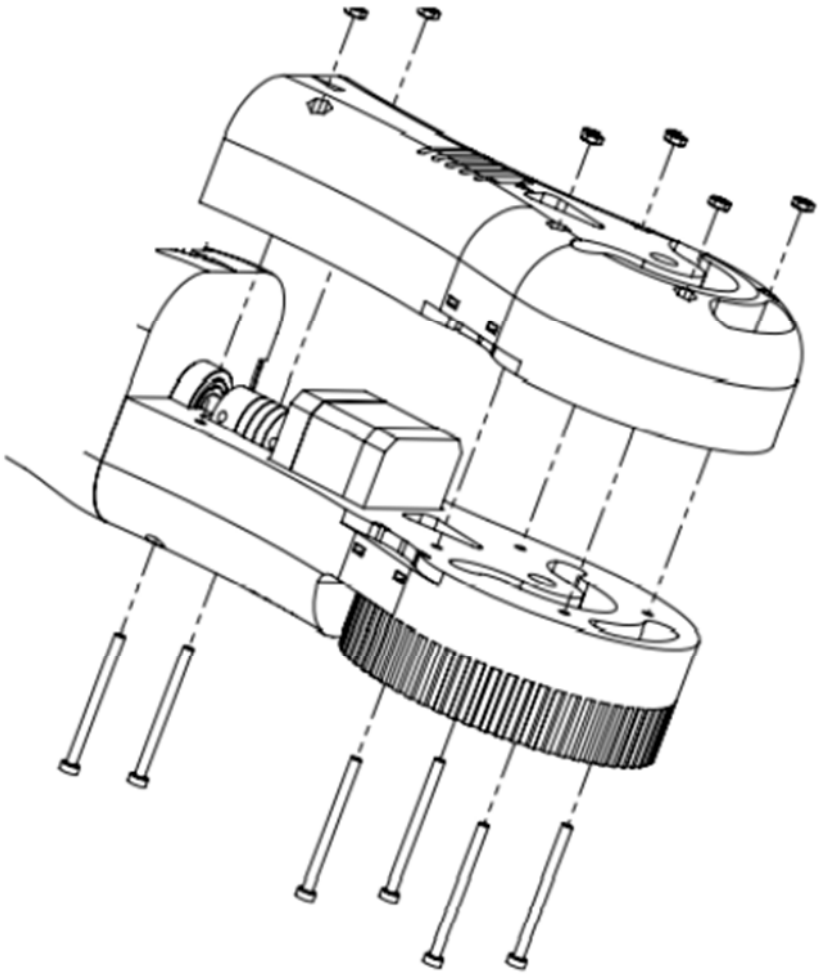


Figure 36: Montage de l'articulation 4 [15].

Cette articulation doit être construite avant l'articulation 3, car l'engrenage de cette articulation fait partie de cette pièce (celle de l'articulation 3).

2.4.6 Articulation 5 : le poignet2 (Wrist2) :

On commence par le placement de la base de cette articulation sur l'articulation précédente à l'aide d'un roulement à billes et un écrou M8 :

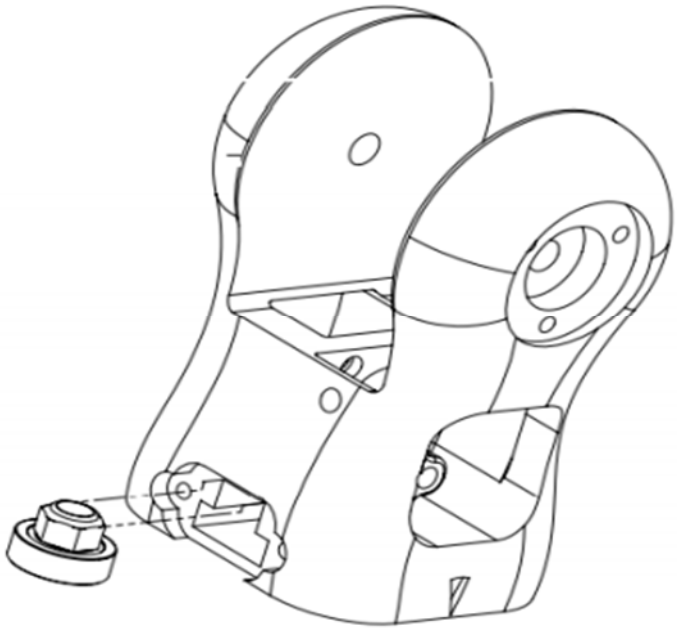


Figure 37: La base de l'articulation 5 [15].

Le système de moteur + poulie + courroie T5 + engrenage reste le même que dans les articulations précédentes.

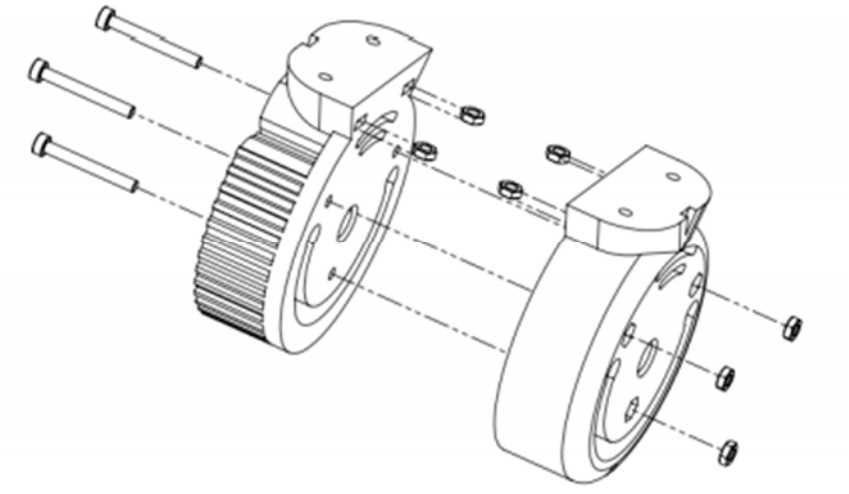


Figure 38: L'engrenage de l'articulation 5 [15].

On place l'engrenage et on le fixe avec une barre lisse, après ça on ajoute le moteur Nema 17 et la poulie :

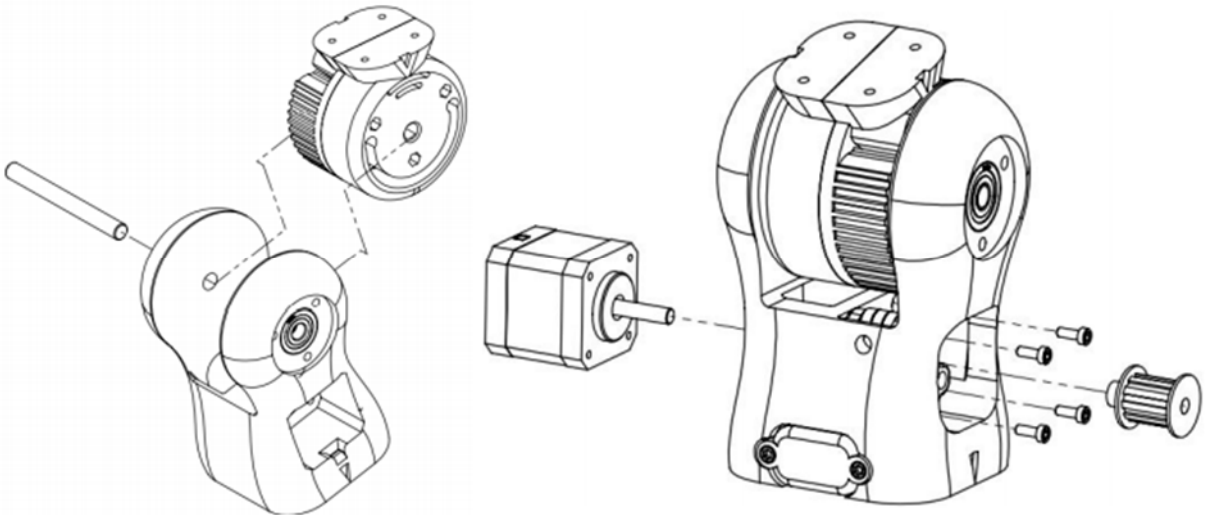


Figure 39: Montage de l'articulation 5 [15].

2.4.7 La pince ou l'organe terminale (gripper) :

On assemble la pince (gripper) et on la fixe sur la 5ème articulation :

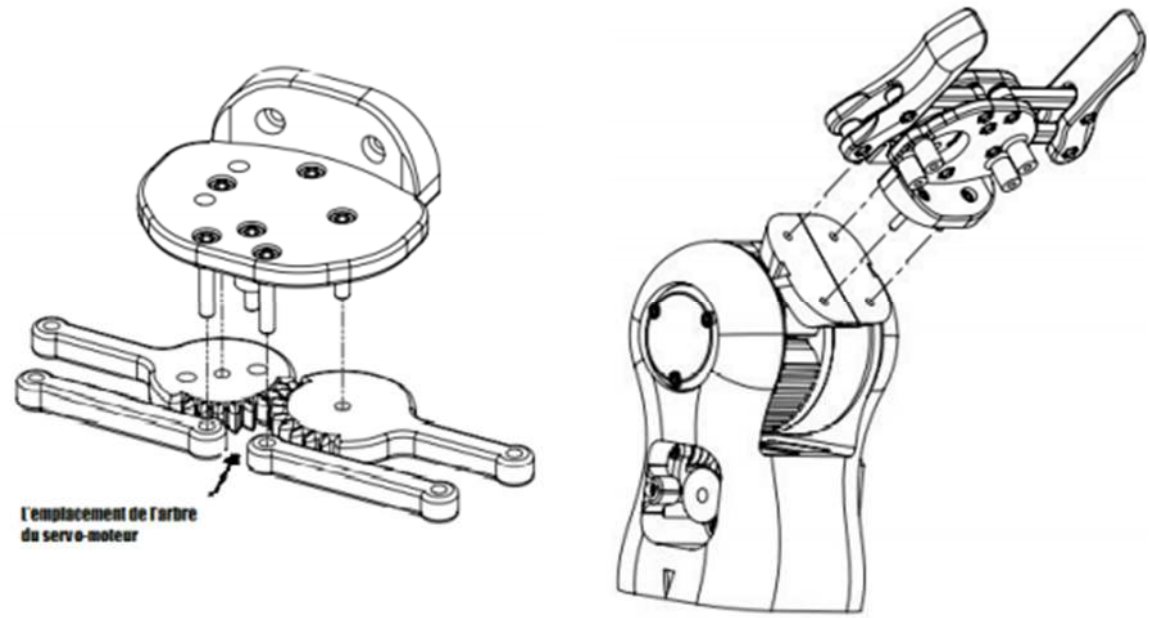


Figure 40: L'assemblage du gripper et sa fixation sur l'articulation 5 [15].

Pour que la pince fonctionne, on lui ajoute un servomoteur. Quand ce dernier fait des pas, il fait tourner un système d'engrenage qui engendre l'ouverture et la fermeture de la pince.

Après qu'on a fini de construire l'articulation 5 avec la pince, on la relie avec l'articulation 4 :

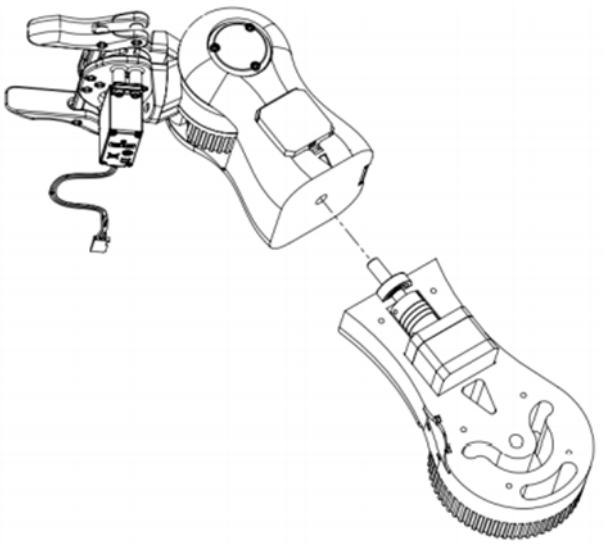


Figure 41: Assemblage des Articulations 4 et 5 + le gripper [15].

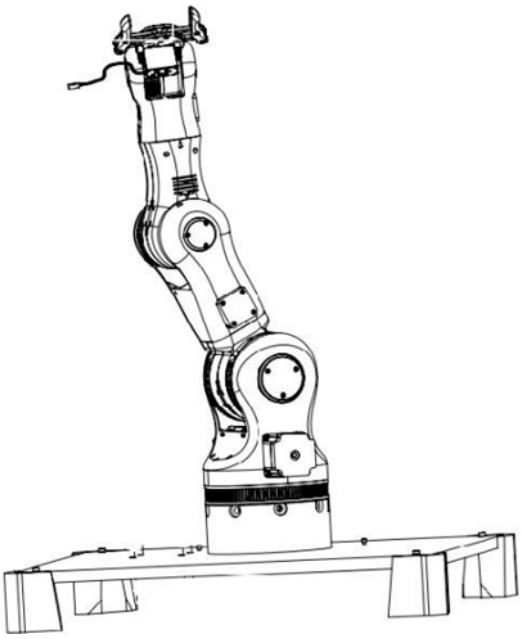


Figure 42: La forme finale du bras de robot après assemblage [15].

2.5 Conclusion :

Dans ce chapitre nous avons détaillé la construction du bras de robot dans deux parties : la partie électrique et la partie mécanique. Maintenant et vue que la construction est terminée on va attaquer la phase de commande du bras de robot.

Chapitre 3

L'ensemble des logiciels utilisés et leurs fonctionnements.

3 L'ensemble des logiciels utilisés et leurs fonctionnements :

Introduction :

Dane ce chapitre, nous allons détailler l'ensemble des logiciels utilisés pour la réalisation de ce projet, on va les introduire et expliquer leurs rôles et leurs fonctionnalités, ainsi que les raisons qui nous ont poussé pour les choisir.

3.1 Linux/GNU :

Linux est un système d'exploitation comme Windows 10 ou Mac OS. La partie importante de n'importe quel système d'exploitation est le noyau ou le Kernel. Dans le système GNU/Linux, Linux (www.linux.org) est le composant du noyau. Le système d'exploitation basé sur Linux est inspiré du système d'exploitation Unix. Le noyau (Kernel) Linux est capable d'effectuer plusieurs tâches dans des systèmes multi-utilisateurs. GNU/Linux est gratuit à tout le monde et ces programmes et logiciels sont open source, ce qui le rend très attirant pour les développeurs et les programmeurs. Les utilisateurs ont un contrôle total sur le système d'exploitation [20].

Linux est largement utilisé dans les serveurs. Le système utilisé dans les téléphone smart Android fonctionne à base d'un noyau Linux. Il existe de nombreuses distributions, de Linux, qui utilise essentiellement le noyau Linux comme composant principal. Parmi les distributions Linux les plus utilisées, on trouve : Ubuntu, Debian et Fedora [20].

Figure 43: Les logs de quelques distributions de Linux [20].

Dans notre projet, on va utiliser la distribution dite **Ubuntu** du Linux.

3.2 Ubuntu :

Ubuntu (www.ubuntu.com) est une distribution Linux très populaire basée sur l'architecture Debian. Il est disponible gratuitement, et il est open source.

Pour réaliser une application en robotique, il faut être capable de communiquer avec les actionneurs et les capteurs présent dans les robots. Un système d'exploitation basé sur Linux peut fournir ces capacités et peut nous permettre d'interagir avec le matériel utilisé. Les avantages d'Ubuntu dans ce contexte sont la rapidité de sa réponse, sa légèreté ainsi que sa sécurité. Au-delà de ces facteurs, Ubuntu a une grande communauté qui fournit beaucoup de support en ligne. Ubuntu a également un support à long terme (LTS), qui dure jusqu'à cinq ans. Ces facteurs ont conduit les développeurs de la robotique à s'en tenir à Ubuntu, et ainsi, c'est le seul système d'exploitation à base de Linux qui est entièrement pris en charge par ROS [19] et [20].

Le terminale et les commandes dans Ubuntu :

Les lignes de commande sont plus rapides que la méthode traditionnelle de l'exécution des applications et des programmes, et est souvent utilisé par les utilisateurs de Linux. L'interface de lignes de commandes sous Linux peut être

Figure 44: Terminale sous Ubuntu.

Comparé au système (DOS) sous Windows. L'interface de lignes de commandes sous Ubuntu se trouve dans un outil appelé Terminal [19] et [20].

3.3 ROS :

ROS est un middleware, il exécute de nombreuses fonctions d'un système d'exploitation, mais il nécessite toujours un système d'exploitation comme Linux pour pouvoir fonctionner correctement. L'un de ses principaux objectifs est d'assurer la communication entre l'utilisateur, le système d'exploitation et le matériel externe à l'ordinateur. Ce matériel peut inclure des capteurs, des caméras, ainsi que des robots. L'avantage de ROS est sa capacité de permettre à l'utilisateur de contrôler un robot sans devoir savoir toutes ses détails [21] et [23].

Les distributions ROS sont très similaires aux distributions Linux. Chaque distribution maintient un ensemble stable de packages qui assurent le

fonctionnement de ROS. Les distributions ROS sont complètement compatibles avec Ubuntu.

Voici les dernières distributions ROS :

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame (**Recommended**)	May 23rd, 2016			April, 2021 (Xenial EOL)

Figure 45: Les distributions de ROS [21].

ROS est utilisé dans un nombre important des robots et est compatible avec ces logiciels et ces capteurs. Voici quelques robots fonctionnant entièrement sur ROS :

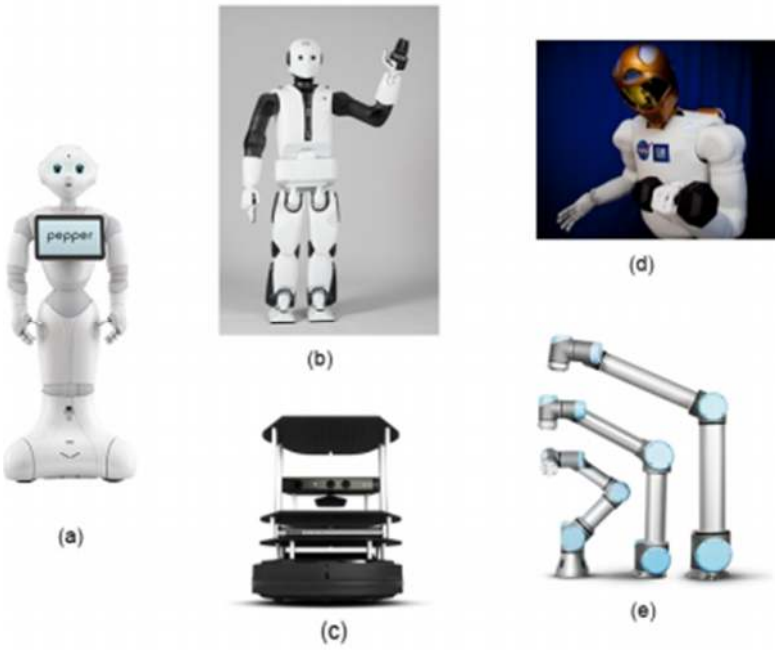


Figure 46: Quelques robot fonctionnant sur ROS : Pepper (a), REEM-C (b), Turtlebot (c), Robonaut (d), et Universal Robots (e) [21].

3.4 URDF :

ROS a une approche différente pour représenter les modèles des robots. Il les définit dans une format XML appelé Universal Robotic Description Format (URDF). URDF est une liste des tags XML qui contient toutes les informations nécessaires pour la modélisation du robot. Pour modéliser un robot dans ROS, on doit créer un fichier et définir la relation entre chaque lien et articulation dans le robot et ensuite, on doit enregistrer le fichier avec l'extension «. urdf » [17] et [22].

Les tags suivants sont les tags URDF couramment utilisées pour composer un modèle de robot URDF [22]:

- **Link :** le tag de lien « Link » représente un lien unique d'un robot. En utilisant ce tag, on peut modéliser un lien robotique et ses propriétés. La modélisation comprend la taille, la forme, la couleur, et on peut même importer un maillage 3D pour représenter le robot (fichier STL). Nous pouvons également fournir les propriétés dynamiques du lien, telles que la matrice d'inertie.

La syntaxe de ce tag est la suivante :

<link name="<nom du lien >">

<inertial>.....</inertial>

<visual> .....</visual>

<collision>.....</collision>

</link>

- **Joint:** le tag d'articulation « Joint » représente une articulation du robot. On peut spécifier la cinématique et la dynamique de l'articulation, et fixer les limites du mouvement articulaire ainsi que ses vitesses. Ce tag prend en charge les différents types d'articulations qui peuvent se présenter dans un robot.

La syntaxe de ce tag est la suivante :

<joint name="<nom de l'articulation>">

<parent link="link1"/>

<child link="link2"/>

<calibration .... />

<dynamics damping ..../>

<limit effort .... />

</joint/>

- **Robot :** Ce tag encapsule l'ensemble du modèle du robot. A l'intérieur de ce tag robot, nous pouvons définir le nom du robot, les liens, et les articulations du robot.

La syntaxe de ce tag est la suivante :

<robot name="<nom du robot>"

<link> ..... </link>

<link> ..... </link>

<joint> ..... </joint>

<joint> .....</joint>

</robot>

- **Gazebo :** Ce tag est utilisé lorsque on veut inclure les paramètres de simulation du Simulateur de Gazebo dans notre fichier URDF. On peut utiliser ce tag pour inclure des plugins de Gazebo, des propriétés des matériaux, etc.

La syntaxe de ce tag est la suivante :

<gazebo reference="link_1">

<material>Gazebo/Black</material>

</gazebo>

3.5 SolidWorks :

SOLIDWORKS est un logiciel de conception assistée par ordinateur 3D fonctionnant sous Windows. SOLIDWORKS est utilisé pour développer des systèmes mécatroniques du début à la fin. Au stade initial, le logiciel est utilisé pour la planification, la modélisation, l'évaluation de la faisabilité, le prototypage et la gestion de projet. Le logiciel est ensuite utilisé pour la conception et la construction d'éléments mécaniques, électriques et logiciels. Enfin, le logiciel peut être utilisé pour la gestion des appareils, l'analyse, l'automatisation des données et les services cloud [25].

SolidWorks nous permet de générer un fichier URDF initial à l'aide d'un outil appelé « SW2URDF » dans lequel on définit chacun des liens dans l'ordre dont ils apparaissent dans le manipulateur robot (parent child), avec le type d'articulation (joints), les axes de rotation, les origines et l'environnement de travail [26].

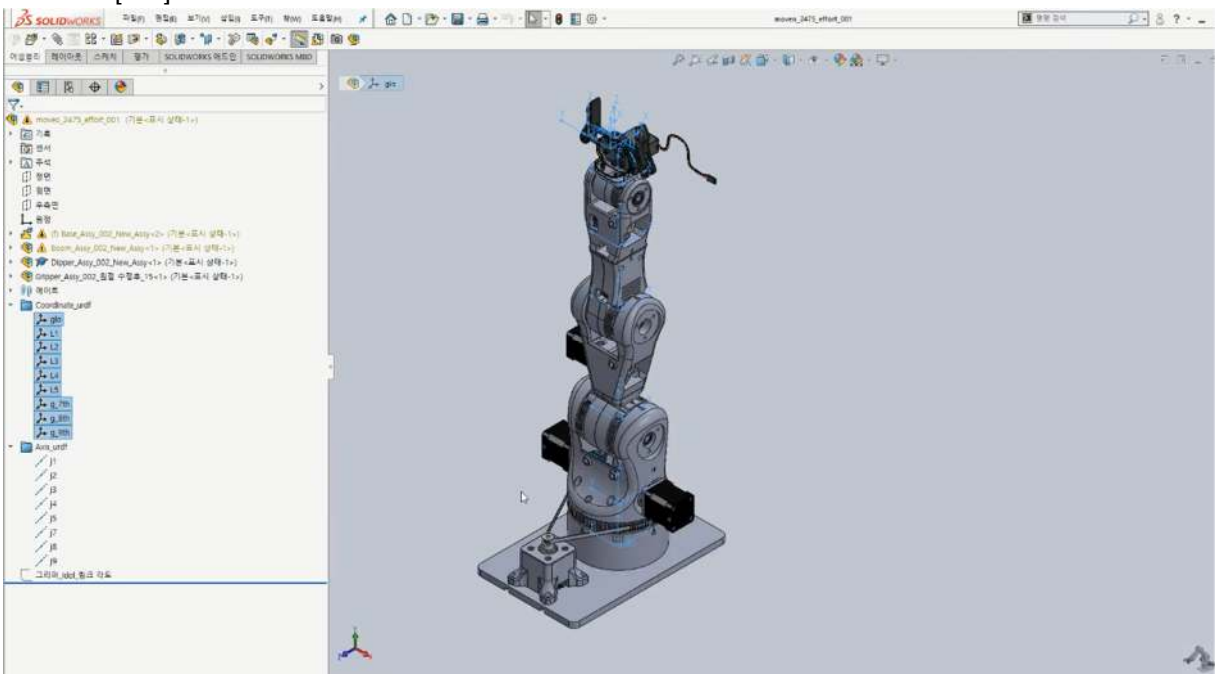


Figure 47: Modélisation du bras Moveo par SolidWorks pour la génération du fichier URDF.

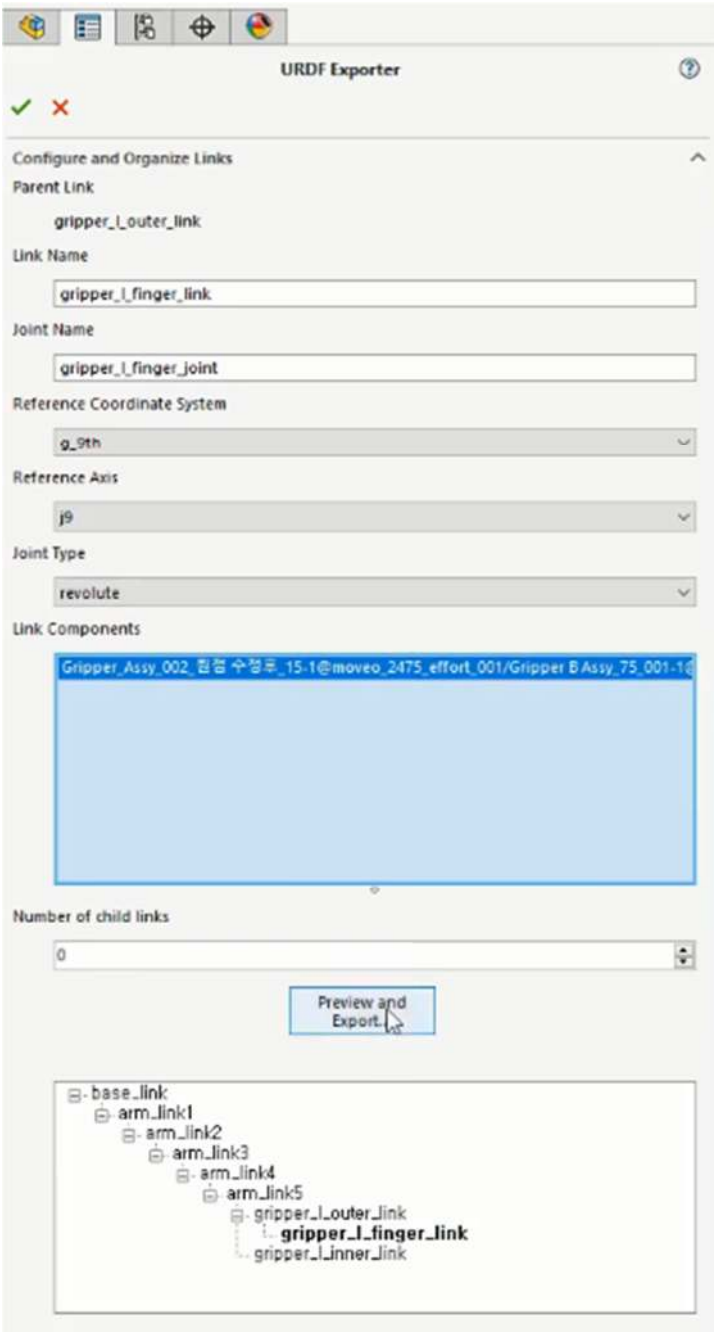


Figure 48: Définition des Liens du robot selon leurs ordres par l'outil SW2URDF de SolidWorks.

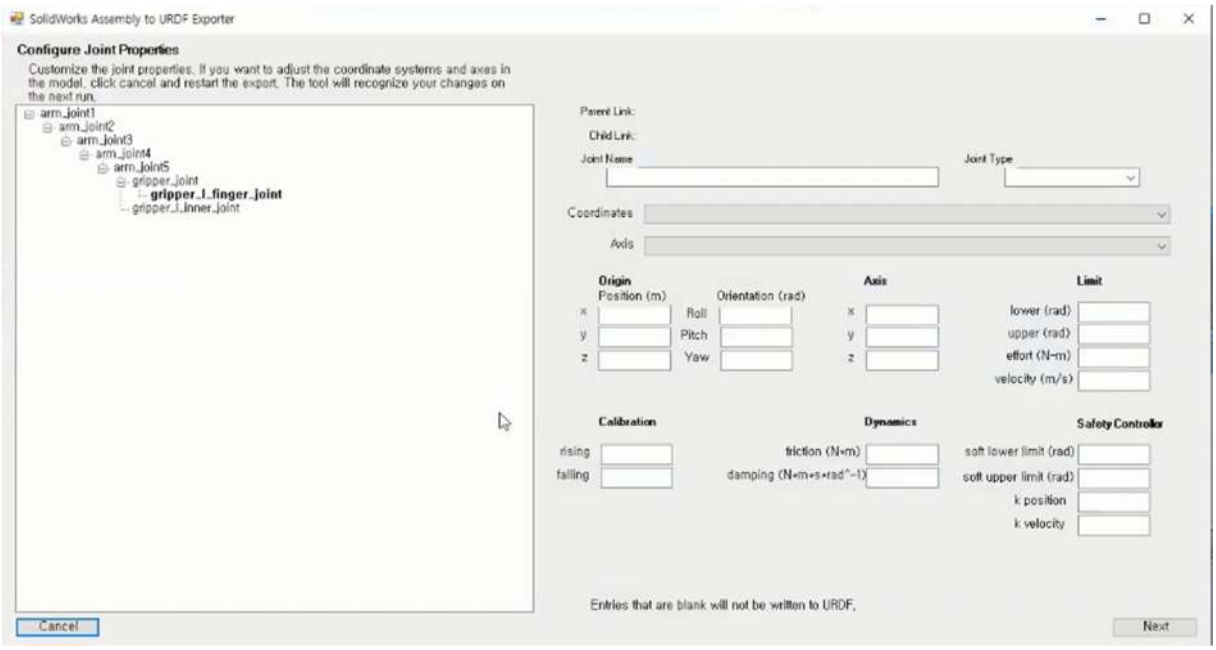


Figure 49: Spécification des paramètres supplémentaires du fichier URDF pour chaque lien du bras Moveo.

```
<?xml version="1.0" ?>
<robot name="manipulator">

    <link name="world"/>
    <link name="base_link">
        <visual>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <mesh filename="package://manipulator_description/meshes/base_link.stl" scale="1 1 1"/>
            </geometry>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <mesh filename="package://manipulator_description/meshes/base_link.stl" scale="1 1 1"/>
            </geometry>
        </collision>
        <inertial>
            <mass value="1.0"/>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
        </inertial>
    </link>

    <joint name="base_link_fixed" type="fixed">
        <parent link="world"/>
        <child link="base_link"/>
        <origin xyz="0 0 0" rpy="0 0 0"/>
    </joint>

    <link name="arm_link1">
        <visual>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <mesh filename="package://manipulator_description/meshes/arm_link1.stl" scale="1 1 1"/>
```

Figure 50: Résultat de la génération du fichier URDF par l'outil SW2URDF.

3.6 ROS Visualizer RViz :

RViz est l'un des outils de visualisation 3D disponibles sur ROS, cet outil nous permet de visualiser des données 2D et 3D sous ROS. RViz nous permet de visualiser les données tels que les modèles des robots, les données de transformation 3D du robot (TF), les point-clouds, les données d'images, et une variété de données de capteurs différents [20],[21] et [22].

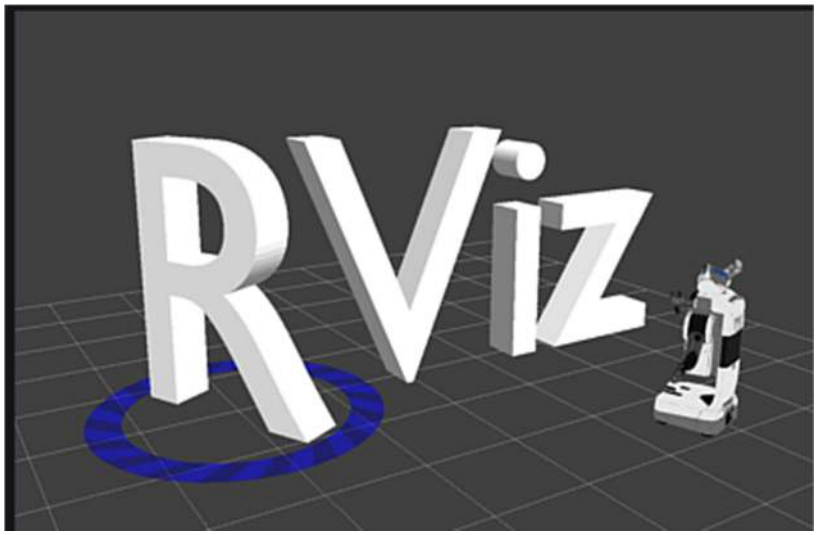


Figure 51: Le Logo de RViz.

On peut visualiser notre bras de robot Moveo et le commander en spécifiant des trajectoires prédéfinies dans RViz en générant les fichiers nécessaires à l'aide d'un autre outil de ROS, cet outil s'appelle MoveIt !

MoveIt ! contient des logiciels très sophistiqués pour la planification de mouvement et trajectoires, la manipulation des robots, la perception 3D, la cinématique, la vérification de collision, ainsi que la détection des obstacles [21].

On peut générer les fichiers nécessaires de manipulation MoveIt ! à l'aide du **MoveIt ! Setup Assistant**. Cet assistant permet la création des packages MoveIt ! à partir du fichier URDF. On peut lancer cet assistant en exécutant la commande suivante : **roslaunch moveit_setup_assistant setup_assistant.launch.** Après l'exécution de cette commande, on obtient la fenêtre suivante :

Figure 52: MoveIt ! Setup Assistant.

On commence par la sélection du modèle URDF du Robot, après cette sélection, on peut voir le modèle de notre robot dans la fenêtre du MoveIt !

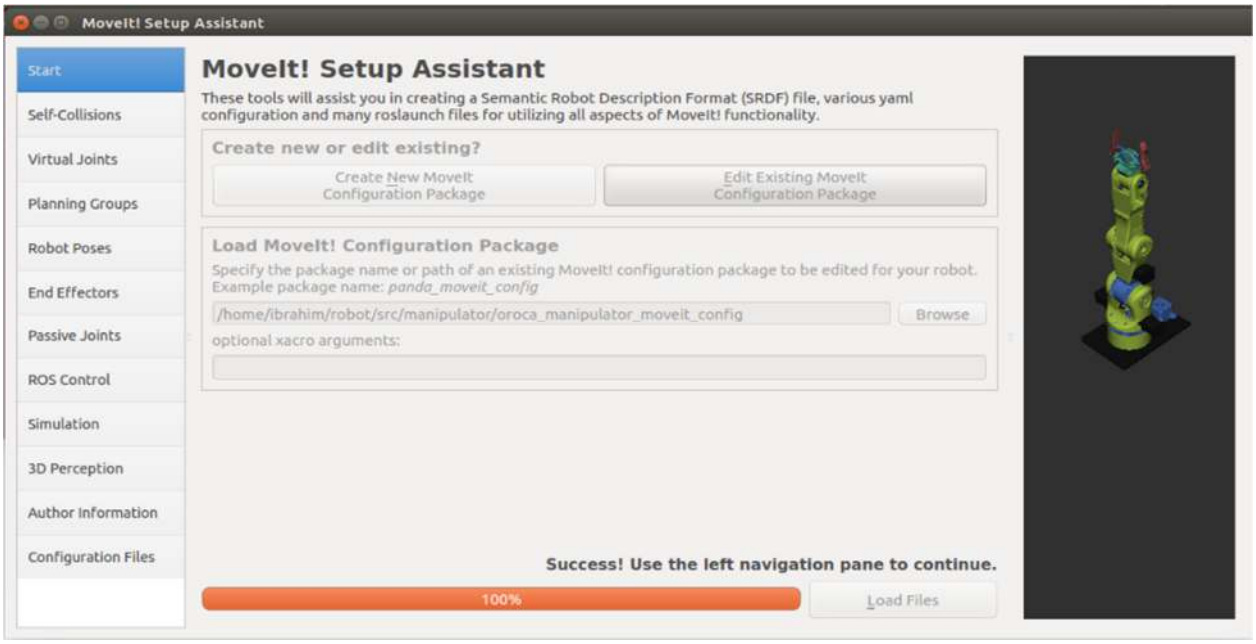


Figure 53: Le modèle du bras Moveo dans MoveIt !

L'étape suivante est la génération de la matrice de collision :

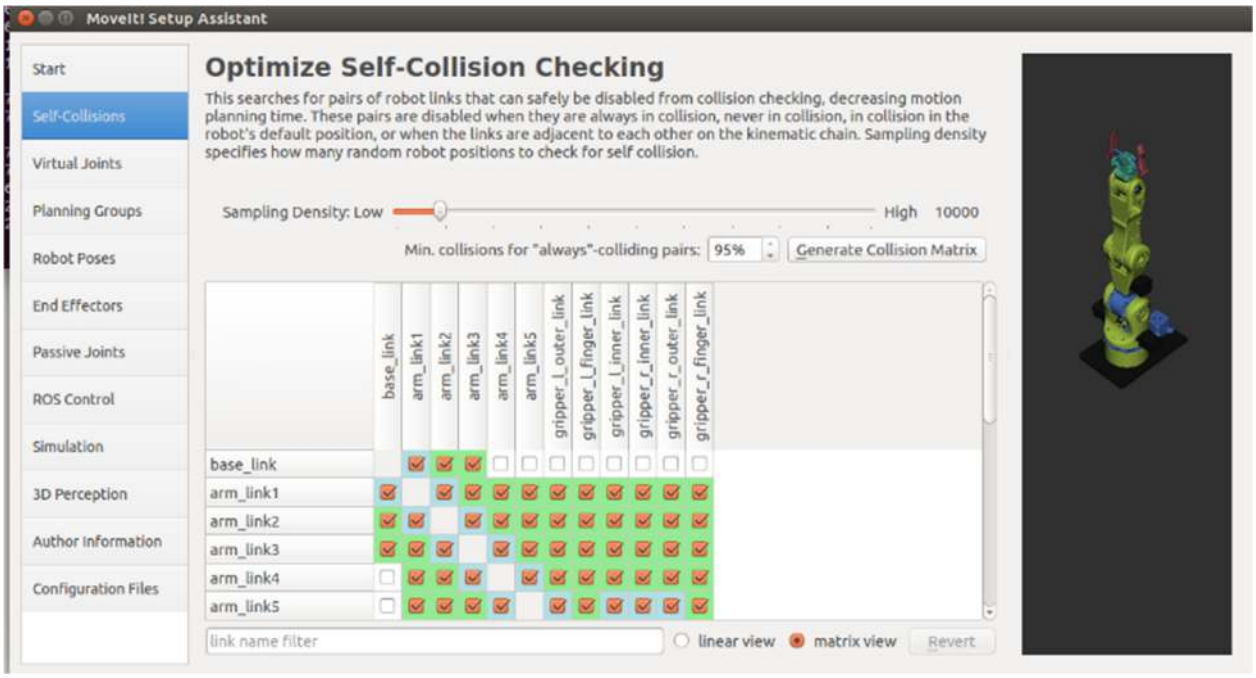


Figure 54: génération de la matrice de collision.

Ensuite, on va définir les **planning groups** de notre robot, ainsi que les algorithmes qu'on souhaite utiliser pour résoudre le problème de la géométrique inverse, **kdl_kinematics_plugin/KDLKinematicsPlugin** dans notre cas :

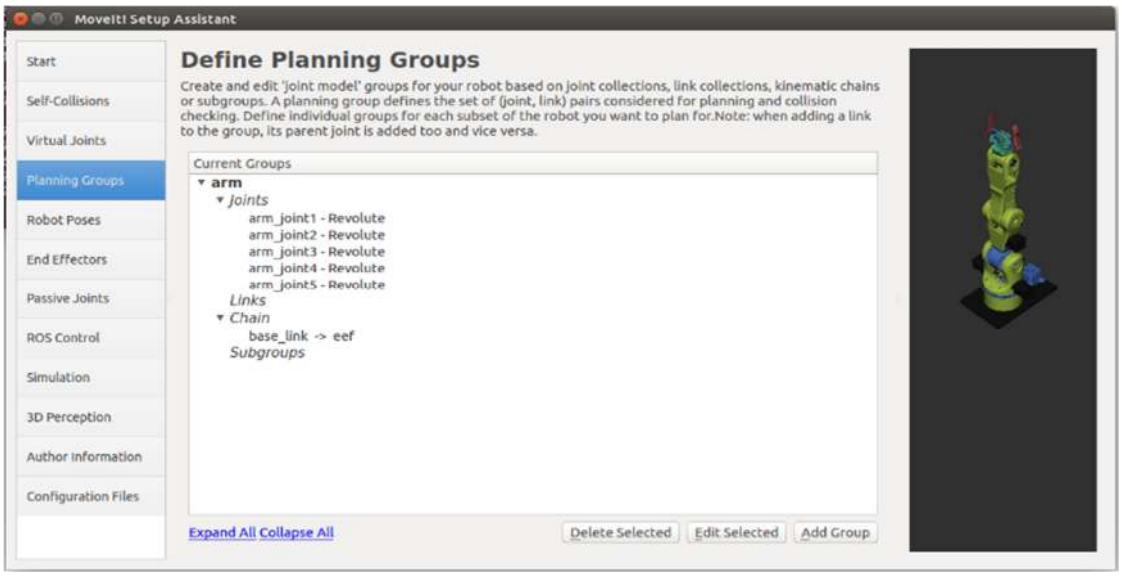


Figure 55: Définition des planning groups.

Ensuite, on peut définir des positions prédéfinies de notre robot en spécifiant la valeur des

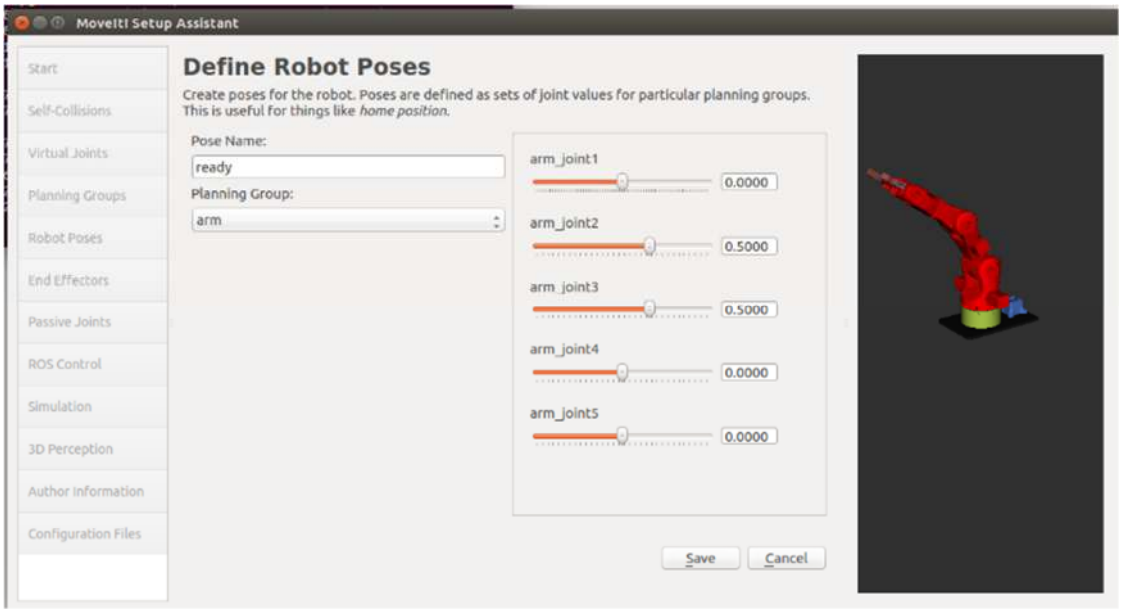


Figure 56: définition des positions pour le bras Moveo.

Ensuite, on peut ajouter des contrôleurs standard pour pouvoir commander notre bras, ces contrôleurs sont des simples régulateurs PID qu'on peut spécifier à chaque articulation :

Figure 57: L'ajout des régulateurs pour chaque articulation.

L'étape finale est la génération des fichiers de configurations qui nous permettent de visualiser et commander notre robot :

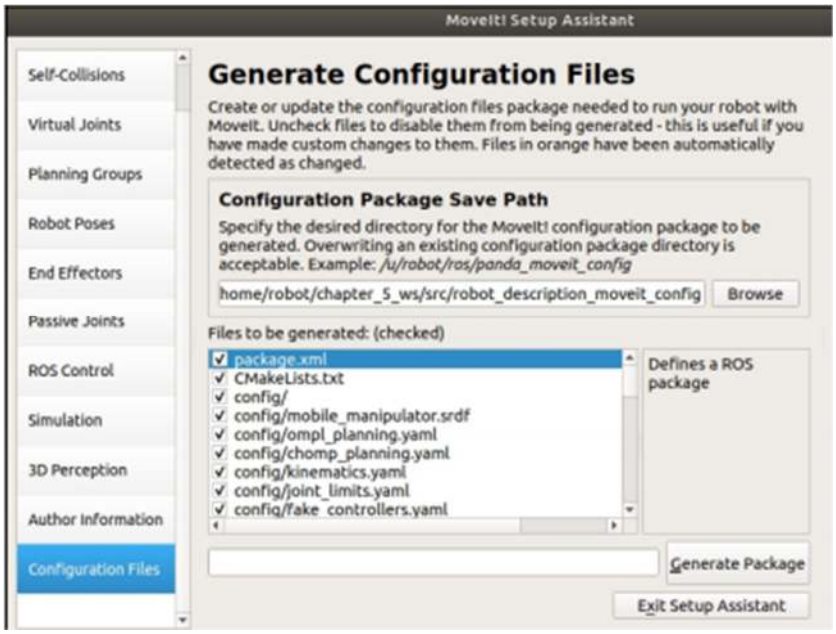


Figure 58: Génération des Packages MoveIt !

Une fois MoveIt ! configuré, on peut tester et manipuler notre bras robotique Moveo à l'aide de l'interface graphique (plugin RViz) :

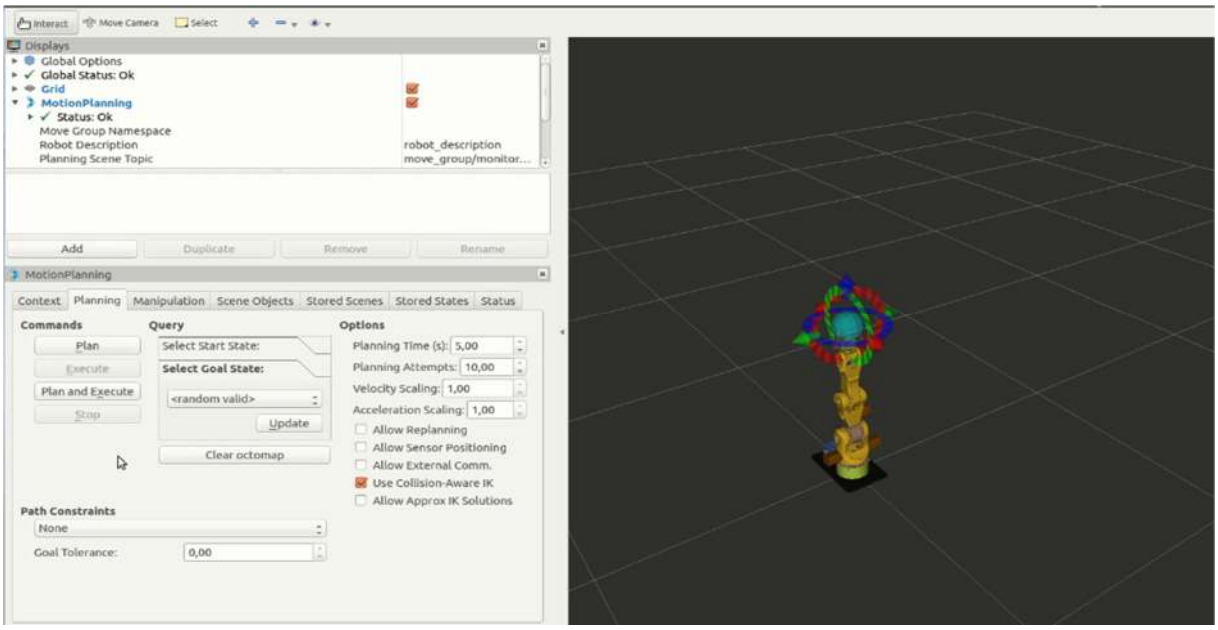


Figure 59: Manipulation du robot dans RViz.

Pour réaliser la tache de reconnaissance d'objets on utilise un système de vision par réseaux de neurones à convolution pour détecter et classifier les objets connus captés par une webcam.

Pour cela, on a utilisé les bibliothèques et les plateformes suivantes :

3.7 TensorFlow

Figure 60: Logo du TensorFlow.

TensorFlow est une plate-forme Open Source de bout en bout dédiée au machine learning. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine learning, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie. Il existe un API [27] de TensorFlow construit par l'équipe de Google qui regroupe tous ces algorithmes et bibliothèques, et c'est ça ce qu'on va utiliser pour réaliser la détection des objets et l'identification de leurs classes.

On peut utiliser cet API pour entrainer des algorithmes et modèles qui peuvent détecter des objets et les classifier. Dans notre cas, on va utiliser des modèles pré-entrainés qui sont déjà inclus dans l'API de Google [28]. Ces modèles sont pré-entrainés sur la base de donnés COCO [29] qui est un ensemble de donnés de détection d'objets à grande échelle. COCO contient 330 000 images, qui peuvent être classées en 80 catégories d'objets, 91 catégories de trucs et elle contient 250000 personnes. COCO propose deux tâches de détection d'objets : en utilisant soit une sortie de cadre de délimitation, soit une sortie de segmentation d'objet (cette dernière est également appelée segmentation d'instance).

Le logiciel TensorFlow gère les jeux de données en les disposant sous forme de nœuds de calcul dans un graphe d'exécution. Les liens qui unissent les nœuds d'un graphe peuvent représenter des matrices ou vecteurs multidimensionnels, créant ce qu'on appelle des tenseurs. Dans la mesure où les programmes TensorFlow utilisent une architecture à flux de données qui tend à généraliser les résultats de calcul intermédiaires, ils conviennent particulièrement bien aux applications à traitement parallèle de très grande échelle : les réseaux de neurones sont un exemple courant. Les applications TensorFlow peuvent s'exécuter sur des unités centrales traditionnelles (CPU), sur des processeurs graphiques de haute performance (GPU) ou sur les unités de traitement de tenseurs de Google (TPU, Tensor Processing Unit).

3.8 OpenCV :

Figure 61: Logo du OpenCV.

OpenCV (Open Source Computer Vision [31] est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur, accessibles à travers un API pour les langages C, C++, et Python. Elle est distribuée sous une licence BSD (libre) pour les plateformes Windows, GNU/Linux, Android et MacOs.

C'est la bibliothèque de référence pour la vision par ordinateur, aussi bien dans le monde de la recherche que celui de l'industrie [32].

Les principaux modules accessibles à travers son API C++ sont :

- **Core**: les fonctionnalités de base. Cette bibliothèque permet de manipuler les structures de base, réaliser des opérations sur des matrices, dessiner sur des images, sauvegarder et charger des données dans des fichiers XML...
- **Imgproc** : traitement d'image. Nous entrons dans le cœur du sujet. Les fonctions et structures de ce module ont trait aux transformations d'images, au filtrage, à la détection de contours, de points d'intérêt...
- **Highgui** : entrées-sorties et interface utilisateur. OpenCV intègre sa propre bibliothèque haut-niveau pour ouvrir, enregistrer et afficher des images et des flux vidéo. Celle-ci contient aussi un certain nombre de fonctions permettant de réaliser des interfaces graphiques très simples, mais largement suffisantes pour tester nos programmes. Les 2500 algorithmes fournit par OpenCV permettent d'effectuer tout un tas de traitements sur des images (extraction de couleurs, détection de visages, de formes, application de filtres, ...). Ces algorithmes se basent principalement sur des calculs mathématiques complexes, concernant surtout les traitements sur les matrices car une image peut être considérée comme une matrice de pixels qui sont repérés avec un degré

de gris (c'est une propriété de OpenCV) comme le représente la figure suivant [33] :

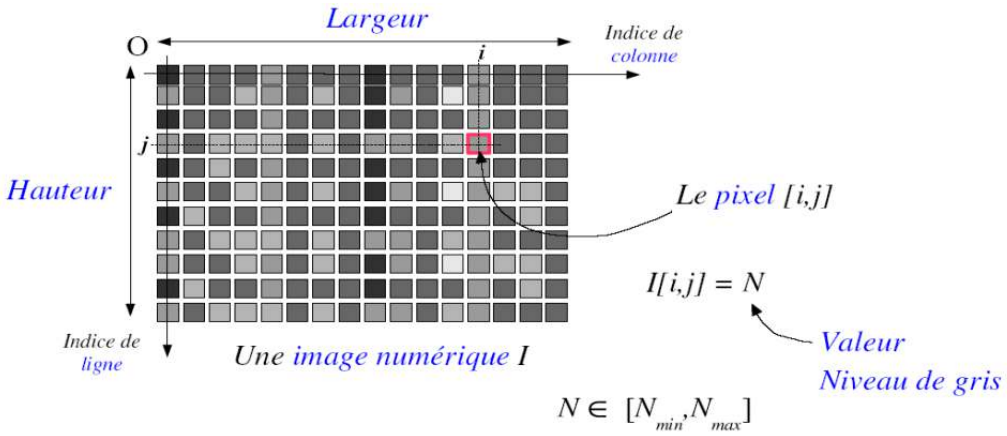


Figure 62: Une matrice de pixels.

Voici un exemple sur la représentation d'une image sous forme d'une matrice ou ses éléments sont les degrés de gris de chaque pixel :

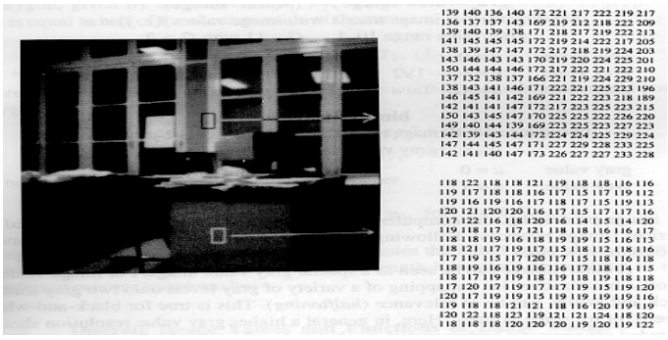


Figure 63: Représentation d'une image par une matrice de pixels.

Les traitements d'images fournis par OpenCV :

- **Le Thresh-holding (seuillage)** : d'une image permet de définir une valeur de pixel (correspondant à une couleur) qui servira de seuil. Au-dessus ou en dessous de cette valeur (selon l'algorithme), tous les pixels se verront assigner une autre valeur. Il existe plusieurs algorithmes de Thresh-holding [34] comme le **Adaptive Thresh-holding**, **Dynamic Thresh-holding**, … etc. Ces algorithmes se différencient selon le type de seuil : fixe ou adaptative. Voici quelques types de seuillage :

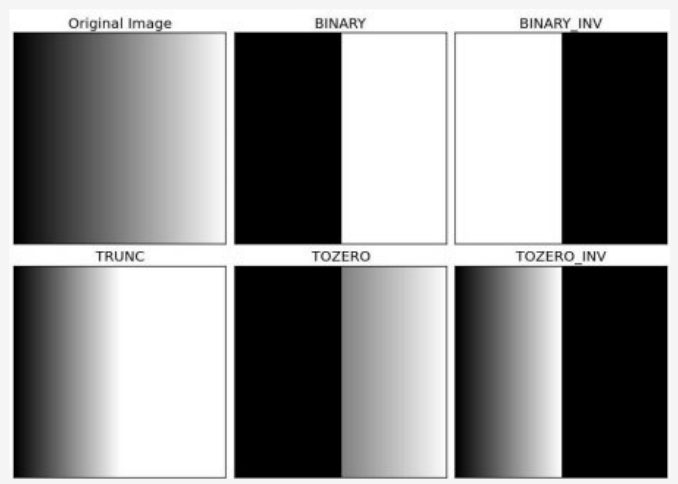


Figure 64: Différentes Types de Seuillage.

- **Les filtres de détection de contours** : d'un objet sur une image (comme le Filtre de Canny). Les contours sont l'une des deux méthodes de segmentation d'images.

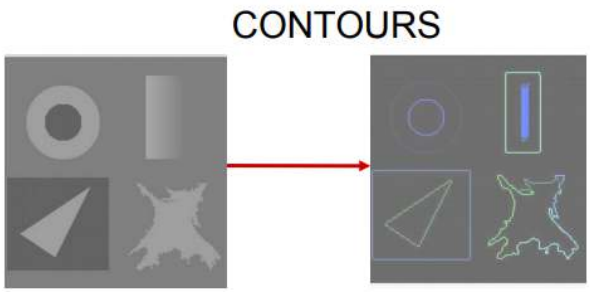


Figure 65: Différentes Types de Contours.

Les algorithmes de détection de contours sont faits à base de produit de convolution, ou la matrice représentant l'image est multipliée par un filtre pour produire une nouvelle image qui illustre les objets présents dans l'image originale comme des contours. Voici un exemple :

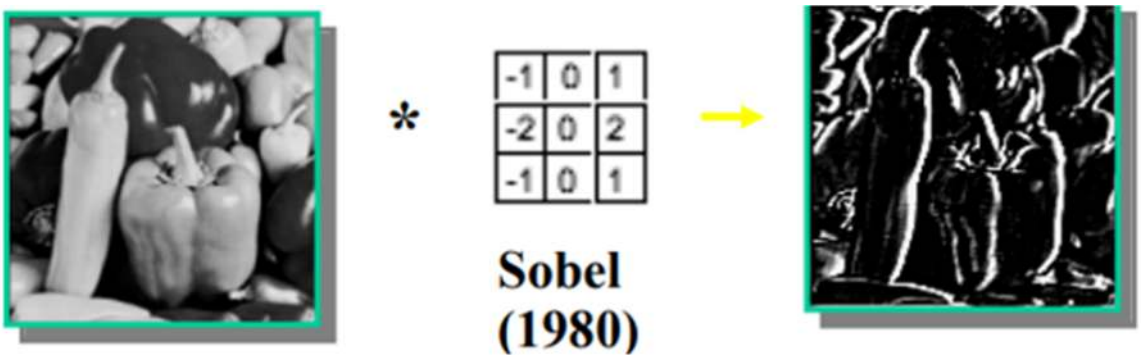


Figure 66: Multiplication d'une image par un filtre pour la détection des objets.

- **Les filtres de lissage** : permettant de réduire le bruit d'une image. Parmi eux, nous avons notamment les filtres par convolution, se basant sur les pixels voisins pour faire une moyenne.

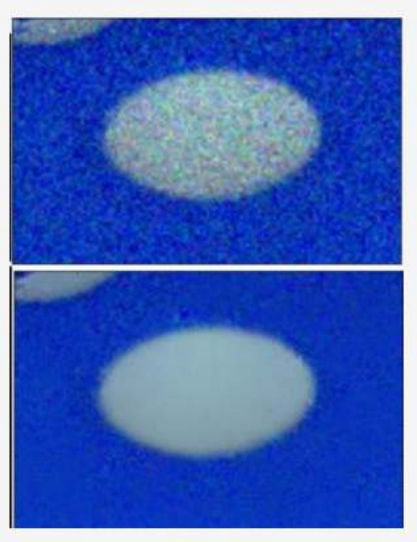


Figure 67: Exemple du résultat d'un filtre de lissage.

- **Les filtres morphologiques** : permettant, sur une image binaire (uniquement noire et blanche avec le fond en noir et l'objet sur lequel on effectue le traitement en blanc) d'appliquer une érosion ou dilatation, c'est-à-dire affiner l'objet (érosion) ou l'élargir (dilatation).

Figure 68: Résultat de l'érosion et la dilatation d'une image.

Il est possible de combiner l'érosion et la dilatation pour la suppression de bruit dans une image. L'application de l'érosion puis dilatation est appelé Opening (Ouverture) et l'application de la dilatation puis de l'érosion est appelé Closing (Fermeture).

Figure 69: Application du filtre de l'ouverture pour la suppression du bruit.

Figure 70: Application du filtre de la fermeture pour la suppression du bruit.

3.9 Gazebo :

Figure 71: Logo du Gazebo.

Gazebo est un simulateur 3D, cinématique, dynamique et multi-robot permettant de simuler des robots articulés dans des environnements complexes, intérieurs ou extérieurs, réalistes et en trois dimensions. Il s'agit d'un programme open source distribué sous licence Apache 2.0, utilisé dans les domaines de la recherche en robotique et en intelligence artificielle. Il permet de faire des simulations réalistes de la physique des corps rigides. Les robots peuvent interagir avec le monde (ils peuvent ramasser et pousser des objets, rouler et glisser sur le sol) et inversement (ils sont affectés par la gravité et peuvent se heurter à des obstacles dans le monde). Pour ce faire, Gazebo utilise de multiples moteurs physiques tels qu'Open Dynamics Engine (ODE) ou Bullet.

Il existe la possibilité de développer et de simuler ses propres modèles de robot et même de les charger au moment de l'exécution et pour cela il faut d'abord construire le modèle URDF (Universal Robot Description File). De plus, il est possible de créer des scénarios de simulation (mondes), en modifiant les caractéristiques des contacts avec le sol, des obstacles et même

des valeurs de gravité dans les trois dimensions. Il est également possible de faire varier les caractéristiques de contact de chaque lien individuellement. Gazebo contient divers plug-ins pour ajouter des capteurs au modèle du robot et les simuler, tels que des capteurs d'odométrie (GPS et IMU), de force, de contact, de laser et des caméras stéréos [30].

On commande le robot à l'aide de MoveIt ! et RViz et on visualise le résultat sur Gazebo.

Voici un exemple de simulation de notre robot ou on a créé un monde qui contient une table, un support pour le robot, une balle qui sera pris par le robot ainsi qu'une caméra qui détecte les objets selon leurs formes et couleurs :

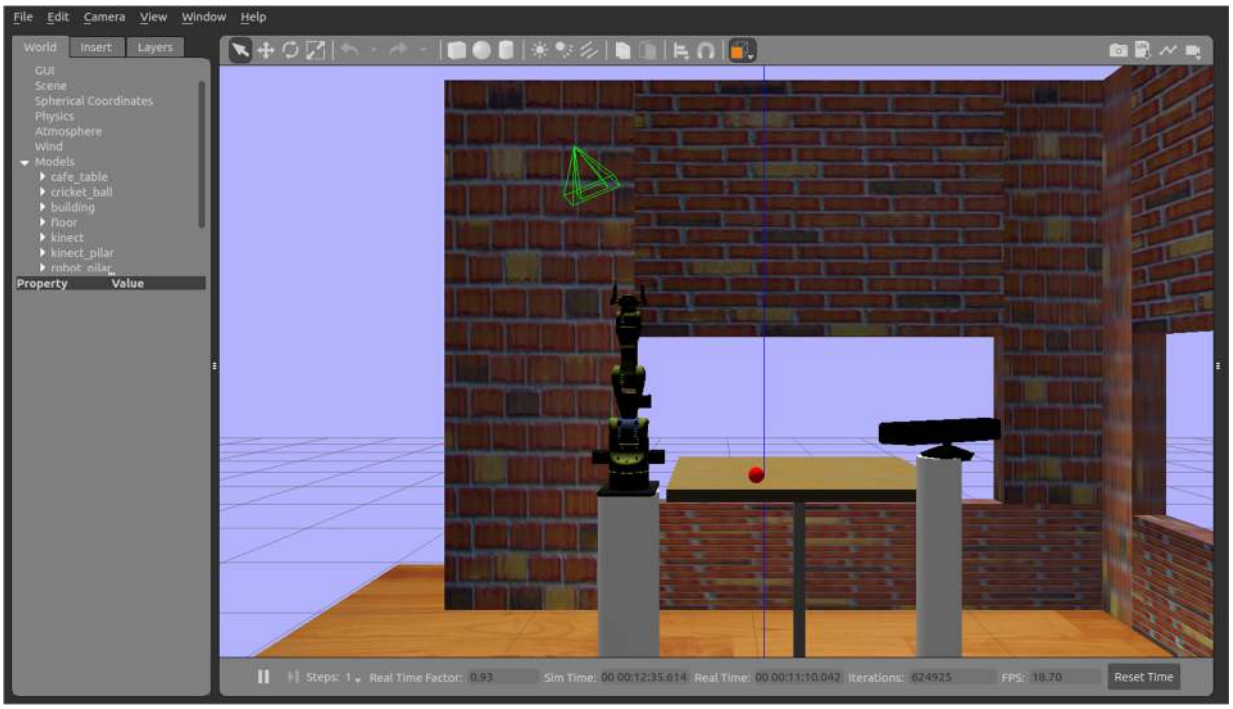


Figure 72: Simulation de l'application pick & place avec le bras Moveo dans le simulateur Gazebo.

3.10 Conclusion :

Dans ce chapitre, nous avons présenté l'intégralité des logiciels utilisés lors de l'élaboration de ce travail, on a expliqué leurs principes de fonctionnement et on a leurs choix.

Chapitre 4

La commande du robot, la détection des objets et l'exécution des trajectoires selon leurs classes.

4 La commande du robot, la détection des objets et l'exécution des trajectoires prédéfinies :

4.1 Introduction :

Dans ce chapitre nous allons détailler les étapes suivies pour commander le robot et réaliser la tâche d'intelligence à partir des commandes dans le terminal. Les commandes servent à exécuter les scripts et les programmes qui assurent la commande et le fonctionnement du bras de robot .

4.2 La commande du robot :

Le robot est commandé en utilisant ROS et ses outils (Rviz, Moveit! , rosserial, … ect) . On détaille les étapes pour faire ça :

4.2.1 Création des différents packages et nœuds nécessaires pour la commande du robot :

On les met ensemble dans le même espace de travail « catkin_ws ». On les place dans le dossier ' src ' (source) :

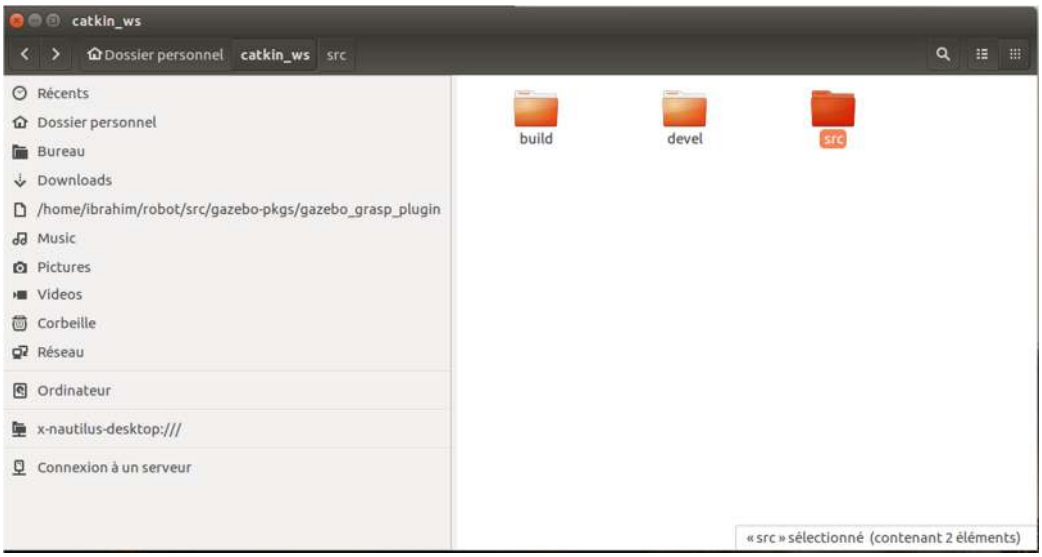


Figure 73: L'espace de travail « catkin_ws ».

On a créé un dossier dans '**src**' et on l'a nommé '**robot**', et on place les packages qu'on a besoin et les scripts qu'on utilise dedans :

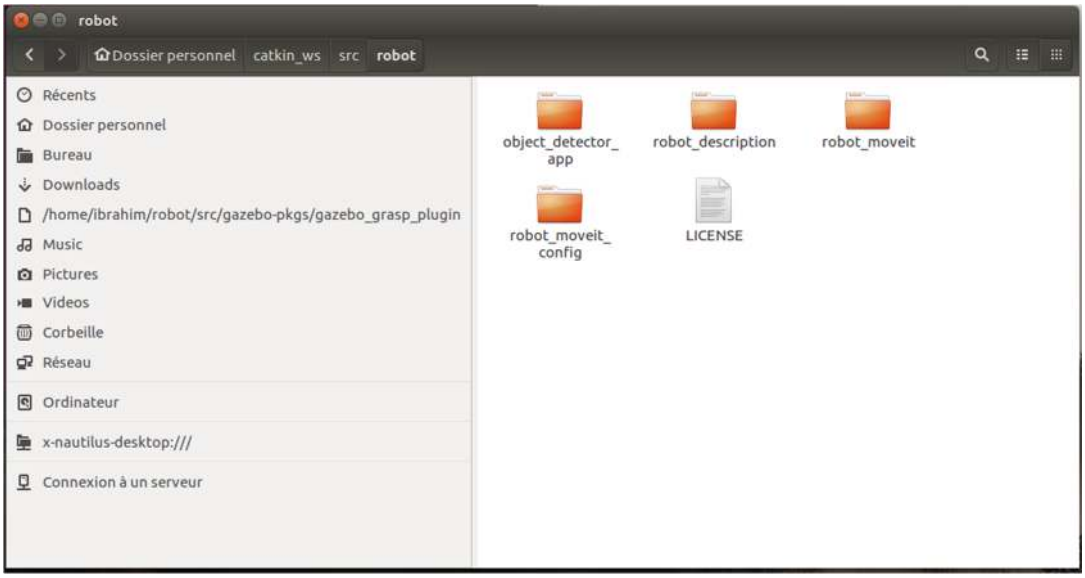


Figure 74: Le dossier « robot ».

- Le dossier 'robot_description' contient la description du robot : l'URDF et les meshes du robot (les fichiers STL de chaque partie du robot) :

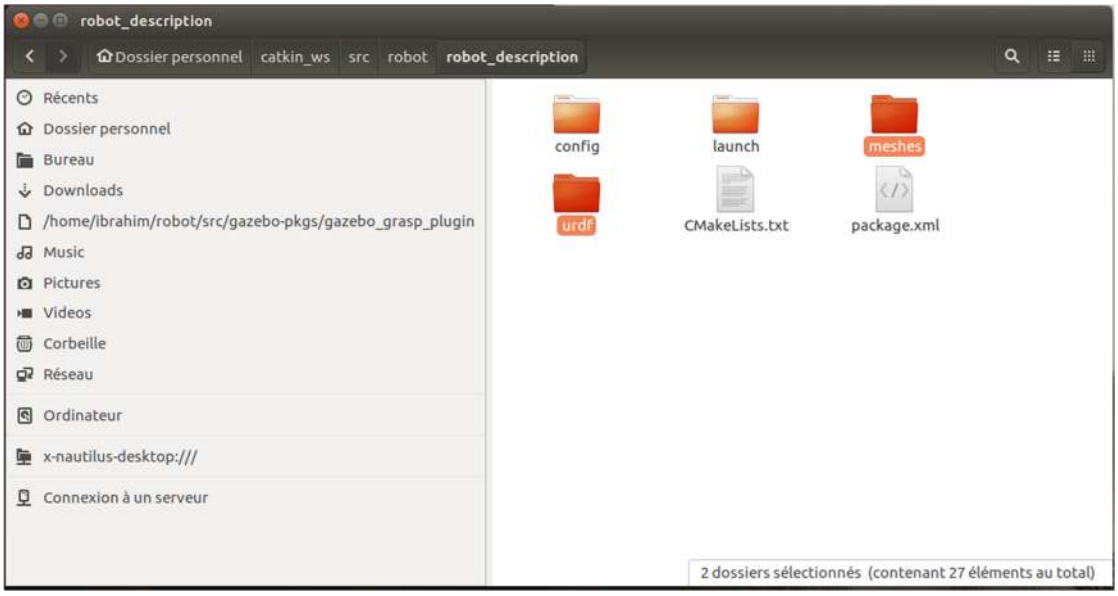


Figure 75: Le dossier « robot_description ».

- Le dossier ‘robot_moveit’ contient tous les scripts et packages qu’on utilise pour commander le robot :

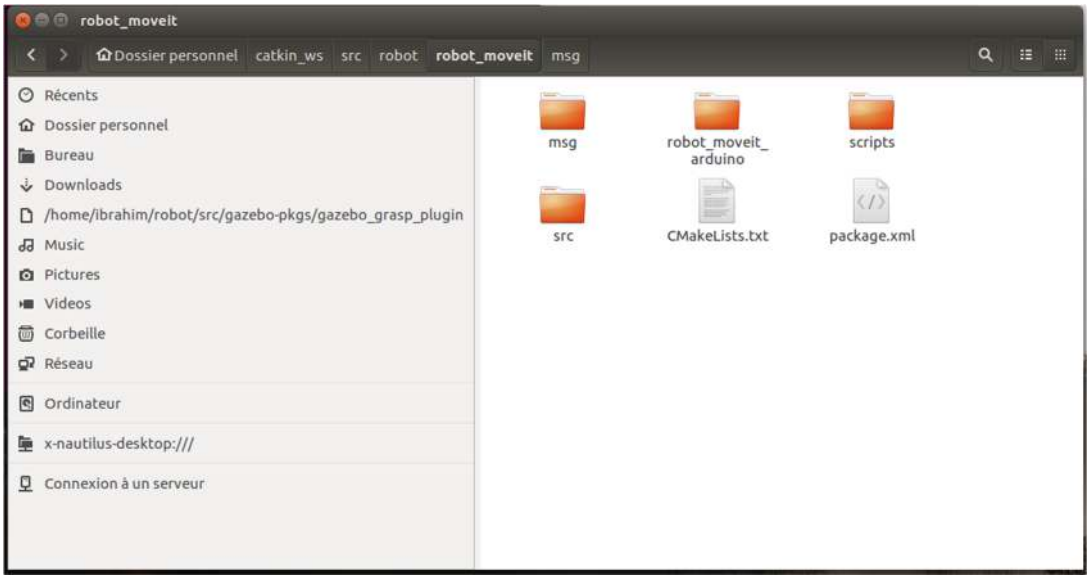


Figure 76: Le dossier « robot_moveit ».

- Le dossier ‘robot_moveit_arduino’ contient le script Arduino nommé ‘moveo_moveit_arduino.ino’ qu’on téléverse à la carte afin de commander le robot :

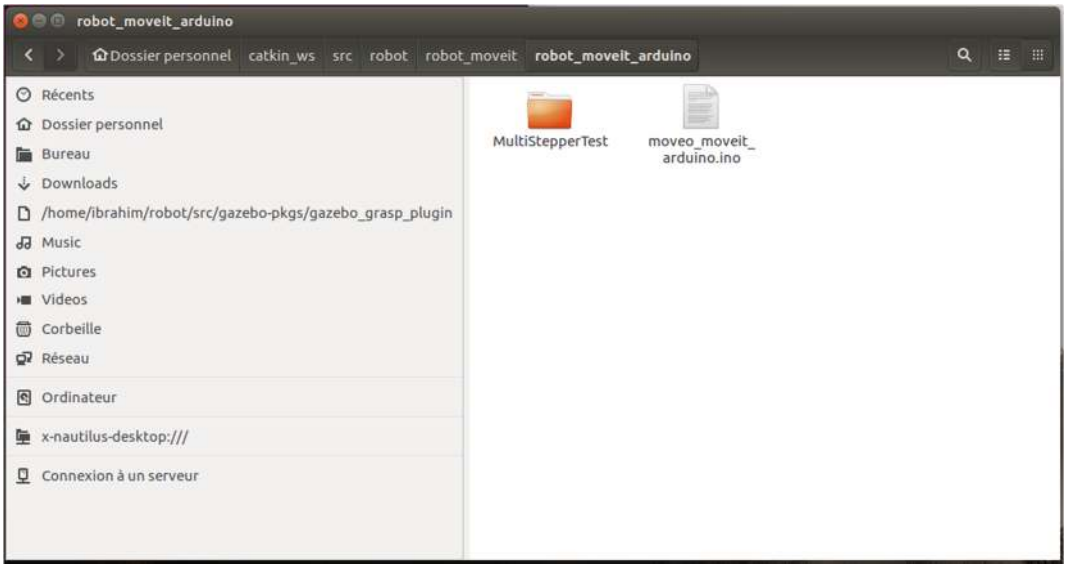


Figure 77: Le dossier « robot_moveit_arduino ».

- Le dossier ‘MultiStepperTest’ contient un script Arduino qu’on utilise pour faire le microstepping des moteurs afin de savoir le nombre de pas par révolution de chaque moteur. Ce nombre est nécessaire pour la commande du robot , car tout mouvement de ce dernier est l’exécution des rotations par pas par l’ensemble des moteurs .

- Le dossier ‘msg’ contient les messages (données) qui circulent entre la carte Arduino et le PC :

  Ces données sont les états des articulations (moteurs): le nombre de micro-pas fait par chaque moteur. On les utilise comme un feedback pour assurer si le robot a suivi la trajectoire prédéfinie ou non.

- Le dossier ‘src’ contient les scripts C++ :

  - ‘move_groupe_interface’ : qui assure la liaison entre l’interface ‘Move group’ et la carte Arduino . L’interface Move group fournit des fonctionnalités faciles à utiliser pour la plupart des opérations qu'un utilisateur souhaite effectuer, notamment la définition d'objectifs d'articulation ou de pose, la création de plans de mouvement, le déplacement du robot, l'ajout d'objets dans l'environnement et la fixation/détachement d'objets du robot. Cette interface communique via des topics, des services et des actions ROS avec le nœud MoveGroup .
  - ‘moveit_convert’ : qu’on utilise pour imiter les mouvements du robot dans Moveit ! et Rviz . Avec ce script on fait la synchronisation du mouvement réel du robot et la simulation dans Moveit ! .

- Le dossier ‘robot_moveit_config’ contient les packages qui créent la relation entre Moveit ! et la carte Arduino :

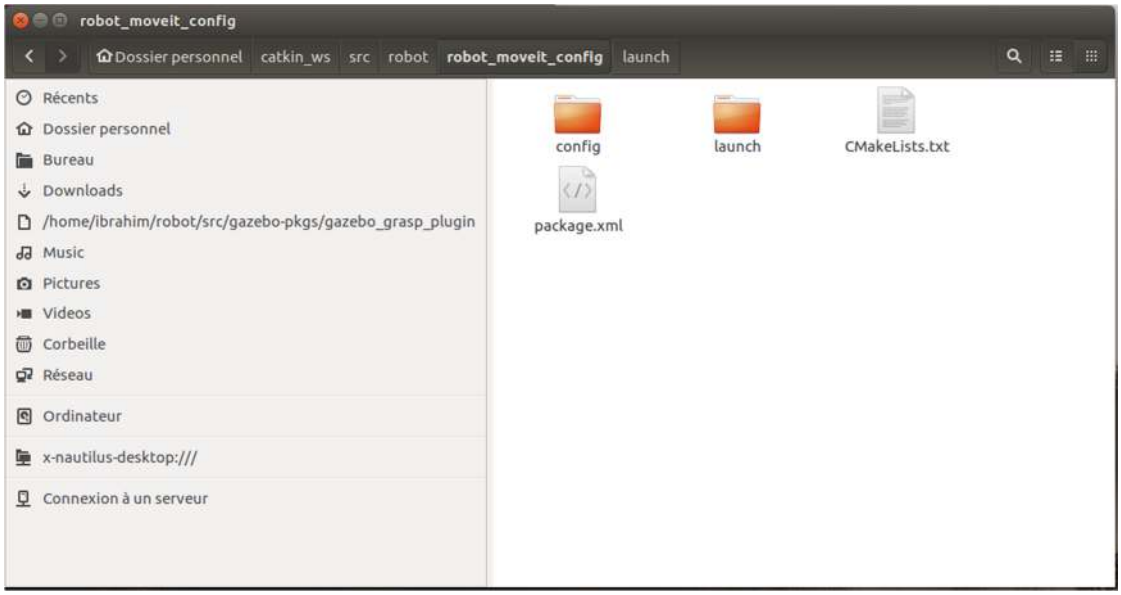


Figure 78: Le dossier « robot_moveit_config ».

- Le dossier 'object_detector_app' contient les scripts python qu'on utilise pour détecter les objets et les classifier avec un classificateur entrainé. La création de ce classificateur est détaillée dans la deuxième partie de ce chapitre.

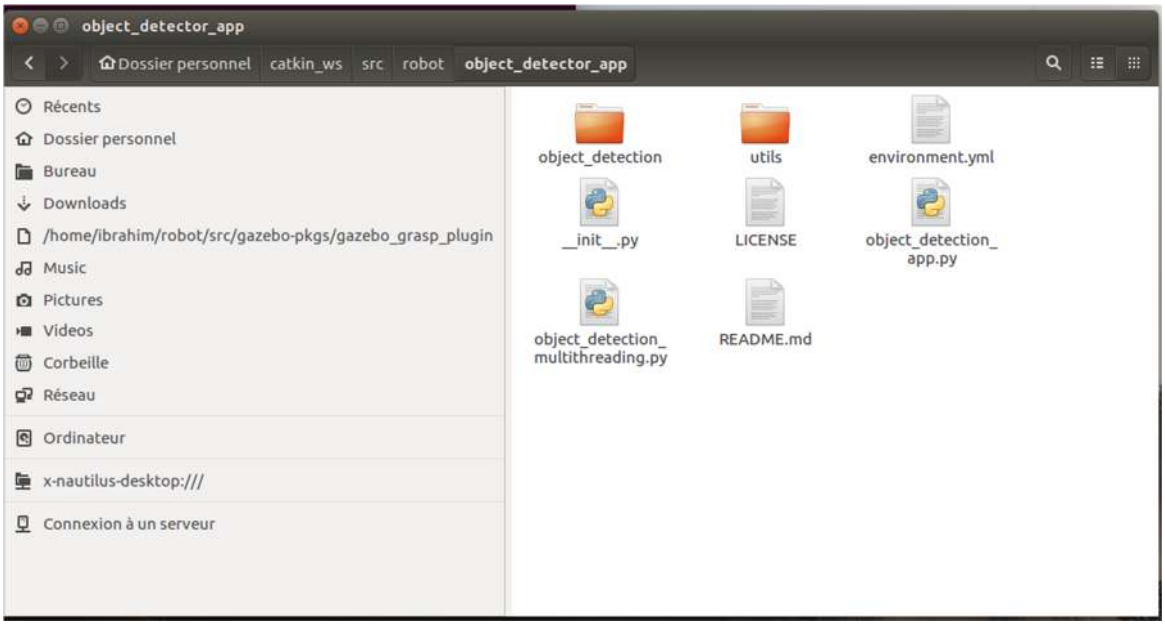


Figure 79: Le dossier « object_detector_app».

- Le dossier 'object_detection' contient le classificateur entrainé avec L'API de Tensorflow :

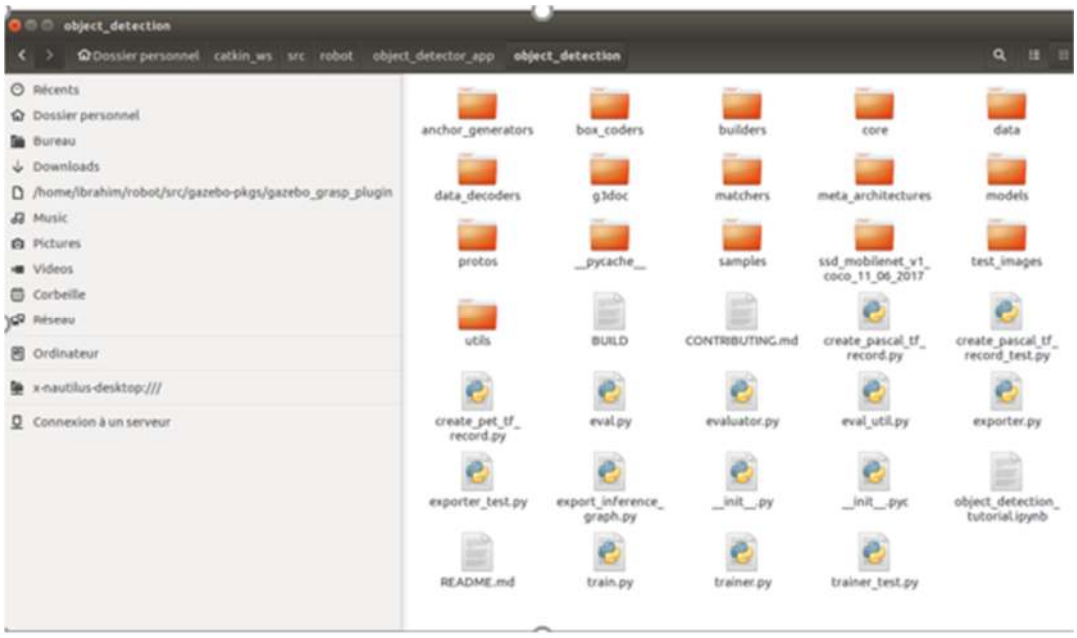


Figure 80: Le dossier « object_detection».

- Les scripts python ‘object_detection’ et ‘object_detection_multithreading’ sont les plus importants. On les utilise pour détecter les objets et les classifier et après ça communiquer avec la carte Arduino pour lui donner les commandes représentants les trajectoires prédéfinies.

Figure 81: Les scripts « object_detection» et « object_detection_multithreading ».

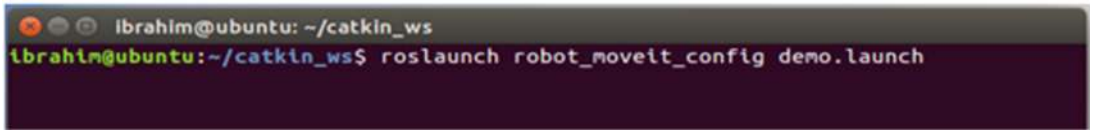
4.2.2 Exécution des commandes depuis la terminale :

On va citer les commandes utilisées et leurs résultats :

Au début, on accède à l’espace de travail ‘catkin_ws’ :

On exécute la commande ‘catkin_make’ pour activer tous les packages dans l’espace de travail.

On lance Move Group , Rviz et Moveit ! en exécutant la commande ‘ roslaunch robot_moveit_config demo.launch’ . Le fichier de lancement ‘demo.launch’ situant dans le dossier ‘robot_moveit_config’ est généré par le ‘ Moveit Setup Assistant’ et il lance Move Group , Rviz et Moveit ! afin qu’on peut manipuler le robot .



Voici le résultat d'exécution de cette commande :

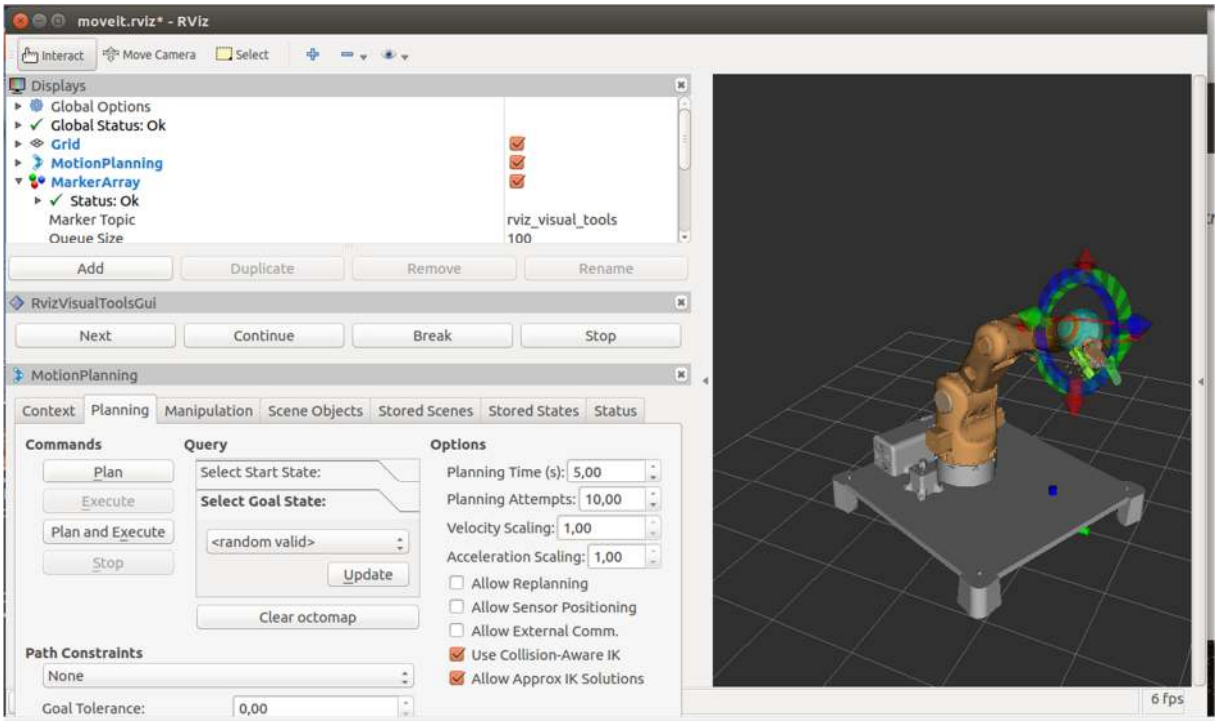


Figure 82: Le résultat de l'éxecution.

On peut maintenant manipuler le bras en choisissant une trajectoire prédéfinie avec le Moveit Setup Assistant ou bien en bougeant la balle bleue située sur le pince du robot. On peut aussi utiliser le 'rqt_gui' pour manipuler les articulations séparément.

Apré ça, on téléverse le script 'moveo_moveit_arduino.ino' à la carte Arduino, après qu'on choisit le port et le type de la carte , dans notre cas , le port est /ttyACM0 et la carte est Arduino Mega 2560 .

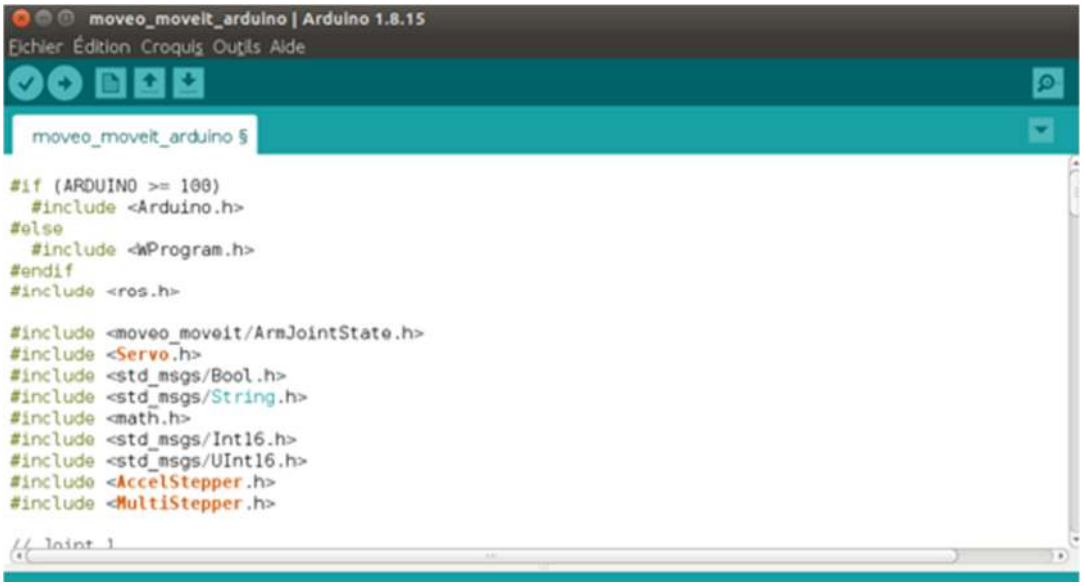


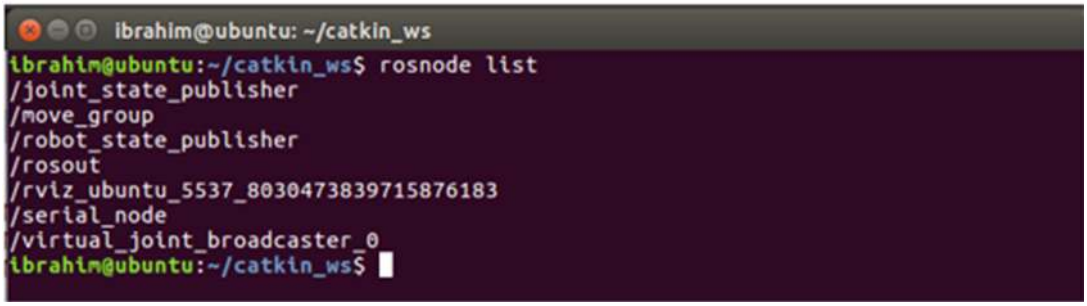
Figure 83: Le résultat de l'éxecution.

Maintenant, on exécute la commande 'rosrun rosserial_python serial_node.py /dev/ttyACM0' qui assure la liaison entre ROS et la carte Arduino moyennant le port ACM0. La carte Arduino devienne un nœud de point de vue ROS.

Voici le résultat de cette commande :

On voit que la carte Arduino est devenue un nœud de type subscriber qui s'abonne aux deux nœuds : 'joint_steps' et 'gripper_angle' . Voici la liste des

nœuds actifs après l'exécution de cette commande : 'joint_state_publisher' et 'robot_state_publisher' publient les données sur la carte Arduino . Ces données sont les pas des moteurs des articulations et l'angle du servomoteur qui assure l'ouverture et la fermeture du pince.



On exécute la commande 'rosrun robot_moveit moveit_convert' :

Cela créera la relation entre le robot en simulation (Rviz) et le robot réel. Maintenant tout ce qu'on fait dans Rviz se traduit en mouvement réel.

On peut aussi manipuler le bras en exécutant la commande 'rostopic pub joint_steps robot_moveit/ArmJointState joint1 joint2 joint3 joint4 joint5 0'

Ou on remplace 'joint1 joint2 …' avec le nombre de microsteps que le moteur de l'articulation doit faire. Avec cette commande on publie des données sur le nœud 'joint_steps'.

Et de même pour la pince, on peut contrôler l'angle d'ouverture et fermeture de ce dernier à partir du Terminal avec la commande 'rostopic pub gripper_angle std_msgs/UInt16 angle' ou on remplace 'angle' par l'angle qu'on veut (l'angle doit être dans l'intervalle 0 – 180) . Avec cette commande on publie des données sur le nœud 'gripper_angle'.

Voici tous les nœuds et topics actifs lors de la commande du robot :

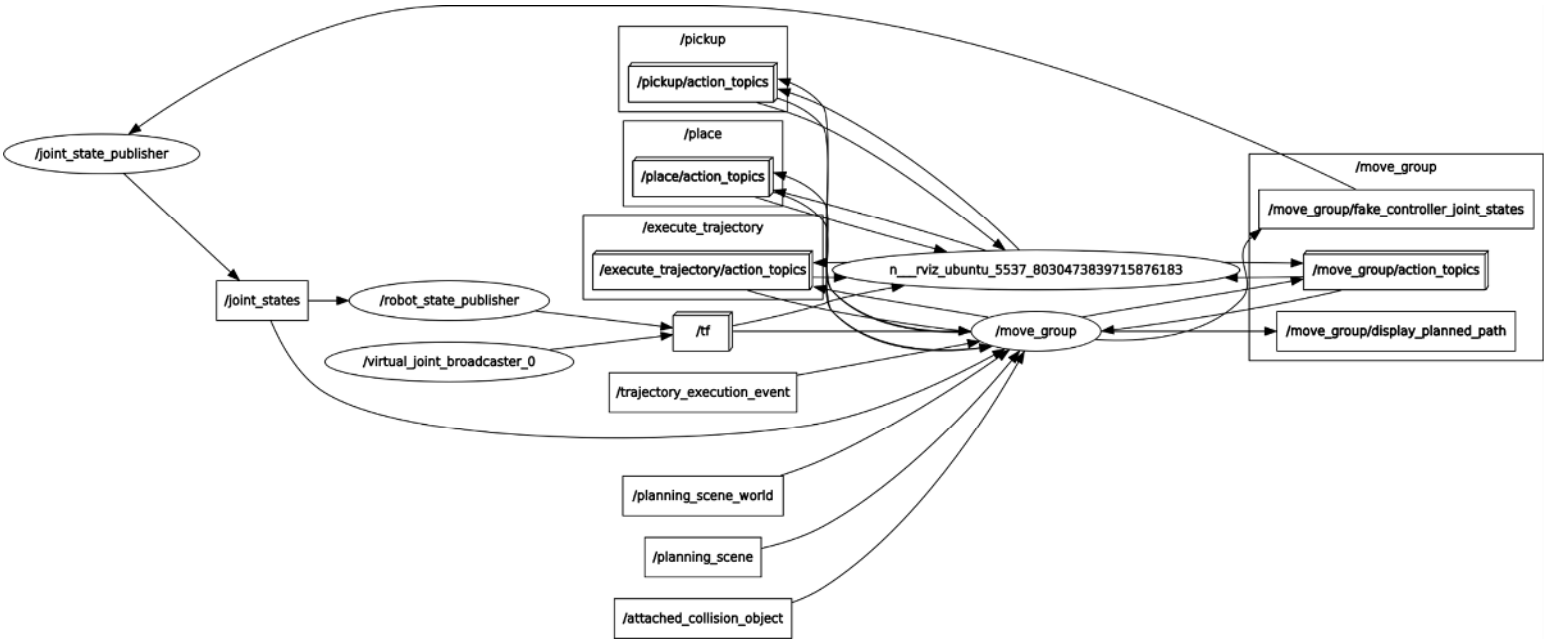


Figure 84: Les nœuds actifs.

4.3 Le processus d'entrainement des modèles avec L'API de TensorFlow :

On va commencer en détaillant le processus d'entrainement des modèles avec L'API de TensorFlow utilisé pour le développement de l'application de détection des objets captés par la webcam.

L'entrainement sera fait sur une base de données qui a une relation avec l'objet (voir les objets) qu'on veut que la webcam capte et qu'on doit classifier. On peut créer notre propre base de données (images) en prenant plusieurs photos de l'objet (le nombre de photo le plus grand possible le mieux) qu'on veut détecter et classer dans des différentes situations et avec différentes poses, et on peut aussi utiliser des bases de données existantes et valables sur Internet.

Les étapes d'entrainement sont les suivantes :

4.3.1 Installation du Anaconda [35], CUDA [36] et cuDNN [37] :

Anaconda est un logiciel qui sert à la création des environnements virtuels opérants sur Python afin de pouvoir installer et utiliser des bibliothèques Python sans se soucier des différences des versions installées existantes qui peuvent créer des conflits lors de l'exécution. Anaconda installe automatiquement les versions CUDA et cuDNN dont on a besoin pour utiliser TensorFlow sur un GPU. On doit assurer la compatibilité entre TensorFlow, CUDA et cuDNN [38].

Figure 85: Anaconda [35], CUDA [36] et cuDNN [37].

4.3.2 Configuration de TensorFlow et l'environnement virtuel de Anaconda :

L'API de détection des objets **TensorFlow** nécessite l'utilisation de la structure spécifique fournie dans son répertoire GitHub. Il nécessite également plusieurs paquets Python supplémentaires, des ajouts spécifiques aux variables PATH [39] et PYTHONPATH ainsi que quelques commandes supplémentaires de configuration pour exécuter ou entraîner un modèle de détection d'objets.

Pour commencer le processus de configuration, on télécharge et installe l'API de détection d'objets TensorFlow depuis son répertoire GitHub [40]. Ce répertoire de travail contiendra l'intégralité des programmes de détection d'objets intégrale au fonctionnement du TensorFlow, ainsi que les images et les données d'entraînement et les classificateurs entraînés, les fichiers de configuration et tout ce qui est nécessaire pour le classificateur de détection d'objets.

TensorFlow fournit plusieurs modèles de détection d'objets (classificateurs pré-entraînés avec des architectures de réseaux de neurones spécifiques) [41]. Certains modèles comme le modèle SSD-MobileNet ont une architecture qui permet une détection plus rapide mais avec moins de précision, alors que d'autres modèles, comme le modèle Faster-RCNN, donnent une détection plus lente mais avec plus de précision.

Si on veut utiliser le détecteur d'objets sur un appareil à faible puissance de calcul (comme un smartphone ou un Raspberry Pi), il est préférable d'utiliser le modèle SDD-MobileNet, et si on veut utiliser un ordinateur portable ou un ordinateur de bureau correctement alimenté, on peut utiliser l'un des modèles RCNN.

Après l'installation du TensorFlow, on doit avoir un dossier qui rassemble à la figure suivante :

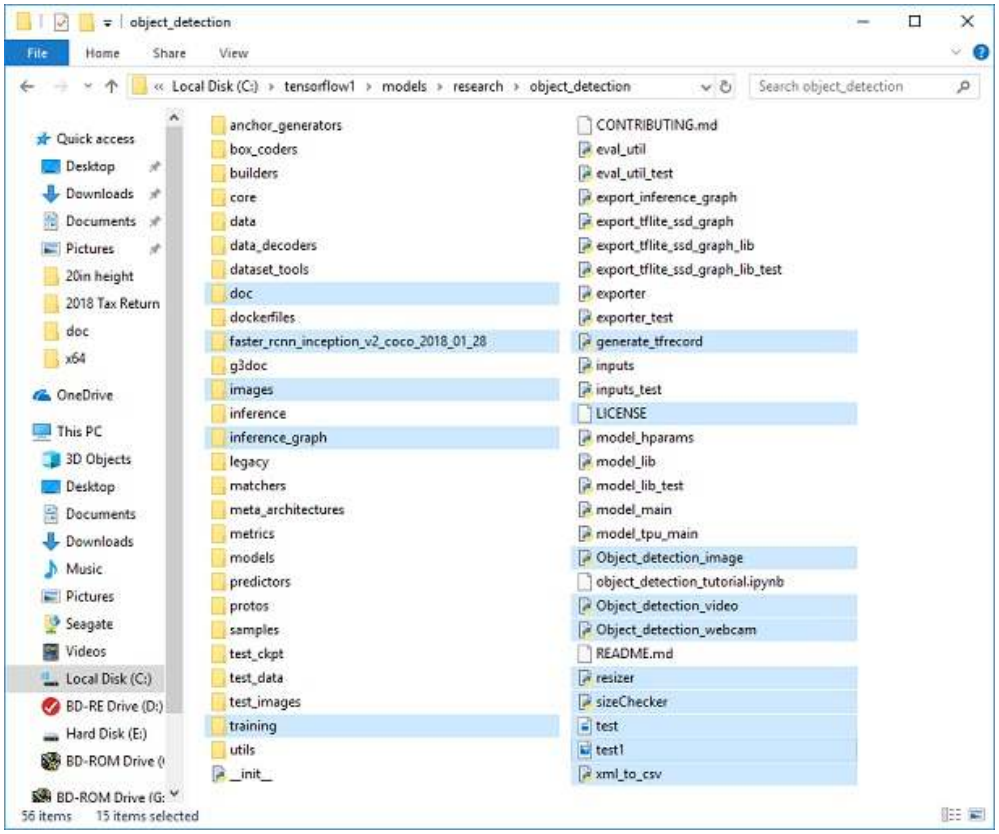


Figure 86: le répertoire d'installation de TensorFlow.

Ce répertoire contient les images, les données d'annotation, les fichiers .csv et les TFRecords nécessaires pour l'entraînement d'un détecteur. Il contient également des scripts Python qui sont utilisés pour la génération des données d'entraînement. Il dispose également de scripts pour tester le classificateur de détection d'objets sur des images, des vidéos ou un Stream de webcam.

Pour former notre propre détecteur d'objets, on doit suivre les étapes suivantes :

4.3.3 Configuration de l'environnement virtuel de Anaconda :

Avant la configuration d'un environnement virtuel dans Anaconda pour tensorflow-gpu, on doit d'abord créer cette nouvel environnement virtuel en tapant la commande suivante :

C:\> conda create -n tensorflow1 pip python=3.5

Puis, on doit activer cette environnement en tapant la commande :

C:\> activate tensorflow1

(tensorflow1) C:\>python -m pip install --upgrade pip

En suite, on doit installer tensorflow-gpu dans cette nouvelle environnement, en tapant :

(tensorflow1) C:\> pip install --ignore-installed --upgrade tensorflow-gpu

On peut également utiliser la version CPU uniquement de TensorFlow, mais cette version est beaucoup plus lente car elle n'utilise pas une carte graphique dédiée et doit compter seulement sur les calculs du processeur.

4.3.4 Configuration de la variable d'environnement PYTHONPATH :

Une variable PYTHONPATH doit être créée pour pointer vers les répertoires **\models, \models\research** et **\models\research\slim**. Pour ce faire, on exécute les commandes suivantes :

(tensorflow1) C:\> set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim

Ensuite, on doit compiler les fichiers Protobuf, qui sont utilisés par TensorFlow pour configurer les paramètres de modèle et d'entraînement.

Enfin, on exécute les commandes suivantes :

(tensorflow1) C:\tensorflow1\models\research> python setup.py build

(tensorflow1) C:\tensorflow1\models\research> python setup.py install

L'API de détection d'objets TensorFlow est maintenant entièrement configurée pour l'utilisation des modèles pré-entraînés pour la détection d'objets, ou pour la création d'un nouveau modèle.

4.3.5 Rassemblement et étiquetage des images :

Maintenant que l'API de détection d'objets TensorFlow est entièrement configurée et prêt à fonctionner, on doit fournir les images qu'il va utiliser pour créer un nouveau classificateur de détection :

4.3.6 Rassemblement des images :

TensorFlow a besoin de centaines d'images d'un objet pour pouvoir construire un bon classificateur de détection. Pour former un classificateur robuste, les images d'apprentissage doivent avoir des objets aléatoires dans le contenu de l'image avec les objets souhaités, et doivent avoir une variété d'arrière-plans et de conditions d'éclairage.

Une fois qu'on a toutes les images dont on a besoin, on doit déplacer 20% d'entre elles dans le répertoire **\object_detection\images\test**, et 80% d'entre elles dans le répertoire **\object_detection\images\train**. Et on doit assurer qu'il y a une variété d'images dans les répertoires **\test** et **\train**.

4.4 Etiquetage des images :

Avec toutes les images rassemblées, il est temps d'étiqueter les objets souhaités dans chaque image.

LabelImg est un excellent outil pour étiqueter les images, on peut télécharger et installer cet outil depuis son répertoire GitHub [42].

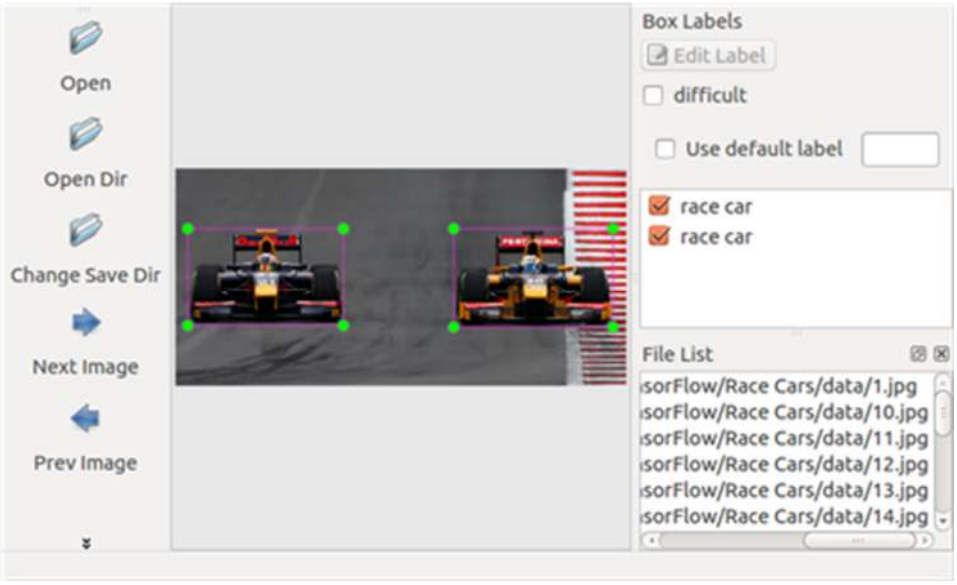


Figure 87: Exemple d'utilisation de Labe1Lmg.

LabelImg enregistre un fichier .xml contenant les données d'étiquetage pour chaque image. Ces fichiers .xml seront utilisés pour générer des TFRecords, qui sont l'une des entrées de l'entraîneur TensorFlow.

4.5 Génération des données d'entraînement :

Une fois les images étiquetées, il est temps de générer les TFRecords qui représentent les données d'entrée au modèle d'entraînement TensorFlow. Tout d'abord, les données .xml des images seront utilisées pour créer des fichiers .csv contenant toutes les données pour les images d'entraînement et de test. À partir du dossier **\object_detection**, on exécute la commande suivante dans Anaconda :

(tensorflow1) C:\tensorflow1\models\research\object_detection> python xml_to_csv.py

Cela crée un fichier **train_labels.csv** et **test_labels.csv** dans le dossier **\object_detection\images**.

Ensuite, on doit ouvrir le fichier **generate_tfrecord.py** dans un éditeur de texte et remplacer les étiquettes prédéfinies par nos propres étiquettes, où chaque objet se voit attribuer un numéro d'identification.

Ensuite, on génére les fichiers **TFRecord** en exécutant ces commandes à partir du dossier **\object_detection** :

python generate_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\train --output_path=train.record

python generate_tfrecord.py --csv_input=images\test_labels.csv --image_dir=images\test --output_path=test.record

Ceux-ci génèrent un fichier **train.record** et un fichier **test.record** dans ***\object_detection***. Ceux-ci seront utilisés pour construire le nouveau classificateur de détection d'objets.

4.6 Création des étiquettes et configuration de l'entrainement :

Les étiquettes :

Les étiquettes indiquent au trainer ce qu'est chaque objet en définissant une relation entre les noms des classes et les numéros d'identification des classes. Les numéros d'identification des étiquettes doivent être les mêmes que ceux définis dans le fichier **generate_tfrecord.py**.

Configuration de l'entrainement :

Enfin, le pipeline d'entraînement à la détection d'objets doit être configuré. Il définit quel modèle et quels paramètres seront utilisés pour l'entraînement. C'est la dernière étape avant le lancement de l'entraînement.

Il y a plusieurs modifications à apporter au fichier **.config**, principalement en changeant le nombre de classes et d'exemples, et en ajoutant les chemins de fichiers aux données d'apprentissage.

4.7 Lancement de l'entrainement :

On doit lancer la commande suivante pour commencer l'entraînement :

python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_pets.config

TensorFlow initialise l'entrainement. L'initialisation peut prendre jusqu'à 30 secondes avant le début de l'entraînement réel. Elle ressemblera à ceci :

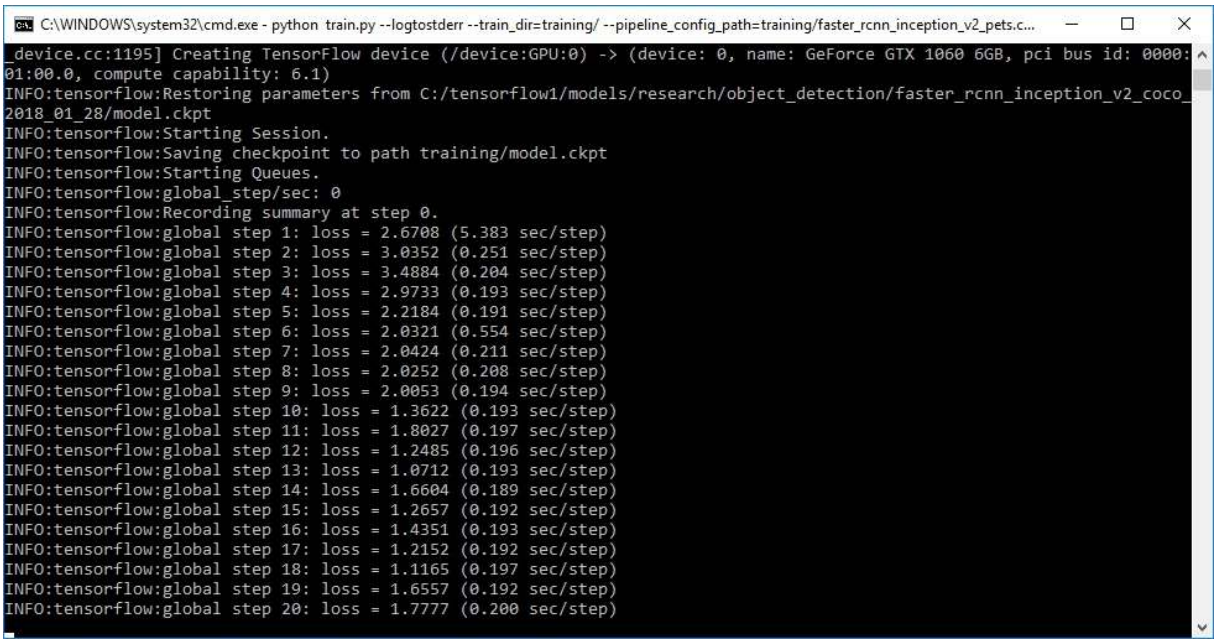


Figure 88: Lancement de l'entrainement.

Chaque étape de l'entrainement rapporte la perte. Cette perte diminuera de plus en plus au fur et à mesure que l'entraînement progresse.

On peut voir la progression de l'entrainement en utilisant **TensorBoard**. La page TensorBoard fournit des informations et des graphiques qui montrent la progression de l'entrainement. Un graphe très important est celui de la perte, qui montre la perte globale du classificateur au fil du temps.

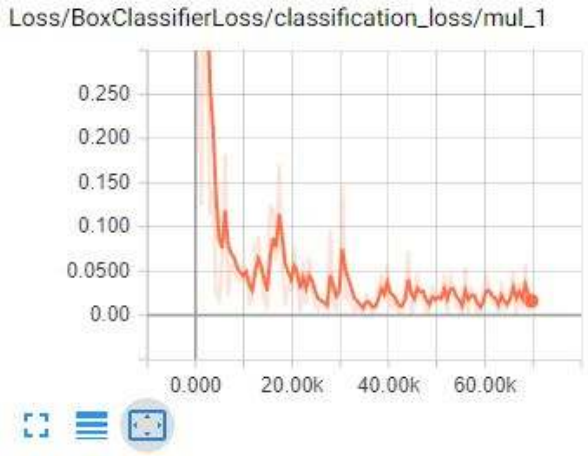


Figure 89: Graphe de la perte.

Le classificateur de détection d'objets est prêt à fonctionner maintenant.

4.8 Pick and Place spécifique à l'objet utilisant le bras BCN-3D Moveo :

Dans cette partie, on va parler de la procédure suivie pour réaliser le « pick and place » spécifique en utilisant le bras de robot BCN-3D Moveo, en expliquant le principe et les commandes utilisées :

Le nœud '**moveo_objrec_publisher'** reçoit l'étiquette de l'objet reconnu depuis le script **object_detection_multithreading.py** et ensuite publie une séquence de trajectoires sur le topic **joint_steps** pour pouvoir effectuer le « pick and place » spécifique à l'objet en question. La carte Arduino s'abonne au topic **joint_steps** et effectue les trajectoires lui données en envoyant les commandes nécessaires au moteurs pas-à-pas.

Afin de pouvoir utiliser le « pick and place » spécifique, on doit avoir accès à une reconnaissance d'objet en temps réel, pour réaliser ça, on utilise Python 3, L'API de détection d'objet de TensorFlow, OpvnCv et une webcam qui servira à prendre les images des objets à détecter et reconnaitre. La procédure à suivre pour la réalisation de cette tâche est la suivante :

On doit connecter la webcam au laptop, ainsi que la carte Arduino.

Téléverser le programme Arduino **moveo_moveit_arduino.ino** à la carte Arduino.

Créer l'environnement virtuel qui contient Python3, OpenCv, et TensorFlow.

Dans cet environnement virtuel, on doit exécuter la commande suivante : **python object_detection_multithreading.py** dans un terminal.

Dans un autre terminal, on doit exécuter la commande : **roscore.**

Dans un autre terminal, on doit exécuter la commande : **rosrun rosserial_python serial_node.py /dev/ttyACM0.** Cette commande sert à la création du nœud **rosserial** qui comminique avec la carte Arduino.

On doit définir les trajectoires qu'on souhaite que chaque objet suit, ce sont des trajectoires prédéfinies relatives à chaque objet.

Dans un autre terminal, on doit exécuter la commande : **rosrun moveo_moveit moveo_objrec_publisher.py.**

Lorsqu'un objet est placé dans le champ de vision de la caméra, une trajectoire sera effectuée en fonction de son classe.

4.9 Conclusion :

Dans ce chapitre nous avons détaillé les étapes suivies pour commander le robot et réaliser la tâche d'intelligence à partir des commandes dans le terminal.

Chapitre 5

Résultats, Avantages, Conclusion générale et perspectives.

5 Résultats, Avantages et perspectives :

5.1 Introduction :

Dans ce chapitre nous allons présenter les résultats que nous avons obtenu lors du travail sur ce projet. Nous allons citer quelques avantages de ce bras de robot et nous allons conclure et donner quelques idées qui peuvent améliorer et développer notre travail au futur.

5.2 Résultats :

Résultats de la construction et la commande du bras de robot :

On est parvenu à la réalisation de ce bras manipulateur, nous avons pu le construire et le câbler et le faire fonctionner, les figures suivantes montrent cette construction et démontrent notre réalisation réelle du bras :

D'abord, on a commencé par l'impression de toutes les pièces nécessaires pour la réalisation de ce projet, l'impression était faite avec une imprimante 3D de marque Creality Ender 5 Plus, avec du filament du type PLA.

Figure 90 : les pièces imprimés par l'imprimante 3D.

Ensuite, on est passé à l'étape de construction de toutes les articulations individuelles, afin de pouvoir réaliser l'assemblage de toutes les pièces mécaniques ensemble :

Figure 92: l'assemblage mécanique de toutes les articulations ensemble.

Après ça, on est passé au câblage du circuit de la commande du bras, en reliant les moteurs avec les drivers TB6560, et en reliant ces derniers avec la carte shield Ramps v1.4, cette dernière sera lui-même fixée sur la carte Arduino. Pour l'alimentation des différents composants, les dirvers seront alimentés par la source de tension en 24V, la carte shield Ramps v1.4 sera alimentée par la sortie du convertisseur DC-DC en 12V, et la carte Arduino sera connectée avec le PC via un câble USB qui assure son alimentation.

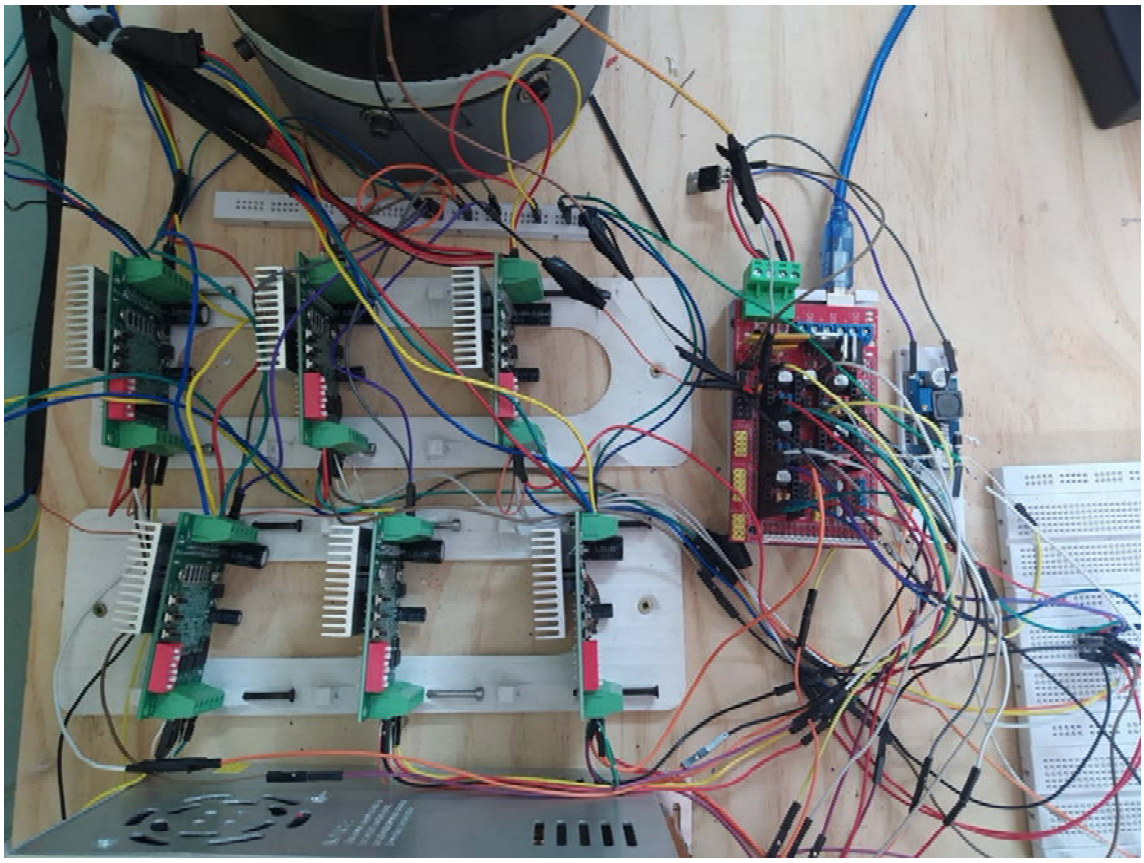


Figure 93: Câblage du Circuit de commande.

On est parvenu aussi à réaliser la commande du bras en manipulant les différents outils de ROS présentés dans les chapitres précédents. Les figures suivantes montrent la réalisation de cette commande :

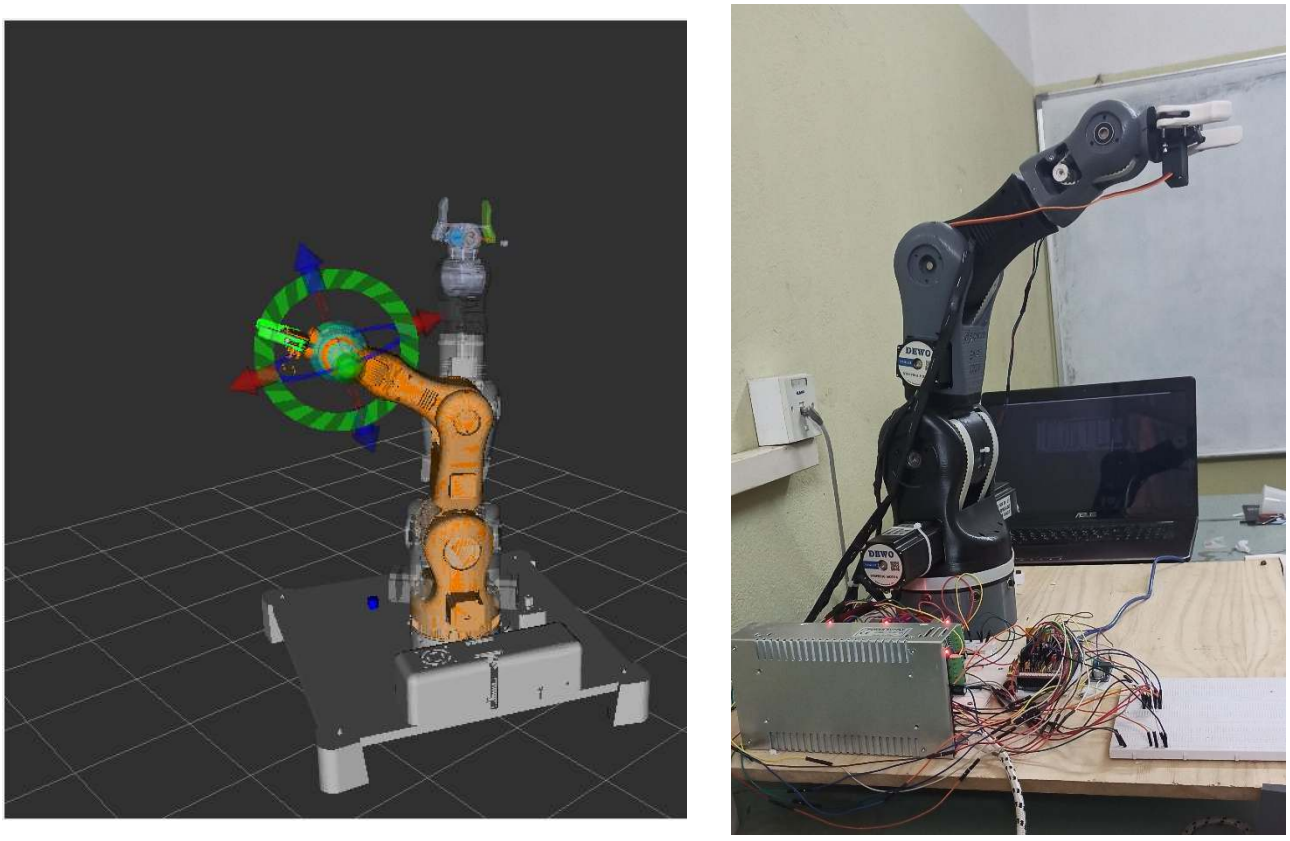


Figure 94 : La commande en temps réel du bras de robot avec Rviz et Moveit !

On est parvenu à la commande du bras de robot, et effectuer des taches de pick & place spécifique, mais on n'a pas pu réaliser la tache de pick & place intelligente à cause de l'absence d'une caméra compatible avec Linux.

Voici quelques applications de pick & place d'une bouteille en plastique :

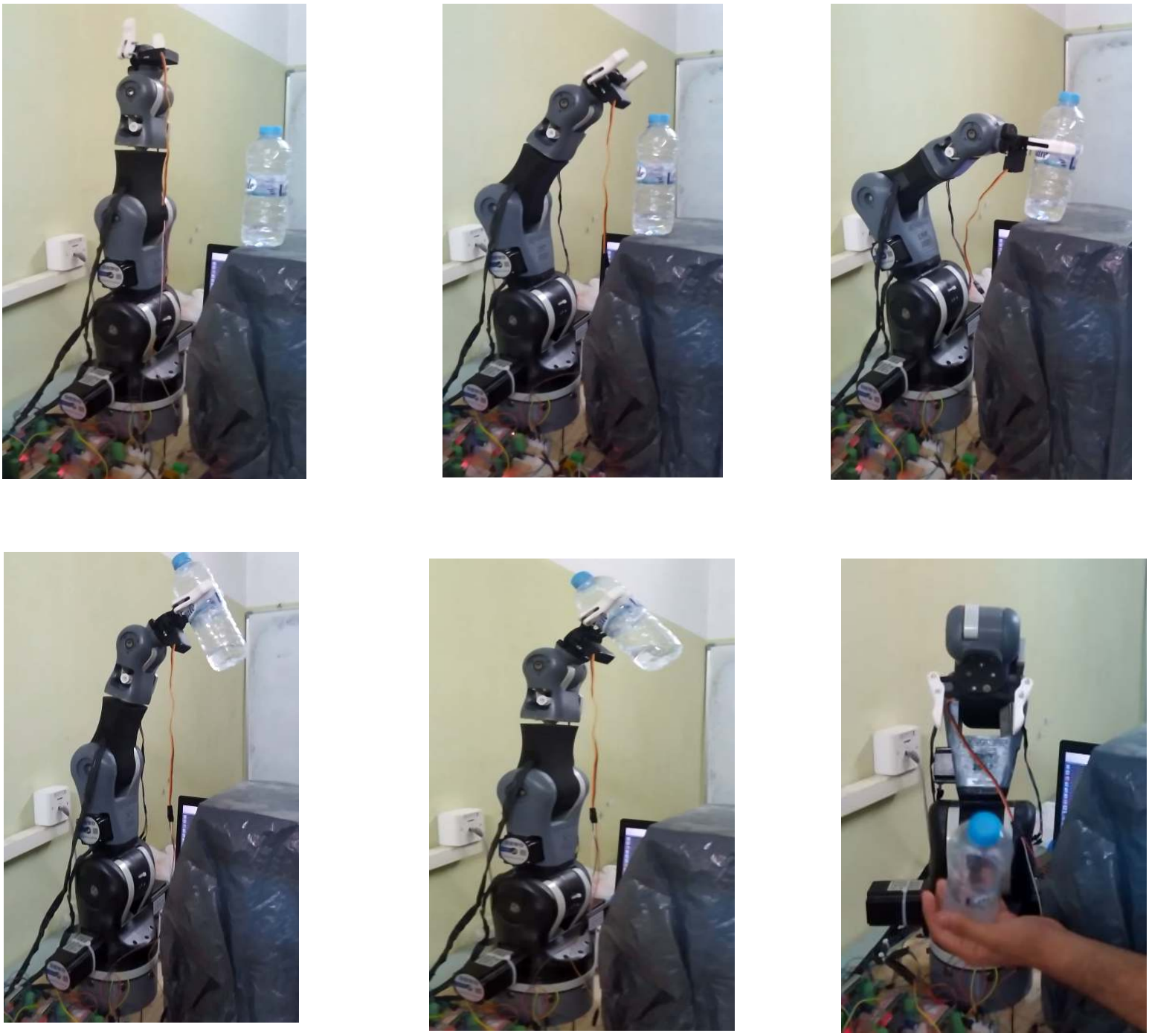


Figure 95: Quelques manipulations de Pick & Place spécifiques.

Résultats de la tâche de l'intelligence artificielle :

On n'a pas pu encore réaliser cette tâche en pratique. On a terminé les scripts mais on n'a pas pu les tester en réalité à cause du manque d'une webcam compatible avec ROS qui est à base de Linux.

On va essayer de terminer cette tâche une fois une webcam est disponible.

On peut résumer le processus du fonctionnement de cette tâche par le schéma suivant :

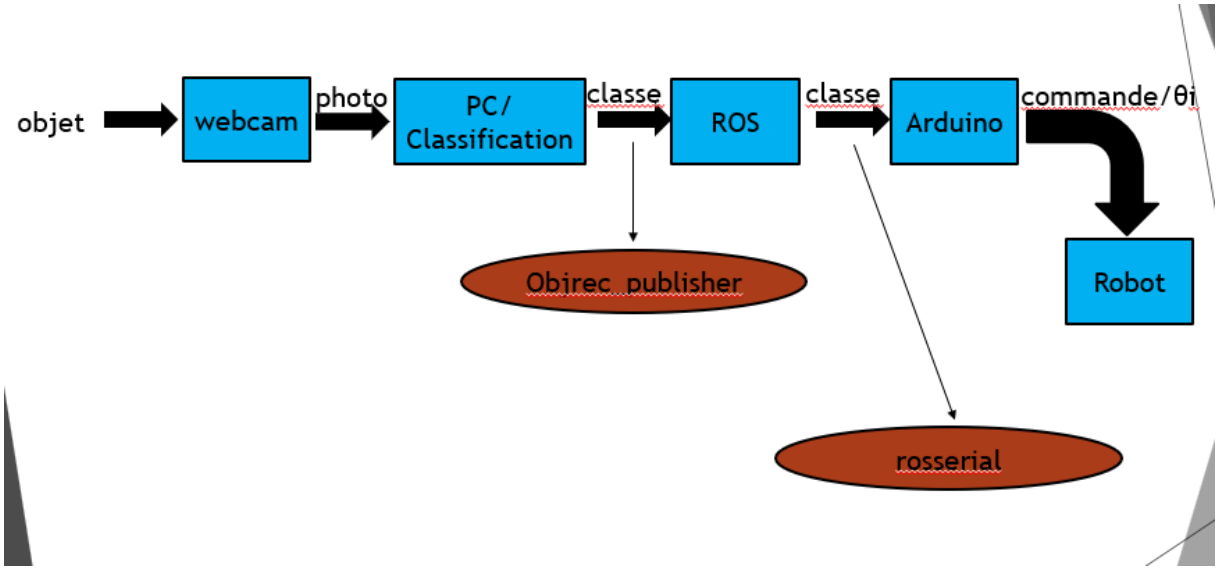


Figure 96: le schéma qui résume le processus.

5.3 Les Avantages du bras de robot :

L'avantage majeur de ce bras est la flexibilité quand il s'agit de son organe terminal, car la base sur laquelle il se fixe peut supporter plusieurs types d'organes terminaux (ventouses, Laser, système de soudure …etc.).

On peut aussi changer les moteurs utilisés et les remplacer par d'autres moteurs plus puissants pour permettre au robot de réaliser des tâches plus demandant.

Vu que le processeur essentiel de la commande du bras est la carte Arduino, qui est utilisé par une grande communauté et équipée de plusieurs modules électroniques externes qui assurent des fonctions très utiles pour commander le bras de robot, on peut par exemple utiliser des capteurs de son, des modules Bluetooth ou Wifi, des capteurs de fin de course, on peut aussi utiliser d'autres cartes Shield qui ont plus de fonctionnalités que le Ramps v1.4.

5.4 Conclusion générale et Perspectives :

Nous avons tenté à travers ce travail d'entrer à un domaine très vaste et qui est toujours en développement qui est celui de la combinaison entre la robotique classique et l'intelligence artificielle qui fait partie de l'automatique avancée.

Ce travail se divise en 03 parties :

La première partie concerne la modélisation et la construction du bras de robot.

La deuxième partie concerne la commande électrique du bras de robot à base de Microstepping.

La troisième partie concerne l'ajout de l'aspect d'intelligence au bras de robot.

Dans la perspective d'une continuité de ce travail, nous proposons :

- Le changement du capteur (la webcam) par d'autres capteurs afin d'utiliser le bras dans d'autres environnements que l'environnement industriel.
- L'avancement dans l'idée de pick & place dans des scénarios plus riches en termes de systèmes présents dans l'environnement du bras de robot, par exemple : la simulation d'une chaine de production avec un taper roulant.
- L'ajout de l'aspect de mobilité, le bras sera alors capable de faire plus de taches que quand il est fixé. Il y'a des entreprises qui utilisent des bras de robot manipulateurs pour des applications de classification et stockage.
- Pour élargir la gamme des objets qu'on peut performer des pick & place sur eux par notre bras , on peut utiliser un moteur DC ou un autre moteur à la place du servo moteur et on commande le couple à la place de distance .

Bibliographie

[1] Hashmi, Salman. Pick-Place Robot [en ligne]. [Consulté le 05/04/2021]. Disponible à l'adresse : https://github.com/Salman-H/pick-place-robot.

[2] Craig, John J. Introduction to Robotics, Mechanics and Control, Third Edition. Pearson Education, Inc., Upper Saddle River, New Jersey, 2005, 408p. ISBN 0-13-123629-6, 2001.

[3] Kevin M. Lynch and Frank C. Park. Modern Robotics Mechanics, Planning, and Control. Cambridge University Press, 05/2017, 642p. ISBN 9781107156302.

[4] Shimon Y. Nof. HANDBOOK OF INDUSTRIAL ROBOTICS, Second Edition. JOHN WILEY & SONS, INC. New York, 1999, 1327p. ISBN 0-471-17783-0.

[5] W.Khalil, E.Dombre. Modeling, identification and control of robots. Kogan Page Science, 2004, 483p. ISBN 1 9039 9666 X.

[6] MATEJ KAUKAL. INVERZNI KINEMATIKA SERIOVYCH MANIPULATORU S OMEZENOU ARCHITEKTUROU. Thèse de Magistère. Zapado ceska univerzita v Plzni Fakulta aplikovanych ved Katedra kybernetiky. 2013.

[7] MQ PTMS. Modèle Géométrique Inverse par la Méthode de Paul [en ligne]. [Consulté le 09/05/2021]. Disponible à l'adresse : https://www.youtube.com/watch?v=2e_lU7UIHqo&ab_channel=CMQPTMS.

[8] HART Tristan. DEVELOPPEMENT D'UN COBOT. Mémoire Final de PFE. INGÉNIEUR POLYTECH LILLE, Département Informatique Microélectronique Automatique. 2018.

[9] Thingiverse.com, Utilisateur : Jag. Nema 17 Stepper 5:1 Planetary Reducer [en ligne]. [Consulté le 17/05/2021].

Disponible à l'adresse : https://www.thingiverse.com/thing:8460.

[10] Ordinoscope.net. Tb6560 stepping motor driver V20 [en ligne]. [Consulté le 18/05/2021]. Disponible à l'adresse : https://www.ordinoscope.net/images/9/93/Tb6560_stepping_motor_driver.pdf.

[11] reprap.org. RAMPS 1.4/fr [en ligne]. [Consulté le 17/05/2021]. Disponible à l'adresse : https://reprap.org/wiki/RAMPS_1.4/fr.

[12] store.arduino.cc. ARDUINO MEGA 2560 REV3 [en ligne]. [Consulté le 17/05/2021]. Disponible à l'adresse : https://store.arduino.cc/arduino-mega-2560-rev3.

[13] JESSE WEISBERG. Moveo with ROS [en ligne]. [Consulté le 17/05/2021]. Disponible à l'adresse : https://www.jesseweisberg.com/moveo-with-ros.

[14] Github.com, Utilisateur : jamesarm97. BCN3D-Moveo [en ligne]. [Consulté le 18/05/2021]. Disponible à l'adresse : https://github.com/BCN3D/BCN3D-Moveo/tree/master/STL%20files.

[15] Github.com, Utilisateur : kitusmark. BCN3D-Moveo [en ligne]. [Consulté le 18/05/2021]. Disponible à l'adresse : https://github.com/BCN3D/BCN3D-Moveo/tree/master/USER%20MANUAL.

[16] Lanceral.com. Planetary Gears : Principles of Operation [en ligne]. [Consulté le 19/05/2021]. Disponible à l'adresse : https://www.lanceral.com/planetary-gears-principles-of-operation .

[17] Wyatt S. Newman. A Systematic Approach to Learning Robot Programming with ROS. CRC PRESS, Taylor & Francis Group, New York, 2018, 531p. ISBN-13: 978-1-4987-7782-7.

[18] Carol Fairchild, Dr. Thomas L. Harman. ROS Robotics By Example, Bring life to your robot using ROS robotic applications. Packt Publishing Ltd, BIRMINGHAM – MUMBAI, 06/2016, 428p. ISBN 978-1-78217-519-3.

[19] Anil Mahtani, Luis Sanchez, Enrique Fernandez, Aaron Martinez. Effective Robotics Programming with ROS Third Edition. Packt Publishing Ltd, BIRMINGHAM B3 2PB, UK, 12/2016, 556p. ISBN 978-1-78646-365-4.

[20] Lentin Joseph. Robot Operating System for Absolute Beginners, Robotics Programming Made Easy. Cheerakathil House Aluva, Kerala, India, 2018, 293p. ISBN-13 (pbk): 978-1-4842-3404-4.

[21] Lentin Joseph. ROS Robotics Projects: Build a variety of awesome robots that can see, sense, move, and do a lot more using the powerful Robot Operating System. Packt Publishing Ltd, BIRMINGHAM – MUMBAI, 2017, 446p. ISBN 978-1-78217-519-3. ISBN 978-1-78355-471-3.

[22] Lentin Joseph, Jonathan Cacace. Mastering ROS for Robotics Programming, Second Edition Design, build, and simulate complex robots using the Robot Operating System. Packt Publishing Ltd, BIRMINGHAM – MUMBAI, 2018, 570p. ISBN 978-1-78217-519-3. ISBN 978-1-78847-895-3.

[23] Morgan Quigley, Brian Gerkey & William D. Smart. Programming Robots with ROS: A PRACTICAL INTRODUCTION TO THE ROBOT OPERATING SYSTEM. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2015, 447p. ISBN 978-1-4493-2389-9.

[24] Ramkumar Gandhinathan, Lentin Joseph. ROS Robotics Projects Second Edition: Build and control robots powered by the Robot Operating System, machine learning, and virtual reality. Packt Publishing Ltd, BIRMINGHAM – MUMBAI, 2019, 449p. ISBN 978-1-83864-932-6.

[25] Capitol Technology University. What is SolidWorks [en ligne]. [Consulté le 22/05/2021]. Disponible à l'adresse : https:// www.captechu.edu/blog/solidworks-mechatronics-design-and-engineering-program .

[26] Wiki.ros.org. sw_urdf_exporter [en ligne]. [Consulté le 22/05/2021]. Disponible à l'adresse : http://www.wiki.ros.org/sw_urdf_exporter .

[27] Github.com, Utilisateur : qlzh727. TensorFlow Object Detection API [en ligne]. [Consulté le 22/05/2021]. Disponible à l'adresse :

https://github.com/tensorflow/models/tree/master/research/object_detection.

[28] Github.com, Utilisateur : qlzh727. TensorFlow Object Detection models [en ligne]. [Consulté le 22/05/2021]. Disponible à l'adresse : https://github.com/tensorflow/models/tree/master/research/object_detection/models.

[29] Cocodataset.org. Coco : Common Objects in Context [en ligne]. [Consulté le 23/05/2021]. Disponible à l'adresse : https://cocodataset.org/#home.

[30] Wikipédial'encyclopédie libre. Gazebo (logiciel) [en ligne]. [Consulté le 24/05/2021]. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Gazebo_(logiciel).

[31] Opencv.org. OpenCv [en ligne]. [Consulté le 24/05/2021]. Disponible à l'adresse : https://opencv.org/.

[32] ai.univ-paris8.fr. OpenCV [en ligne]. [consulté le 24/05/2021]. Disponible à l'adresse : https://www.ai.univ-paris8.fr/~chalencon/Vision/openCV.html.

[33] Paris Descartes. Image Processing and Analysis: Introduction for Computational Biology [en ligne]. Paris : Paris Descartes, 2010, 38p. [consulté le 24/05/2021]. Disponible sur < http://helios.mi.parisdescartes.fr/~lomn/Cours/BC/Publis/CompBio4.pdf >.

[34] Tsi.enst.fr. Le seuillage adaptatif (Adaptive Thresholding) [en ligne]. [consulté le 25/05/2021]. Disponible à l'adresse : http://www.tsi.enst.fr/pages/enseignement/ressources/beti/dither/arnaud/adapt.htm#contrained.

[35] Anaconda.com. Anaconda [en ligne]. [Consulté le 18/06/2021]. Disponible à l'adresse : https://www.anaconda.com/

[36] Nvidia.com. Cuda zone [en ligne]. [Consulté le 18/06/2021]. Disponible à l'adresse : https://developer.nvidia.com/cuda-zone

[37] Nvidia.com. Nvidia Developer [en ligne]. [Consulté le 18/06/2021]. Disponible à l'adresse :

https://developer.nvidia.com/cudnn

[38] Tensorflow.org. Tensorflow install [en ligne]. [Consulté le 18/06/2021]. Disponible à l'adresse : https://www.tensorflow.org/install/source#tested_build_configurations

[39] Oracle.com. Définition de la variable PATH [en ligne]. [Consulté le 19/06/2021]. Disponible à l'adresse : https://docs.oracle.com/cd/E19620-01/805-1608/customize-10/index.html

[40] Github.com, Utilisateur : fyangf and tensorflower-gardener. TensorFlow Model Garden [en ligne]. [Consulté le 22/06/2021]. Disponible à l'adresse : https://github.com/tensorflow/models

[41] Github.com, Utilisateur : syiming. TensorFlow 1 Detection Model Zoo [en ligne]. [Consulté le 22/06/2021]. Disponible à l'adresse : https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md

[42] Github.com, Utilisateur : tzutalin. LabelImg [en ligne]. [Consulté le 22/06/2021]. Disponible à l'adresse : https://github.com/tzutalin/labelImg