الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA
كلية التكنولوجيا
Faculté de Technologie
قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Automatique

Spécialité Automatique et systèmes

Conception d'un Bras Manipulateur asservi par une Caméra Embarqué

présenté par :

Ouchefoun Imane

Proposé par : Mr. KAZED Boualem

Année Universitaire 2018 - 2019

En préambule à ce mémoire je remercie ALLAH qui m'aide et me donne la patience et le courage durant ces langues années d'étude.

Je souhaite adresser mes remerciements les plus sincères aux personnes qui nous ont apporté leurs aides et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Je tien à remercier sincèrement Monsieur, B. KAZED qui, en tant que Directeurs de mémoire, s'est toujours montrés à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Je remercie les membres du jury pour avoir bien voulu examiner et juger ce travail.

Dédicace

***J**e dédie ce mémoire à mes chers parents ma mère OURDIA et mon père Mohammed pour leur patience, leur amour, leur soutien et leurs encouragements.*

***A** ma chère sœur, NASSIMA*

***A** mon cher frère ISMAIL*

***A** ma grand mère FATIMA et mon cher grand père Mohammed paix à leurs âmes*

***A** mon cher ami Ahmed*

***A** mes amis Samah, Linda, Katia, Maroi, Sihem et mes camarades de classe*

Merci d'avoir été toujours là pour moi.

ملخص:

يتضمن هذا المشروع تصميم وبناء والتحكم في ذراع مناور على منصة بعجلتين مع كاميرامن اجل كشف و تتبع الاجسام المتحرك باستخدام مكتبة برمجية للرؤية الحاسوبية بواسطة رازبيري باي.

الكمبيوتر الرازبري باي مسؤول عن تنفيذ معالجة الصور المتنوعة على فيديو البث،ثم يقوم بحساب موضع الجسم لإرساله إلى بطاقة اردوينو باستخدام ناقل تسلسلي لالتقاط هذا الكائن بذراع مناور ونقلها من مكانها بعد حساب لوحة للمفاصل الأربعة لذراعنا الآلي

كلمات المفاتيح:كمبيوتر الرازبري باي، معالجة الصور بالكمبيوتر، حافلة i2س، كاميرا، اردوينو

Résumé :

Ce projet concerne la conception, la réalisation et la commande d'un bras manipulateur sur une plateforme mobile avec une caméra embarquée pour détecter et suivre un objet de couleur prédéfinie en utilisant la bibliothèque OpenCV implémentée sur le nano ordinateur Raspberry Pi3. Ce dernier sera chargé d'effectuer les différents traitements d'images sur la vidéo diffusée, ensuite il calcule la position de l'objet pour transmettre les coordonnées de cet objet à une carte Arduino via un bus série. Le rôle de cette carte arduino sera de calculer la cinématique inverse afin de déterminer les valeurs angles des 4 articulations constituant notre bras manipulateur.

Mots clés : OpenCV ; Suivi d'objets ; Raspberry Pi ; Traitements d'images ; le bus I^2C ; Arduino

Abstract :

This project involves the design, construction and control of a manipulator arm on a mobile platform with an onboard camera to detect a colored object using the OpenCV library implemented on a Raspberry pi3 nano computer. The latter will be responsible for performing the various image processing on the broadcasted video, it then calculates the position of the object to transmit to the Arduino card via a serial bus. The Arduino board will then calculate the inverse kinematics in order to determine the angular positions of the 4 articulations of our robot arm.

Keywords: object tracking; OpenCV; Raspberry pi; Images processing; I^2C bus; Arduino.

Listes des acronymes et abréviations

BGR: Blue, Green, Red

BSD: *Berkeley Software Distribution license*

BVR: Bleu, Vert, Rouge

C°: Celsius°

CC: courant Continue

CD-Rom: Compact Disc - Read Only Memory

Cm: Centimètre

CSI: Camera Serial Interface

FPS: Frame per Second

HD: Haute Definition

HSV: Hue, Saturation, Value

I²C: *Inter Integrated Circuit*

IDE: Integrated Development Environment

KB: Kilo Bytes

KO: kilo Octets

mA: milliampère

Mac OS: *Macintosh Operating System*

MHz: Mega hertz

MP: Mega Pixels

openBR *: Open source Biometrics*

OpenCV: *Open source Computer Vision*

Pixels: *Picture Elements*

PWM: Pulse Width Modulation

SoC: system on Chip

Tr/min: Tour/minute

TTL: logique transistor-transistor

USB: Universal Serial Bus

V: Volts

Sommaire

Liste des figures :

Liste des tableaux :

Introduction Générale

Introduction Générale

Depuis le développement de la technologie, les robots manipulateurs sont devenus un outil de travail dans les différents domaines touchant l'activité humaine. Le secteur le plus concerné est certainement celui de l'industrie manufacturière qui nécessite, dans plusieurs cas des tâches répétitives et parfois dangereuses pour un opérateur humain. Ceci représente l'une des raisons principales qui ont conduit à un développement très rapide de nombreux types de bras manipulateurs aujourd'hui sur le marché.

En ce qui concerne le présent travail, il représente le prolongement d'un projet réalisé en 2018 [1] et qui avait comme objectif de concevoir un bras manipulateur de type PUMA 560 à 4 degrés de liberté, contrôlé par une carte Arduino. La partie qui nous a été confiée consiste à donner à ce robot une autonomie de mouvement dans le sens où il devra lui-même déterminer la position d'un objet de couleur choisie pour déplacer son effecteur final afin atteindre cet objet. La détection de l'objet en question sera réalisée grâce à une caméra dont sera muni notre bras manipulateur. Bien que la carte Arduino a des capacités non négligeables elle n'est pas en mesure de réaliser les traitements d'images vidéo nécessaires à ce genre d'applications. C'est ainsi que nous avons opté pour l'utilisation d'un nano ordinateur de type Raspberry pi 3B, qui possède suffisamment de ressources pour subvenir aux besoins de notre projet. L'essentiel du travail sera donc axé sur la partie vision et la communication entre les deux cartes pour faire en sorte que les coordonnées de l'objet recherché, obtenues dans le repère de la caméra, soient converties en celles du repère du bras avant d'être transmises, en temps réel, vers la carte Arduino. Afin d'alléger le traitement au niveau de la carte Raspberry pi une simulation, sur Matlab, sera effectuée pour déterminer la Matrice de Transformation Homogène assurant le changement de repère entre la position caméra et celle de la base du bras manipulateur.

Tous les détails de chacune des parties citées ci-dessus seront expliqués dans les chapitres qui suivent :

- Le premier chapitre : contient sur les généralités des robots

- Le deuxième chapitre : définition sur le traitement d'image qui est une science récente qui a pour but d'offrir des différents domaines.

- Le troisième chapitre : présentation détaillé sur les robots manipulateuret introduction sur la cinématique inverse et directe

- Le quatrième chapitre : Présentation générale du projet que ce soit les matériels utilisés ou les logiciels, on explique le mode software et hardware en expliquant le rôle des deux cartes, programmation et la liaison entre elle, ensuite en présente la modélisation d'une façon générale.

- Le cinquième chapitre : les résultats de nos projets pour terminer le mémoire, ont conclue tout le travail en donnant une conclusion générale

Chapitre 1 :
Généralités sur les robots

1.1 Introduction

La robotique est un ensemble des disciplines techniques (automatique, électronique, mécanique, informatique) qui agisse physiquement sur son environnement à fin d'atteindre un objectif qui lui a été assigné.

On distingue deux types de robotique, la robotique industrielle et la robotique mobile. Les robots manipulation ont une base fixe avec un système mécanique articulé devant effectuer des opérations (la peinture, la manutention dans des ateliers dès l'industrie automobile, emboutissage, etc.). Par contre les robots mobiles, on regroupe sous l'appellation robots mobiles l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs (industrielle). Par contre l'usage néanmoins que l'on désigne habituellement par ce terme les robots mobiles à roues.

1.2 Les types de robots

Il existe deux types robots de manipulations et les robots mobiles :

1.2.1 Les robots mobiles

Définition robots mobiles cette une machine est polyvalente et capable de s'adapter à certaines variations ses conditions de fonctionnement. Elle est dotée de fonctions et capacités de perception, de perception de décision et d'action qui lui permettent d'accomplir correctement sa tâche d'une manière autonome.

1.2.1.1 La structure mécanique robotique

Il existe plusieurs de structures mécaniques permets ces structures suivantes :

- **Les mobiles à roues :**

La mobilité par roues est la structure mécanique trop appliquée, ce mécanisme assure l'agencement et les dimensions des roues afin de déplacer dans toutes les directions avec une accélération et une vitesse importante.

Figure 1-1: Représente des exemples sur les robots mobiles à roues

- **La classe des robots mobiles à roues :**

Les robots à roues déterminent plusieurs classes, généralement, par la position et le nombre de roues utilisées. Voilà les quatre classes principales :

- Robot uni cycle. (a) figure 1
- Robot tricycle. (b) figure 1
- Robot de type voiture. (c) figure 1
- Robot à roues omnidirectionnelles. (d) figure1

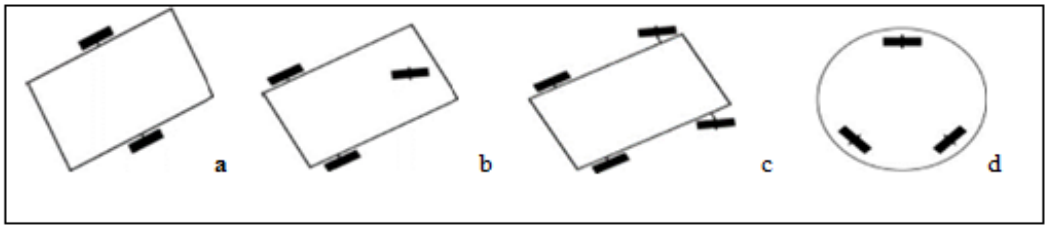


Figure 1-2: Les classes des robots mobiles à roues : roues uni cycle, tricycle, voiture,

- **Les mobiles à chenilles :**

L'avantage d'utiliser les chenilles d'une bonne adhérence au sol et faculté de franchissement d'obstacles

Figure 1-3: Les robots mobiles à chenilles

- **Les mobiles marcheurs :**

L'utilisation les robots marcheurs sont destinés à réaliser différentes taches dont l'accès au site est difficile, impossible ou dangereux à l'homme. Sachant leur l'anatomie à plusieurs degrés de liberté permet un rapprochement avec les robots manipulateurs.

Figure 1-4: Des robots marchants

- **Les robots rampants :**

C'est un robot inspirer par locomotion des animaux qu'ils ont utilisées techniques découlent à leurs méthodes, afin cette structure soit modulaire et flexibles pour les déplacements sur tous les terrains (insertion dans passages étroits).

Figure1-5: Des exemples sur robots rampants

Les robots mobiles prennent plusieurs formes diverses du point de vue de leur structure mécanique et de leur commande. Dans les figures suivantes présentent plusieurs types de robots dans différents domaines :

Figure 1-6: Un mini drone autonome

Figure 1-7: Un robot médical

Figure 1-8: Un sous-marin sans pilote

Figure 1-9 : Un robot militaire

1.2.1.2 La classification des robots mobiles

Comme dit précédemment, la classification des robots mobiles se faits suivant plusieurs critères (degré d'autonomie, systèmes de locomotion, domaine d'application, énergie utilisée ...).

- **Degré d'autonomie**

  Un robot mobile autonome est un système automoteur doté de capacités de prendre ses propres décisions et de moyens d'acquisition et de traitement de l'information qui lui permettent d'accomplir certaines taches sous contrôle humain réduit, dans un environnement non complètement connu.

- **Domaine d'application**

Le domaine d'application des robots mobiles est vaste, nous présentons quelques applications dans le tableau suivant (tableau 1. 1)

Domaines	Applications
Industrie nucléaire	- Surveillance de sites - Manipulation de matériaux radioactifs
Sécurité civile	- Neutralisation d'activité terroriste - Déminage - Pose d'explosif
Chimique	- Surveillance de site - Manipulation de matériaux toxiques
Mine	- assistance d'urgence
Agricole	- Cueillette de fruits - Traite, moisson, traitement des vignes.
Nettoyage	- Coque de navire - Nettoyage industriel
Espace	- exploration
industrie	- convoyage - surveillance

Sous-marine	- pose de câbles - cherche de modules - cherche de navires immergés - inspection des fonds marins
Militaire	- surveillance - pose d'explosif - manipulation de munitions

Tableau 1-1 : Les différents domaines d'application des robots mobiles

1.2.1.3 Les robots mobiles autonomes

Un robot autonome dispose d'un système d'équilibre parfait, cela doive posséder plusieurs capacités de perception, d'action et de décision qui lui permettent d'agir de manière autonome dans quel que soit environnement. Pour parvenir cela il suffit un GPS, capteur ou caméra avec un programme bien préciser afin identifier les objets et parvient ainsi à les éviter, et en fonction de la perception qu'il en a et de ses buts, et savoir réagir en conséquence suivant son niveau d'autonomie.

1.2.2 Les robots de manipulations

Comme dis précédemment, les robots ancrés physiquement à leur place de travail et généralement mis en place pour réaliser une tache répétitive, (tels que les robots industriels, médicaux, etc.).

La structure mécanique de robots manipulateur est animée par des actionneurs, à partir élaborés par un calculateur. Ces ordres dépendent des informations délivrées par les capteurs. L'utilisation des capteurs externes, capteurs extéroceptifs, afin de mesurer et évaluer l'interaction du robot avec l'environnement directement de leur organe terminal devient une pratique de plus en plus courante dans les applications robotiques de haute précision.

Figure 1-10: Représente des robots manipulation

1.2.2.1 Décomposition véhicule / bras / effecteurs

En générale, le robot est constitué d'un bras, d'un outil et d'une base mobile. Par contre dans la réalité, les robots sont à poste fixe (pas de base)

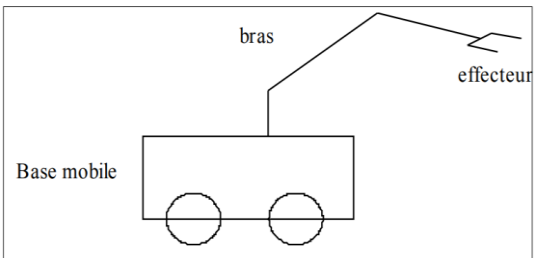


Figure 1-11 : Robot manipulateur mobile

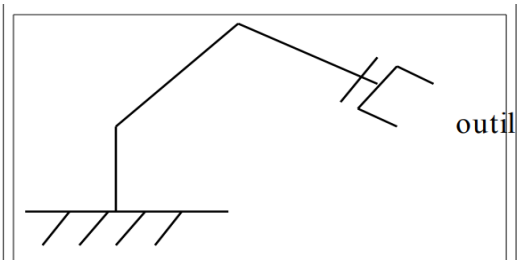


Figure 1-12: Robot manipulateur fixe

De plus l'outil étant une pièce interchangeable et complexe, il n'est jamais pris en considération pour l'étude de la structure.

1.2.2.2 Le bras manipulateur

Autrement dit, c'est un robot généralement programmable, avec des fonctions similaires à un bras humain. Les axes sont reliés à des liens de ce manipulateur afin de permettre, soit du mouvement de rotation (comme dans un robot articulé) ou de translation (linéaire) de déplacement.

Il peut être contrôle manuellement ou autonome et peut effectuer une variété de taches avec une grande précision.

Les manipulateurs peuvent être mobiles ou fixes (sans ou avec roues) et peuvent être conçus pour des applications industrielles.

Figure 1-13: Des exemples sur les bras manipulateurs

1.2.2.2.1 Les constituants du bras d'un robot

Il est composé d'une base, élément porteur et organe terminal comme est montré sur la figure suivante :

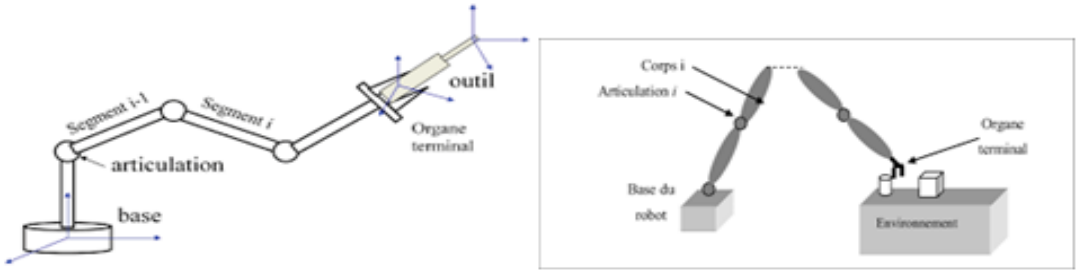


Figure 1-14 : Vocabulaire d'un bras manipulation

- La base : est fixée sur le lieu du travail.
- Elément porteur : représente l'essentiel de système mécanique articulé (segment, actionneur, articulation, l'organe terminal), il a pour rôle d'amener l'organe terminal dans une situation donnée imposée par la tâche. Il est composé d'un ensemble de corps souples ou rigides liés par des articulations, afin de déplacer l'organe terminal d'une configuration à une autre.
- Organe terminal :
  Désigne tout dispositif destiné soit à manipuler des objets comme les dispositifs de serrage (pinces à deux ou trois doigts), les dispositifs magnétiques ou à dépression (ventouse), soit à transformer (outils de découpe, torche de soudage, torche de peinture). Il s'agit d'une interface permettant au robot d'interagir avec son environnement.

1.3 Conclusion

Dans ce chapitre, nous avons donné une idée générale sur la robotique, et un aperçu sur les deux types de robots et les bras de manipulations, leurs structures, leurs utilisations. Par contre notre projet s'intéresse au robot mobile à roue fixé à leur mécanisme un bras manipulateur, pour mieux comprendre notre robot nous allons expliquer dans la chapitre qui suit le traitement d'image.

Chapitre 2 :
Traitement d’image

2.1 Introduction

Le traitement d'images est une science récente qui a pour but d'offrir de différents domaines, ce traitement est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information.

Nous nous attachons dans cette partie à la détection de suivi d'objet dans ce projet

2.2 L'image numérique

Tous d'abord l'image numérique désigne toute image qui a été acquise traitée et sauvegardé sous forme codée représentable par des nombres valeurs numériques sur une clé USB, disque dur, téléphone portable, etc.

Elles sont constituées d'une multitude de pixels organisés en lignes (y) et colonnes (x). Chaque pixel possède une couleur.

Figure 2-1: Une image numérique en couleur en pixels

2.3 Les espaces de couleurs

L'espace de couleur est utilisé pour représenter les couleurs sous forme numérique. Les écrans d'ordinateurs reconstituent une couleur par synthèse additive à partir de trois couleurs primaires

2.3.1 BGR

BGR (Blue, Green, Red) ce qui veut dire BVR (Bleu, Vert, Rouge) basé sur un mélange additif (combinaison de rayons lumineux) de ces trois couleurs

L'intensité de chaque couleur est codée sur 1 octet permettant 256 niveaux différents alors on obtient 256*256*256=16, 777,216 codes de couleurs différents pour chaque pixel.

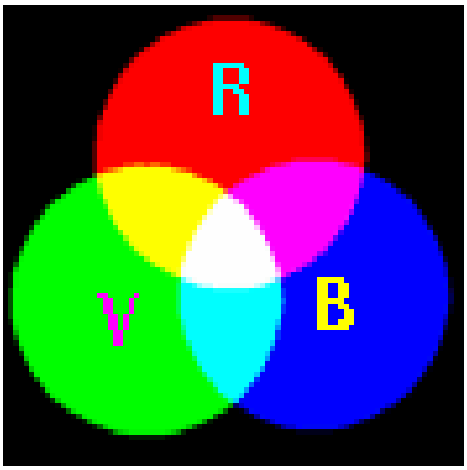


Figure 2-2: Codage RGB

2.3.2 HSV

HSV (Hue, Saturation, Value) ou bien HSB (Hue, saturation, Brightness) qui veut dire :

- **La teinte (H) :** Dans les systèmes de description de couleur informatique, la teinte est la forme pure d'une couleur, c'est-à-dire sans adjonction de blanc. Elle est codée suivant l'angle qui lui correspond sur le cercle des couleurs (de 0° à 360°).
- **Saturation (S)** : Décrit la quantité du blanc dans la couleur pure (0 à 100%).
- **La valeur (V) :** Elle décrit l'obscurité de la couleur, plus la valeur d'une couleur est faible, plus la couleur est sombre (de 0 à 100%).

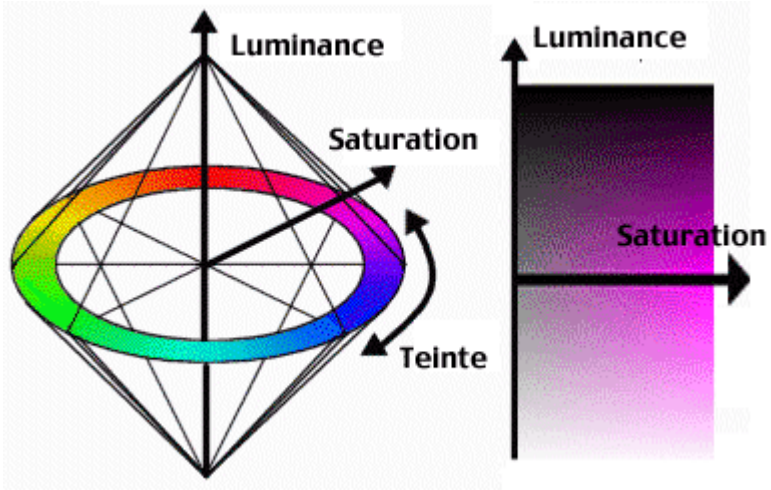


Figure 2-3: Codage HLS ou bien HSV (Hue, Luminance, Saturation)

2.3.3 Niveau de gris (grayscale)

Les pixels de l'image sont représentés par plusieurs niveaux de gris. La valeur de luminosité de chaque pixel d'une image en ce mode est comprise entre 0 (noir) et 255 (blanc).

2.4 Détection d'objet

Le Traitement d'Image est constitué d'un ensemble de traitements que nous appellerons Opérateurs permettant d'extraire de l'information d'une ou de plusieurs images. Cette information peut prendre les trois formes suivantes :

- une image, par exemple pour l'amélioration de la visualisation : on parlera alors de traitement de ***Bas Niveau***,

- de valeurs numériques, éventuellement associées à une image, par exemple l'extraction de segments de contours, les valeurs numériques étant les coordonnées des points extrémités. On parlera de traitement de ***Niveau Intermédiaire***,

- de valeurs symboliques, ou de décisions : par exemple :

Un cercle, une droite, ou un objet prédéfini ont été reconnus dans l'image,

Une pièce manufacturée a été reconnue conforme : le préhenseur du bras de robot, ou du robot delta va prendre la pièce sur le tapis roulant et la stocker,

Un char a été reconnu : le tir va être effectué.

On parlera d'un traitement de ***Haut Niveau***.

L'incrustation d'une séquence d'images passe d'une image à une autre dans un ensemble d'images. Elle vous permet de créer des effets, en traçant une trajectoire qui un ensemble des positions successives occupées par ce point au cours du temps (latrace) que laisse dans un référentiel le système au cours de son déplacement. La trajectoire d'un point dépend du référentiel d'étude.

Figure 2-4: Une séquence d'image

2.4.1 La conversion de l'image

OpenCV capture habituellement des images et des vidéos en format 8-bit, entier non signé, BGR. En d'autres termes, les images capturées peuvent être considérées comme 3 matrices, bleu, rouge et vert avec des valeurs entières comprises entre 0 et 255, chaque petite case représente un pixel de l'image. Dans les vraies images, ces pixels sont si petits que l'œil humain ne peut pas les différencier.

Habituellement, on peut penser que L'espace de couleur BGR est plus approprié pour la segmentation basée sur la couleur. Mais L'espace de couleur HSV est l'espace de couleur le plus approprié pour la segmentation d'image basée sur la couleur.

A la base c'est que le HSV est le meilleur pour la détection d'objet, donc il est préférable de convertir de l'espace BGR à HSV car que l'apparence de couleur d'un objet peut fortement varier avec l'éclairage de la scène, et au moment que le BGR décrit les couleurs en termes de quantité de Bleu, vert et rouge c'est-à-dire qu'il n'est pas le plus

robuste aux variations des conditions, quant à le modèle HSV décrit les couleurs de façon similaire à la façon dont l'œil humain a tendance à les percevoir.

2.4.2 La binarisation

Afin de pouvoir effectuer des traitements sur des images binaires, il faut d'abord en posséder. Nous allons donc apprendre à transformer des images en binaire. La binarisation consiste à transformer un pixel sur plusieurs bits (2, 4, 8 ou plus) en une image sur 1 bit. Pour ça, nous allons faire un seuillage. Si la valeur du pixel est en dessous du seuil, nous lui associons la valeur 0. Si la valeur du pixel est égale ou supérieure au seuil nous lui donnons la valeur 1. Si l'image est en niveau de gris il n'y a pas de problème car il n'y a qu'une seule composante de couleur. C'est à dire qu'un pixel est codé avec un nombre. En ce qui concerne les images couleur, c'est différent. En effet nous avons 3 composantes de couleur (rouge bleu vert par exemple). La première étape consiste donc à transformer une image en couleurs en niveaux de gris puis en image binaire. La moyenne des composantes d'un pixel revient à le transformer en niveau de gris.

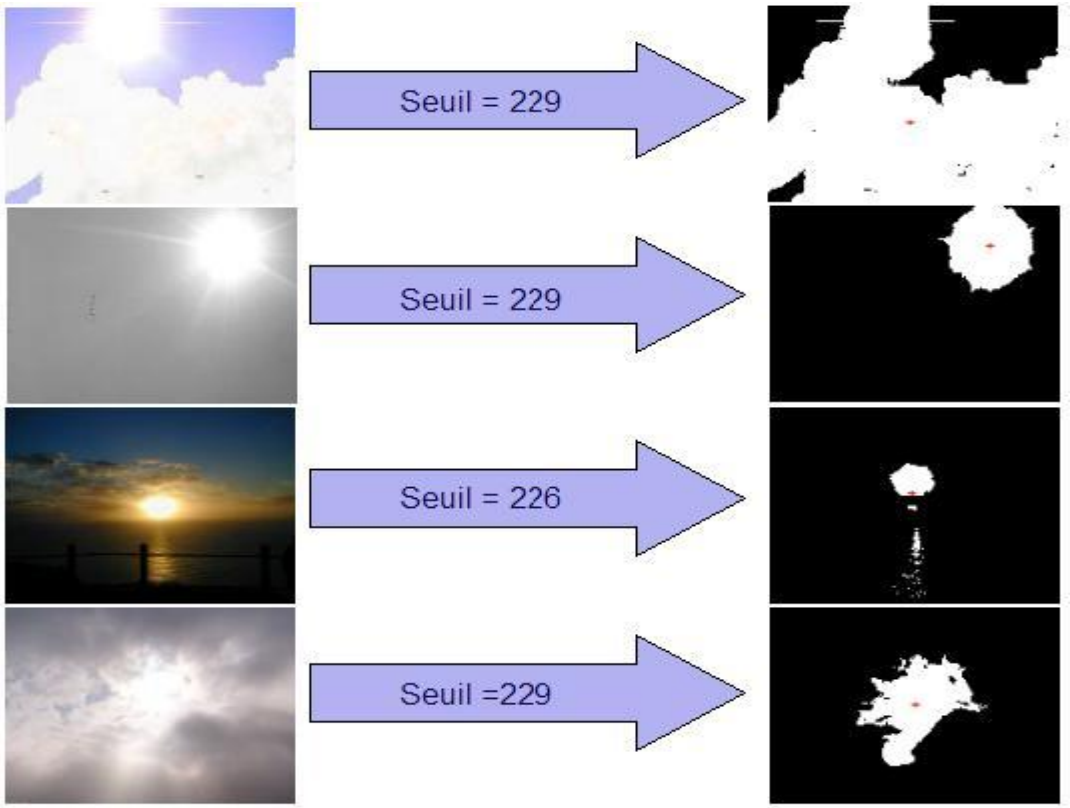


Figure 2-5: la binarisation en fonction d'un seuil

2.4.1 Le masque de l'objet

On appelle masque de l'objet mobile, la projection ou la région occupée par cet objet dans le plan image. L'obtention de ce masque permet de se focaliser sur les zones d'intérêt de l'image. Dans notre image binarisée le masque est rempli en blanc et correspond à l'objet traqué, et tout le reste est en noir. Tous les pixels seront remplacés un à un suivant un intervalle de valeurs HSV sélectionnées (un seuillage Hat). Ainsi, si un pixel à une valeur supérieure au seuil, il prendra la valeur *255* (blanc), et si sa valeur est inférieure, il prendra la valeur **0** (noir). [2]

2.4.2 Opérations sur l'image binaire

Apres l'obtention d'une image binaire, qui nous indique où se trouve la couleur traquée par des taches blanches et tous le reste en noir. Afin de suivre cet objet il faut l'isoler.

Une image binarisée est bruitée par des pixels éparpillés qui appartiennent à l'intervalle des valeurs HSV, mais qui ne présentent pas l'objet traqué. Afin de s'en débarrasser on fait appel à une ouverture morphologique.

L'ouverture morphologique est le résultat d'une érosion suivie d'une dilatation de l'ensemble érodé donc : une application successive de deux operateurs morphologiques :

Une érosion : L'opération d'érosion consiste à éliminer les points blancs isolés. Cette transformation nous permet de supprimer les pixels isolés (qui appartiennent à l'intervalle des valeurs HSV) mais qui ne correspondent pas à l'objet cherché.

Une dilatation : L'opération de dilatation consiste à éliminer les points noirs isolés au milieu des parties blanches. Cette transformation nous permet de renforcer les groupes denses de pixels de l'objet cherché. [2]

2.4.3 Difficultés de la méthode

C'est une méthode assez simple à mettre en œuvre, la rapidité de cette technique la rend très attrayante pour les applications en temps réel, mais en raison de sa simplicité, de nombreux problèmes peuvent entraîner l'échec du suivi.

L'apparence d'un objet peut varier fortement avec l'´eclairage de la scène ce qui rend la détection des couleurs dépendante de la luminosité (obscurité, voyant s'allume, ombres).

Si un autre objet possède la même couleur on ne peut les différencier, excepte s'il y a une grande différence de taille entre les deux objets. [2]

2.5 Conclusion

Dans ce chapitre on a présenté comment se déroule le traitement d'image en expliquant les pixels, le code de couleurs de BGR vers HSV pour mieux comprendre la détection de la balle bleue. Maintenant nous intéressant sur les différents bras robotique pour mieux comprendre notre robot manipulateur dans le prochain chapitre.

Chapitre 3 :
Le robot manipulateur

3.1 Introduction

Dans ce chapitre nous allons présenter la Morphologie générale d'un robot manipulateur et les structures possibles du bras, nous allons introduire les modèles cinématique directe et inverse pour mieux comprendre l'utilisation de notre bras 4 (DDL) (RRRR), puis nous allons montrer le changement des repères (base, caméra, bras) par la transformation homogène.

3.2 Structure générale du robot manipulateur

3.2.1 Morphologie générale

Un robot manipulateur est un ensemble constituer par :

- Une structure mécanique qui supporte l'organe terminale à situer.
- Des capteurs sont importants à la commande.
- Des actionneurs sont utilisés afin d'agir sur la structure mécanique pour changer la situation de l'organe terminal.
- Le robot manipulateur pilote les actionneurs par un système de commande.
- Un système décisionnel qui assure la fonction de raisonnement et élabore le mouvement du robot manipulateur.
- Un système de communication qui s'occupe de message transmis entre le système l'opérateur et décisionnel via une console visualisation.

Les deux corps du robot manipulateur extrêmes ont des rôles particuliers :

- La base, qui sur un véhicule ou fixée au sol.
- L'organe terminal qui porte l'outil (ou effecteur). [3]

3.2.2 Les structures de la base vers l'effecteur

Les structures séries, ou sérielle pour aller vers lesquelles le graphe est arborescent, la base étant la racine et les feuilles étant les organes terminaux.

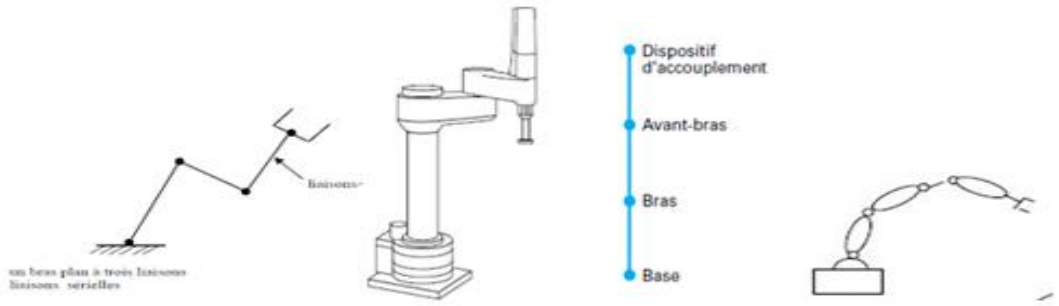


Figure 3-1 : structure séries

Les structures parallèles pour lesquelles toutes les chaines partent de la base pour aller vers l'organe terminal.

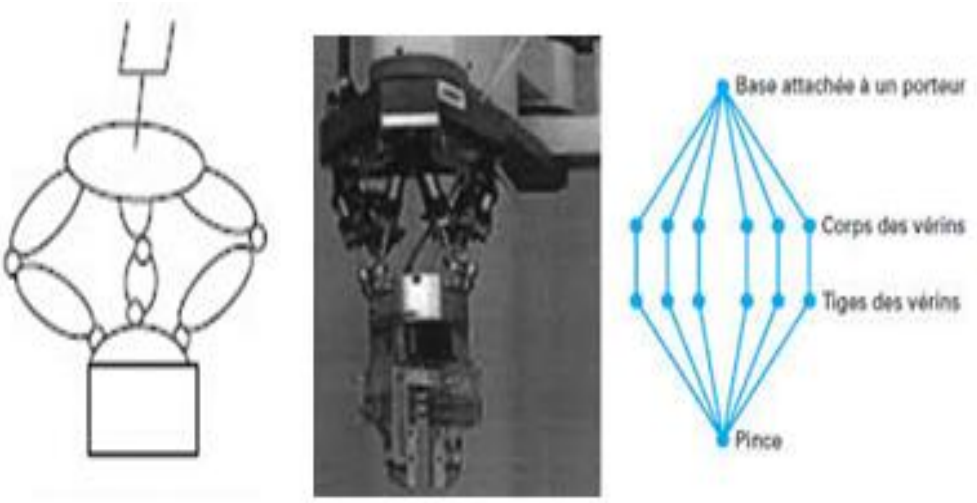


Figure 3-2 : structure parallèle

Les structures mixentles deux structures précédentes, présentant des boucles cinématiques, par exemple des parallélogrammes ou des motorisations par vérins linéaire.

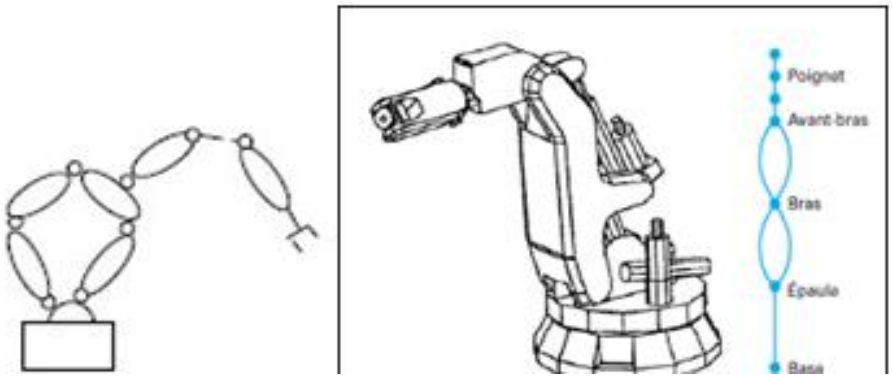


Figure 3-3: structure mixe

3.2.3 Les liaisons mécaniques

Une liaison (articulation) lie deux corps en limitant le nombre de DDL de l'un des solides par rapports à l'autre. Soit m le nombre de DDL résultant, m est alors appelé la mobilité de l'articulation. On a toujours :

$$0 \leq m \leq 6$$

On générale, il existe quartes articulations les plus courantes dans l'assemblage de robots manipulation sont :

- L'articulation rotoides (ou pivot)
- L'articulation prismatique (ou glissière)
- L'articulation rotule S (sphérique)
- L'articulation cardan U (joint universel)

Certaines articulations sont motorisées, on parlera de liaisons actives. Ce sera généralement les cas des articulations rotoides ou prismatiques. Les articulations non motorisées seront appelées les articulations passives.

Ce sera généralement le cas des articulations rotule, cardan et bien sûr appui sur plan.

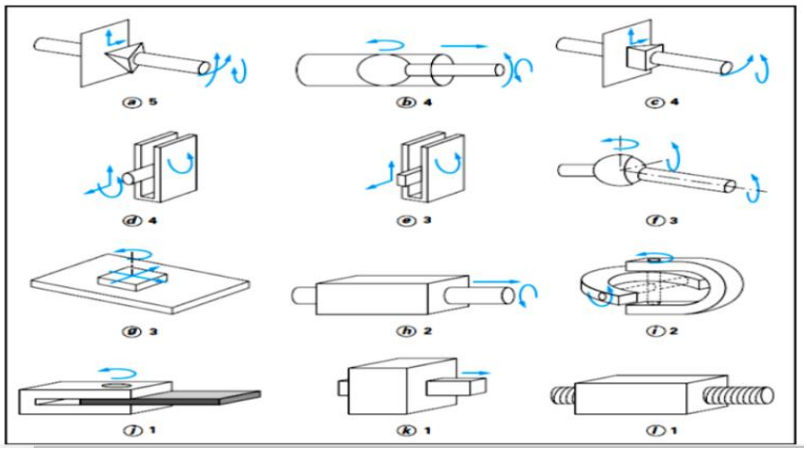


Figure 3-4: différente types d'articulations

3.2.4 Architecture des bras

Les bras sont décrits par :

- La nature des articulations (Rotoide, Prismatique).
- L'angle entre deux axes successifs (0°, 90°).

La possibilité du nombre de morphologie en fonction du nombre d'articulation se déduit par combinaison de quatre paramètres (0°, 90°, Prismatique, Rotoide). [3]

DDL	Nombre de structures
1	2
2	8
3	32
4	128
5	512
6	2048

Tableau 3-1 : Nombre de morphologie en fonction du nombre d'articulation

Par contre en pratique, on sépare les trois premiers DDL qui constituent le porteur du robot et les suivants constituent le poignet.

3.2.4.1 Les porteurs

En utilisant trois axes, on peut obtenir 32 structures théoriquement par contre dans la pratique on trouve juste cinq suivant : [3]

3.2.4.1.1 Les cartésiens TTT (Translation Translation Translation)

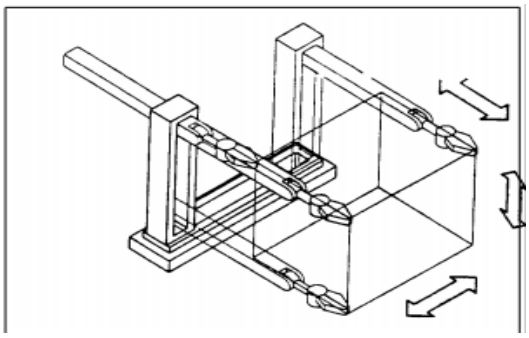


Figure 3-5: Robot à structure cartésienne

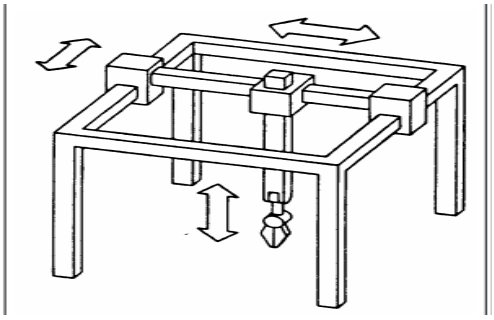


Figure 3-6: Robot portique

- Robot cartésien rectangulaire.
- Volume de travail : parallélépipède.
- Robot de type portique.
- Très bonne précision.
- Lent

3.2.4.1.2 Les cylindriques RTT (Rotoide Translation Translation)

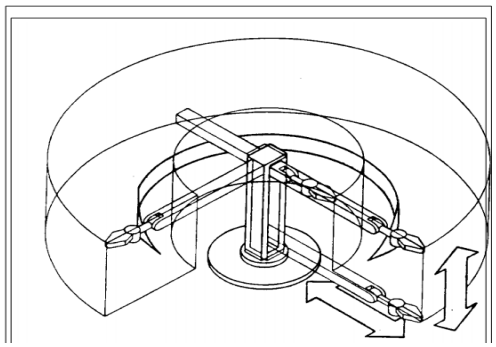


Figure 3-7: robot cylindrique

- Cylindre est un porteur de bon volume de travail.
- Application : assemblage
- Alignement sur Z de la liaison prismatique.

3.2.4.1.3 Les sphériques RRT (Rotoides Rotoides Translation)

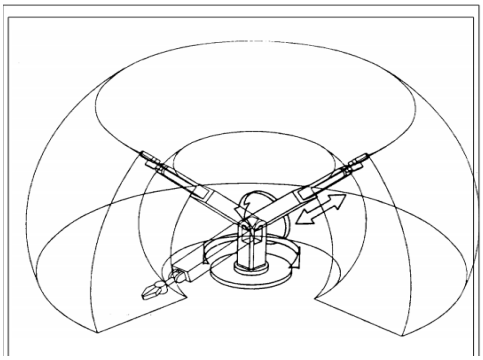


Figure 3-8: le robot sphérique RRT

- Application : déplacement d'objet, soulèvement.
- Volume de travail : sphère.

3.2.4.1.4 Les toriques RTR (Rotoide translation Rotoide)

Une articulation prismatique entre deux rotoides d'axe concourant.

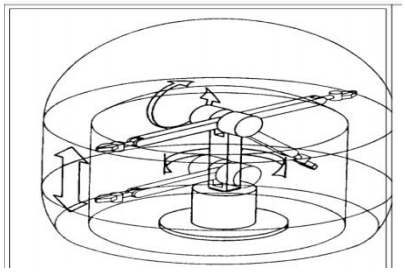


Figure 3-9: Robot polaire

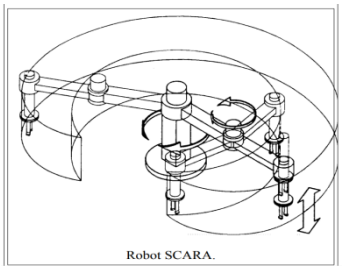


Figure 3-10: Robot SCARA

Ces porteurs sont spéciaux comme robot type SCRA (**S**lective **C**omplaint **A**rticulated **R**obot for **A**ssembly). L'articulation prismatique verticale est caractéristique des robots d'assemblage. Le robot SCARA est une insertion de composants avec rapidité et précision.

Compliance : c'est la capacité de système mécanique à s'ajuster aux contraintes et aux forces qui s'exercent sur lui et de corriger les erreurs d'orientation et de positionnement dont témoignent ces forces.

Caractéristique :

. Espace de travail cylindrique.

. Précis.

. Très rapide.

3.2.4.1.5 Les anthropomorphes RRR (Rotoide RotoideRotiode)

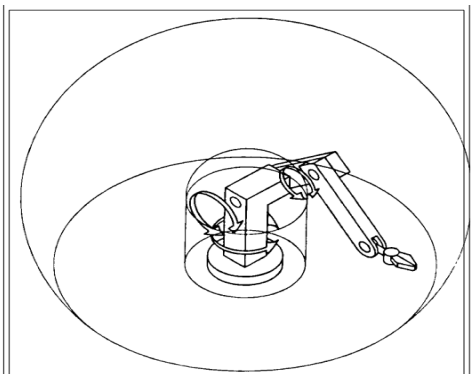


Figure 3-11: Robot anthropomorphique

- C'est le plus agile des manipulateurs car toutes les articulations sont rotoides.
- Le volume de travail grand par rapport à l'encombrement du robot.
- Large gamme d'applications industrielles.

3.3 Introduction sur la cinématique directe et inverse

3.3.1 Cinématique directe

La cinématique directe est le contrôle de notre bras robotique en fonction des positions que nous appliquons à nos servos moteurs. Nous contrôlons la totalité des servos moteurs qui correspondent aux articulations de notre bras, soit l'angle entre 2 os. C'est un contrôle manuel. [4]

3.3.1.1 Méthode géométrique

On peut obtenir une relation entre les coordonnées de l'organe terminal par rapport à un repère fixe dans un cas ou on a un nombre réduit d'articulations. [4]

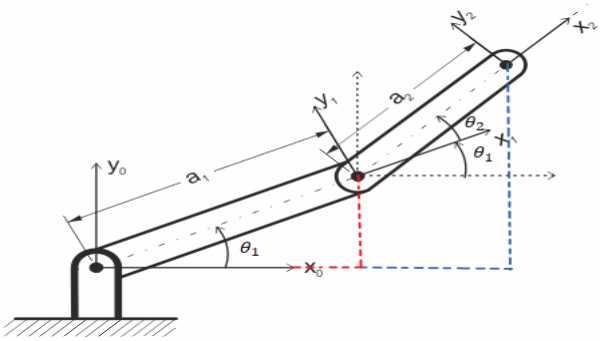


Figure 3-12: Géométrique

D'après cette figure on obtient deux équations X_p et Y_p qui est les coordonnées de l'organe terminal (p) dans un repère fixe (O, X_o, Y_o). Mais elles ne peuvent pas être calculées d'une manière aussi simple si l'architecture est plus compliqué. [4]

$$X_p = a_1 * \cos(\Theta_1) + a_2 * \cos(\Theta_1+\Theta_2) \dots (3.1)$$

$$Y_p = a_1 * \cos(\Theta_1) + a_2 * \sin(\Theta_1+\Theta_2) \dots (3.2)$$

3.3.1.2 Convention de Dénavit Hardenberg (DH)

La convention DH permet de simplifier l'analyse de la cinématique directe en appliquant un ensemble de règles. Elle permet d'exprimer toutes les transformations homogènes en termes d'un produit de 4 transformations de bras comme suit : [4]

$$H^{i-1}{}_i = Trans_{z(i-1)}(d_i) * Rot_{z(i-1)}(\Theta_i) * Trans_{x(i)}(a_i) * Rot_{xi}(\alpha_i) \quad \ldots(3.3)$$

Sous forme matrice comme suit :

$$H^{i-1}{}_i = \left[\begin{array}{ccc|c} \cos(\theta i) & -\sin(\theta i)\cos(\alpha i) & \sin(\theta i)\sin(\alpha i) & ai\cos(\theta i) \\ \sin(\theta i) & \cos(\theta i)\sin(\alpha i) & -\cos(\theta i)\sin(\alpha i) & ai\sin(\theta i) \\ 0 & \sin(\alpha i) & \cos(\theta i) & di \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Les variables Θ_i, ai, di et αi, sont des paramètres associés au lien i et l'articulation i-1. La convention DH avec ces 4 paramètres sont suffisant pour caractériser une transformation homogène avec des conditions essentielles qui faudra respecter sont :

- X_i doit être perpendiculaire à Z_{i-1}
- X_i doit couper l'axe Z_{i-1}
- Les mouvements de l'articulation (Rotoide ou prismatique) sont toujours sur axes Z_i

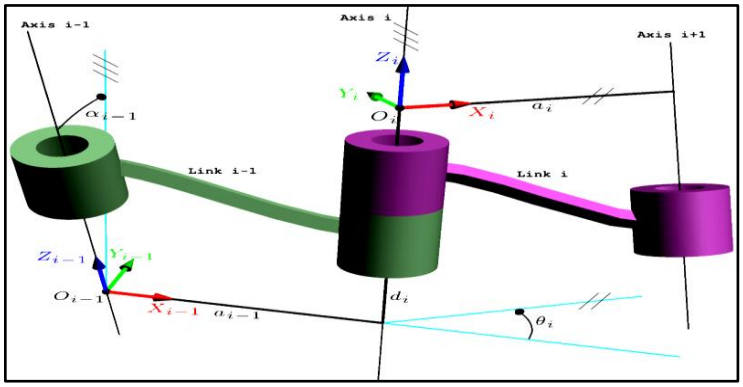


Figure 3-13: Les paramètres DH

La figure précédente représente le cas général où il n'y a qu'une seule et unique droite perpendiculaire aux deux axes de mouvement (Z_{i-1} et Z_i) car ils n'ont pas de point d'intersection. [4]

Les paramètres de la convention DH pour ce cas représentent comme suit :

- **di** représente la distance entre les deux origines Oi-1 et Oile long d'axe (Z_{i-1}).
- **Θi** représente l'angle entre les deux axes Xi-1 et Xi autour de l'axe (Z_{i-1}).
- **ai** représente la distance entre les deux origines Oi-1 et Oi le long de l'axe (Xi).
- **αi** représente l'angle entre les deux axes Zi-1 et Zi autour de l'axe (Xi).

Remarque :

1) Dans le cas où les axes (Zi-1 et Zi) sont parallèles nous avons un nombres infini de droites perpendiculaires aux deux axes, il est préférable de choisir l'axe Xi perpendiculaire passant par l'origine du repère Oi-1. [4]

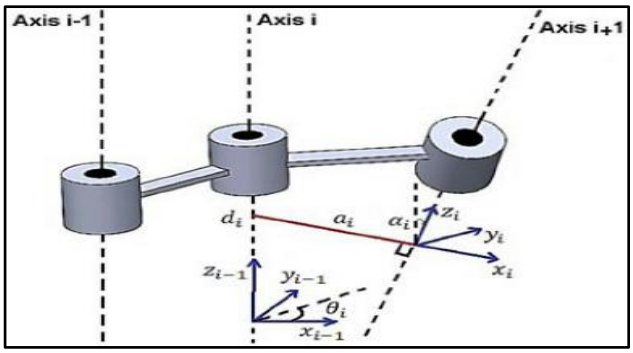


Figure 3-14: les axes (Zi-1 et Zi) sont parallèles

2) Dans le cas où les axes (Zi-1 et Zi) ont un point d'intersection, l'origine du repère(i) sera placé sur ce même point et l'axe Xi choisi comme étant le produit vectoriel (Zi-1 * Zi), il sera aussi perpendiculaire à ces deux axes : [4]

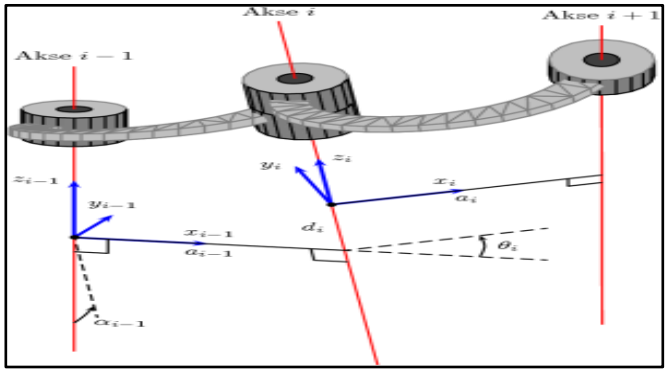


Figure 3-15: les axes (Zi-1 et Zi) ont un point d'intersection

Ce tableau suivant représente les paramètres DH après avoir placé tous les repères relatifs à toutes articulations.

Articulations	**Θ**	**d**	**a**	**α**
1	Θ1	d1	a1	α1
2	Θ2	d2	a2	α2
…..	….	…..	….	…..
i	Θi	di	ai	αi
i+1	Θi+1	di+1	ai+1	αi+1
….	….	….	….	….
n	Θn+1	dn	an	αn

Tableau 3-2 : les 4 paramètres Denavit Hartenberg

3.3.2 Cinématique inverse

La cinématique inverse est le contrôle de notre bras robotique en fonction de la position d'arrivée. Cette position est donnée selon les axes x, y et z que nous avons définis et grâce à des algorithmes, les différents servos moteurs prennent des valeurs optimum afin d'atteindre la position souhaitée. Il est possible de poser une contrainte sur une partie du bras, généralement le poignet; on peut souhaiter que celui-ci reste parallèle au sol ou alors qu'il saisisse l'objet par-dessus. La cinématique inverse nous permet de réaliser toutes ces contraintes.

3.3.2.1 L'étude d'un bras Puma quatre degrés de libertés (4DDl)

Voici un schéma qui présente notre bras 4 (DDL) théoriquement pour déterminer les quatre équations des angles (Θ_1, Θ_2, Θ_3, Θ_4) pour que le bras puisse atteindre la position voulue selon les données (X, Y, Z) imposer par la caméra.

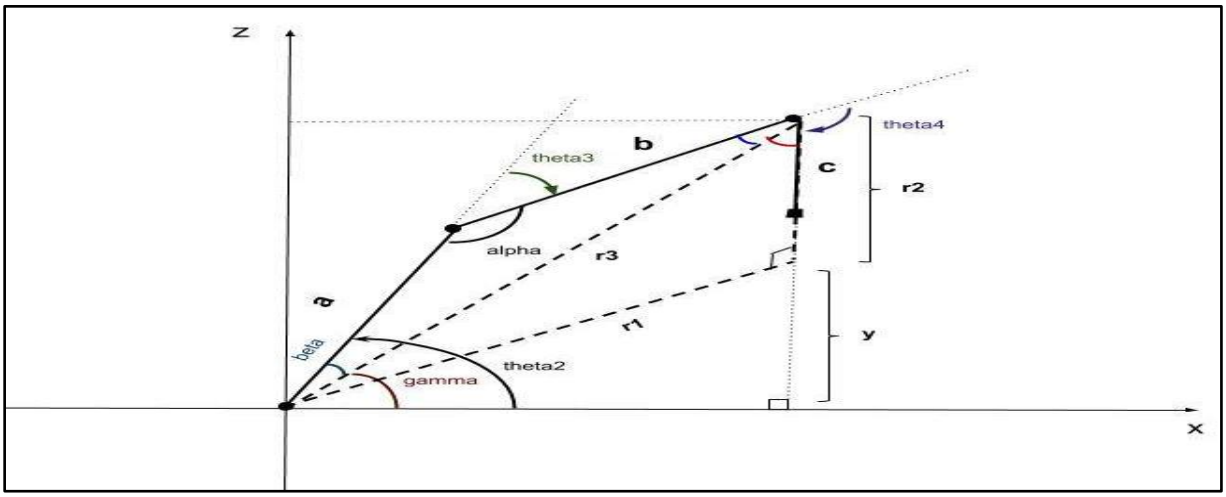


Figure 3-16: manipulateur de bras PUMA de quatre ddl

A partir de cette figure nous allons déterminer ces quatre équations en suivant ces étapes en premier le calcule de Θ_1 comme suit :

$$\boldsymbol{\Theta_1}=\tan^{-1}\left(\frac{y}{x}\right) \dots (3.4)$$

Nous avons Θ_2 donnée par :

$$\Theta_2= \alpha+\beta \dots (3.5)$$

Puis on calcule α et β représenter par les équations suivantes :

$$\alpha = \tan^{-1}\left(\frac{r2}{r1}\right)=\tan^{-1}\left(\frac{z-d1+a4}{\sqrt{x^2+y^2}}\right) \dots (3.6)$$

Avec :

$$\mathbf{r_1}=\sqrt{x^2+y^2} \dots (3.9), \qquad \mathbf{r2}= z-d1+a4 \dots (3.8)$$

Nous calculons β à partir de la loi de cosinus :

$$b^2 = a^2 + r_3^2 - 2*a*r_3*\cos(\beta) \dots (3.9)$$

Nous obtenons :

$$\beta = \cos^{-1}\left(\frac{a^2+r3^2-b^2}{2*a*r3}\right) \dots (3.10)$$

On remplace α et β on trouve Θ_2 égale :

$$\boldsymbol{\Theta_2} = \tan^{-1}\left(\frac{z-d}{\sqrt{x^2+y^2}}\right) + \cos^{-1}\left(\frac{a^2+r3^2-b^2}{2*a*r3}\right) \dots (3.5)$$

On a Θ3 égale :

$$\Theta_3 = \pi+\gamma \dots (3.13)$$

On a:

$$\gamma = \cos^{-1}\left(\frac{a^2+b^2-r3^2}{2*a*b}\right) \dots (3.12)$$

Avec :

$$\mathbf{r_3} = \sqrt{r1^2 + r2^2} \dots (3.13)$$

Alors :

$$\boldsymbol{\Theta_3} = \pi + \cos^{-1}\left(\frac{a^2+b^2-r3^2}{2*a*b}\right) \dots (3.11)$$

On a Θ4 :

$$\boldsymbol{\Theta_4} = \pi-(\pi- \cos^{-1}\left(\frac{b^2+r3^2-a^2}{2*a*r3}\right))-(\pi/2-\text{atan2}\ (z-d_1+a_4, r_1) \dots (3.14)$$

Finalement nous obtenons :

$$\boldsymbol{\Theta_1} = \tan^{-1}\left(\frac{y}{x}\right) \dots (3.4)$$

$$\boldsymbol{\Theta_2} = \tan^{-1}\left(\frac{z-d}{\sqrt{x^2+y^2}}\right) + \cos^{-1}\left(\frac{a^2+r3^2-b^2}{2*a*r3}\right) \dots (3.5)$$

$$\boldsymbol{\Theta_3} = -\pi + \cos^{-1}\left(\frac{a^2+b^2-r3^2}{2*a*b}\right) \dots (3.11)$$

$$\boldsymbol{\Theta_4} = \pi-(\pi- \cos^{-1}\left(\frac{b^2+r3^2-a^2}{2*b*r3}\right))-(\pi/2-\text{atan2}\ (z-d_1+a_4, r_1) \dots (3.14)$$

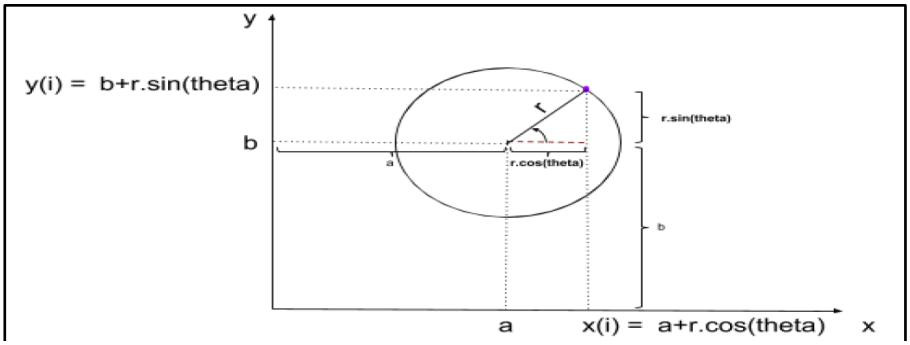


Figure 3-17: Modèle mathématique d'une trajectoire circulaire

[1] Le bout du bras dessine un cercle sur une feuille de papier, alors pour cela il faut deux paramètres, un diamètre et coordonnée centrale comme entrée (tant qu'ils sont à l'intérieur de l'enveloppe de travail nous les choisissons librement) puis on obtient les coordonnées (x, y) de chaque point du cercle en tant que sorties.

3.4 La transformation homogène du repère de la caméra vers le bras

Le repère de la caméra et différent de celui du bras, donc pour que le bras puisse attraper la balle il faut qui reçoit les bonne coordonnées et pour cela nous devons faire une transformation homogène à partie d'une matrice caméra vers une matrice bras, bien sur sans oublier le repère de base.

Nous avons crée un programme sur Matlab pour illustrer les repères dans une figure comme suit :

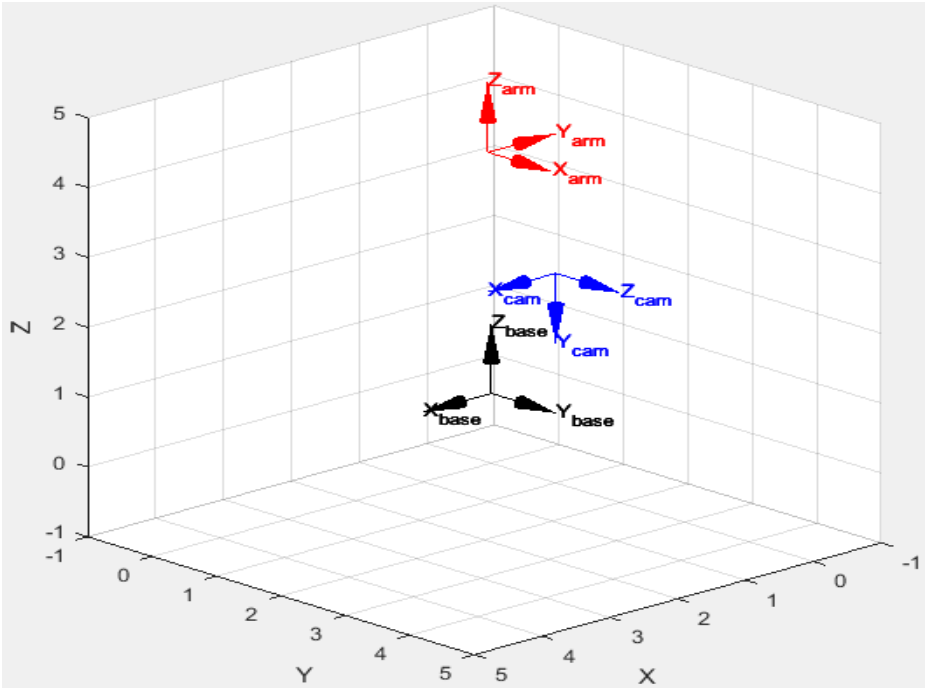


Figure 3-18 : Les repères (base, caméra, bras)

Alors nous avons en premier temps le passage du repère base (H_0) vers le repère caméra (H_{cam}), comme vous constater sur la figure suivante qu'il y a une rotation autour de l'axe X et une translation de 1 sur l'axe Y et de 2 sur l'axe Z:

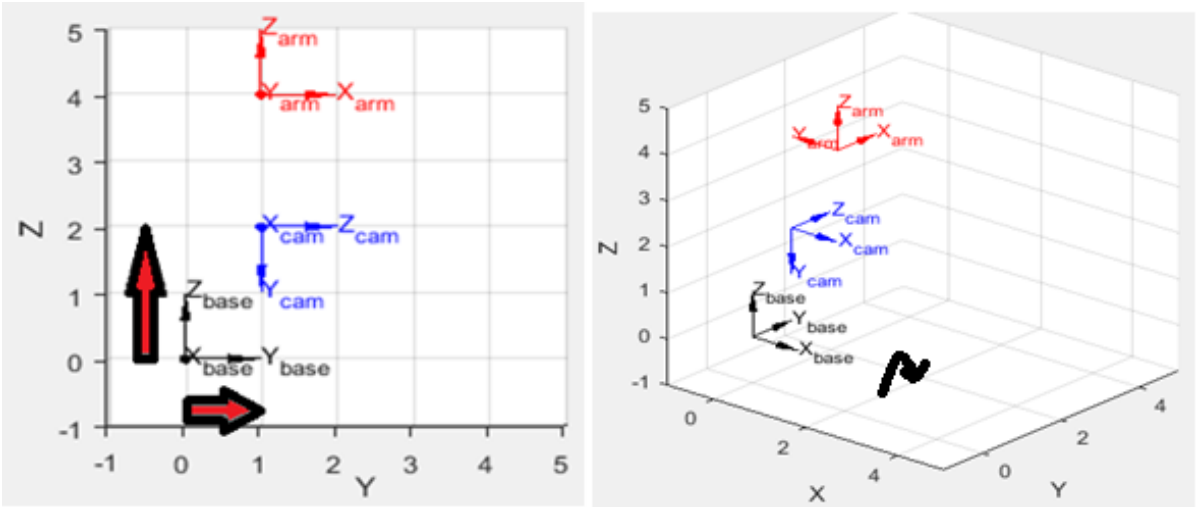


Figure 3-19 : La transformation du repère base vers repère caméra

Nous avons la matrice H^0 et nous obtenons la matrice H^{cam} :

$H^{cam} = H^0 *(Trans_z (2)*Trans_{Y(} (1)*Rot_x (-Pi/2))$

$$H^{cam} = H^0 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * Tranl \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} * Rot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La même chose pour le repère bras (H^{arm}), nous l'obtenons d'après le repère de caméra (H^{cam}) nous avons une rotation sur l'axe Y et sur l'axe X puis une translation sur l'axe Y et l'axe Z:

$H^{arm} = H^{cam} *(Rot_Y (-Pi/2) * Rot_x (Pi/2)* Trans_Y (-1)*Trans_Z (2)*)$

$$H^{arm} = H^{cam} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} * Rot_Y \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * Rot_x \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * Tranl \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H^{arm} = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Alors ici nous avons un exemple sur un point fixe, nous introduisons ces coordonnées sur Matlab comme un vecteur P= (X=1, Y= 2, Z= 2, 1) par rapport au repère base.

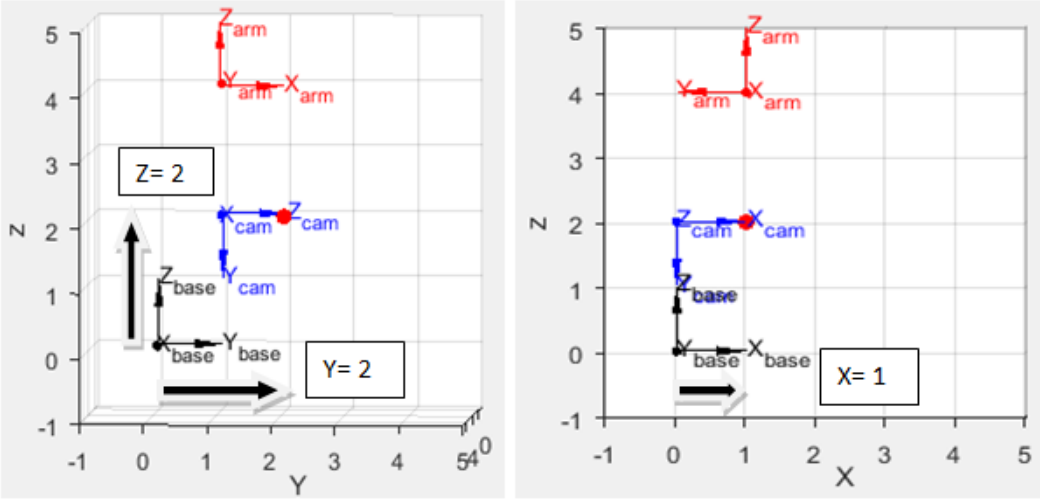


Figure 3-20 : Présentation du vecteur d'un point P par rapport au repère base

Les coordonnées de ce point fixe par rapport au repère base, caméra et bras sont différents. Pour cela nous devons trouver les bonnes coordonnées de chaque repère.

Pour le repère caméra nous avons un vecteur P^C come suit :

$P^C = H_0^C * P^0$ et $H_0^C = (H_C^0)^{-1}$

$Pc = ((H^{cam})^{-1} * P) = (1, 0, 1, 1)$

Nous allons comparer les résultats du vecteur Pc =(X=1, Y=0, Z=1, 1) avec la figure de Matlab:

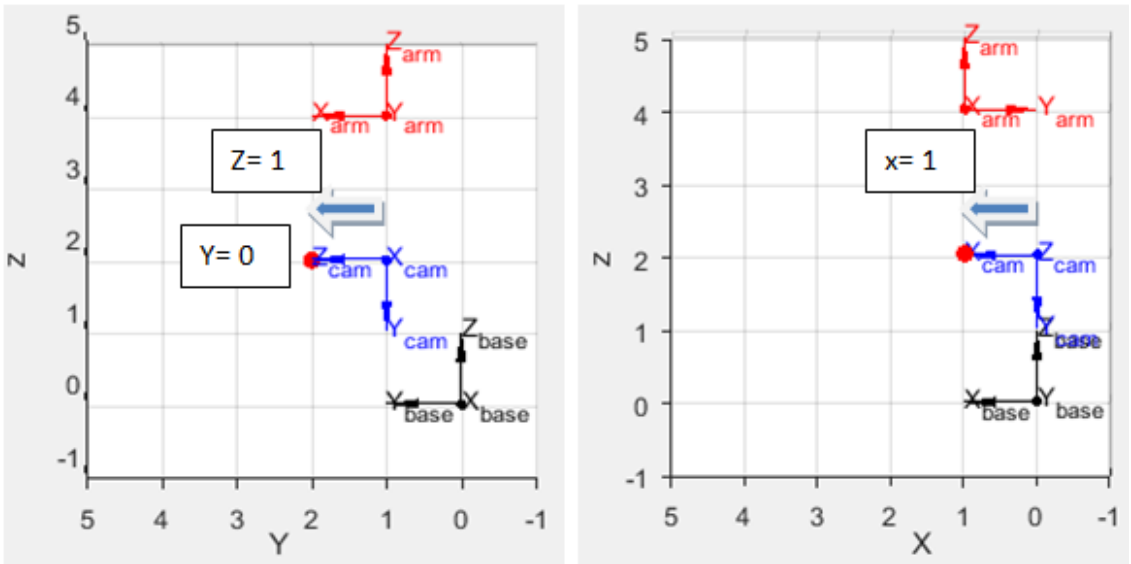


Figure 3-21 : Présentation du vecteur d'un point P par rapport au repère caméra

Pour le repère bras nous avons un vecteur P^B come suit :

$P^B = H_0^B * P^0$ et $H_0^C = (H_C^0)^{-1}$
$Pc = ((H^{arm})^{-1} * P) = (1, 0, -2, 1)$

Nous allons comparer les résultats du vecteur Pc =(X=1, Y=0, Z=-2, 1) :

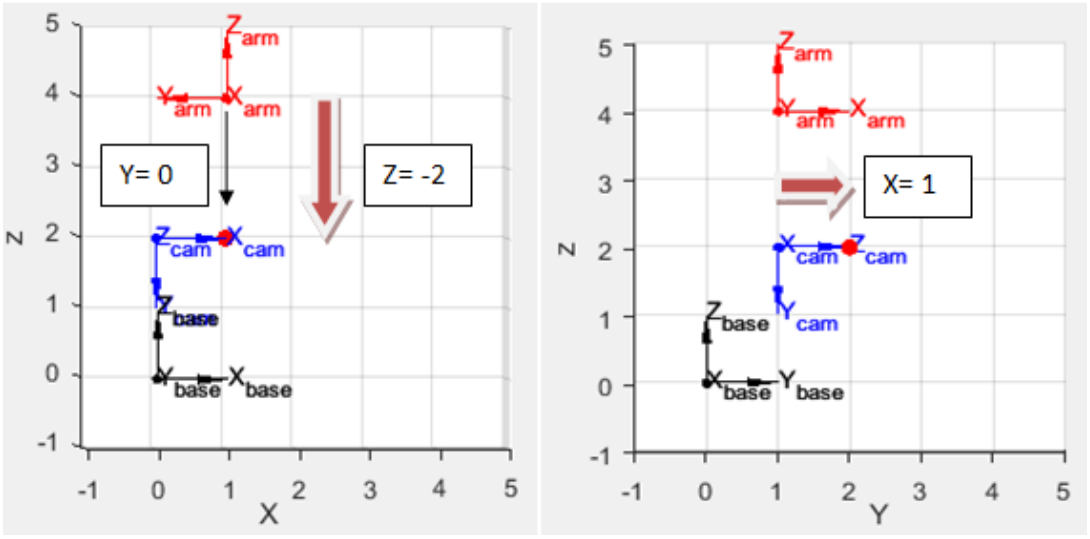


Figure 3-22 : Présentation du vecteur d'un point P par rapport au repère caméra

3.5 Conclusion

Dans ce chapitre nous avons représenté les différents types de robots manipulations d'une manière générale, commençant par la description des modèles cinématique directe et inverse qui ont été décrits d'une manière générale. Ensuite nous avons bien expliqué la transformation homogène pour mieux comprendre le changement du repère de la base vers le repère caméra puis vers le bras car ils sont différents. Dans le chapitre qui suit nous allons définir les matériels de notre projet et les logiciels que nous avons utilisé.

Chapitre 4 :
Présentation Générale du Projet

Chapitre 4 : Présentation Générale du Projet

4.1 Introduction

Nous avons ici un robot manipulateur qui doit être capable en premier temps à l'aide d'une caméra de détecter une balle de sa couleur dans une table, puis la capturée.

Maintenant il est temps de réaliser notre projet après avoir connus nos objectifs principaux. Dans ce qui suit on va présenter les différents composants et logiciels choisit pour sa réalisation.

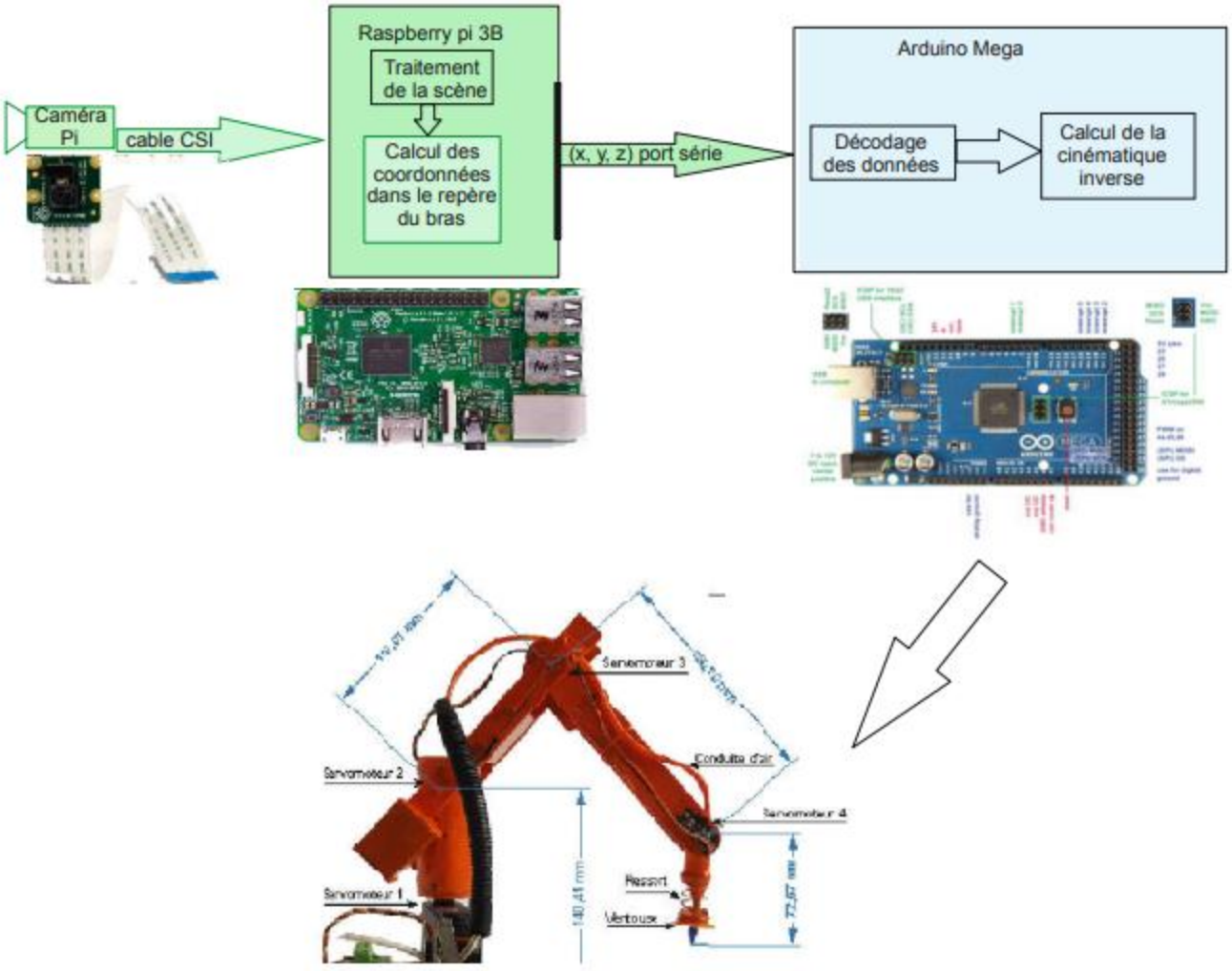


Figure 4-1 : Synoptique globale du projet

4.2 Les matériels utilisés

Notre choix des composants a été fait celant la disponibilité, le cout, tout en respectant les objectifs principaux du projet.

Figure 4-2 : Le robot manipulateur

4.2.1 L'alimentation électrique

Dans ce projet, on utilise une batterie de 12V, car le choix varié et dépend de la consommation énergétique des différents éléments intégrer, parce qu'elle se charge d'alimenter les différents moteurs, servomoteurs, cartes électroniques

4.2.2 Le moteur électrique

On a utilisé un moteur de type DC, pour la venteuse qui attrape la balle

Figure 4-3: Le moteur à courant continu

4.2.3 Les servomoteurs

Figure 4-4: Servomoteurs utilisés dans le bras manipulateur

4.2.4 Les cartes électroniques programmables

La mise en relation électrique des composants électroniques cités est importante, pour réaliser les opérations principales demandées. Alors il faut l'implémentation des cartes électroniques obligatoirement.

4.2.4.1 Le Raspberry Pi

Raspberry Pi est un ordinateur miniature, mono carte entièrement fonctionnel, Cet ordinateur, de la taille d'une carte de crédit, est destiné à encourager l'apprentissage de la programmation informatique, il permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux, notamment Debian, et des logiciels compatibles. Mais il fonctionne également avec l'OS Microsoft Windows : Windows 10 IoT Coreet celui de Google, AndroidPi. Le Raspberry Pi est contenu sur un seul circuit imprimé et dispose de ports pour :

- HDMI
- USB 2.0
- Vidéo composite
- Audio analogique
- Puissance
- L'internet
- Carte SD

L'ordinateur fonctionne entièrement sur un logiciel open-source et donne aux utilisateurs la possibilité de mélanger et d'associer des logiciels en fonction du travail qu'ils souhaitent effectuer.

Depuis la première version sortie en 2011, Raspberry Pi n'a cessé d'évoluer jusqu'à présent avec des différentes versions (A, B, B+, zéro et Pi3) maintenant il est l'un des plis célèbre ordinateur du monde.

Dans notre projet, on utilise la version Raspberry Pi 3 model B.

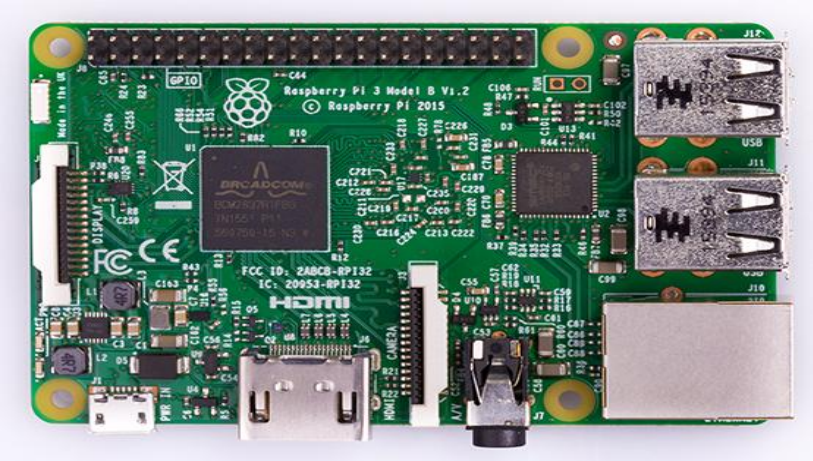


Figure 4-5: La carte Raspberry Pi 3 model B

Paramètres	Raspberry Pi model B
Sortie	29/02/2016
SoC	BCM2837
CPU	Quad Cortex A53 @ 1.2GHZ
GPU	400MHZ VideoCore lV
RAM	1 GB SDRAM
Stockage	Micro-SD
Ethernet	10/100
wireless	802.11n / Bluetooth 4.0
Sortie vidéo	HDMI/Composite
Sortie audio	HDMI / Headphone
GPIO	40
Instruction set	ARM v8- A

Tableau 4-1: Les caractéristiques du Raspberry Pi 3 modèle B

4.2.4.1.1 La caméra Raspberry Pi

Le module caméra Raspberry Pi est un produit officiel de la fondation Raspberry Pi. Le modèle a été lancé en 2013. Il se branche directement sur le connecteur CSI du Raspberry Pi.

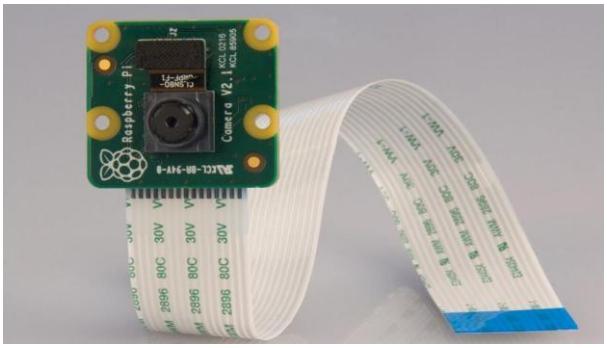


Figure 4-6: Le module camera Rraspberry Pi3 V 1.3

Paramètres	Gamme
La taille et poids	Environ 25 x 24 x 9mm, 3g
Touche résolution	5 Méga pixels
Modes vidéo	1080p30, 720p60 et 640 × 480p60 / 90
Capteur	Omni Vision OV5647
Résolution de l'image	2592 x 1944
Interface	CSI
Compatibilité	Entièrement compatible

Tableau 4-2: Les caractéristiques du module caméra Raspberry Pi v 1.3

4.2.4.1.2 Les pins (GPIO) de Raspberry Pi 3 modèle B

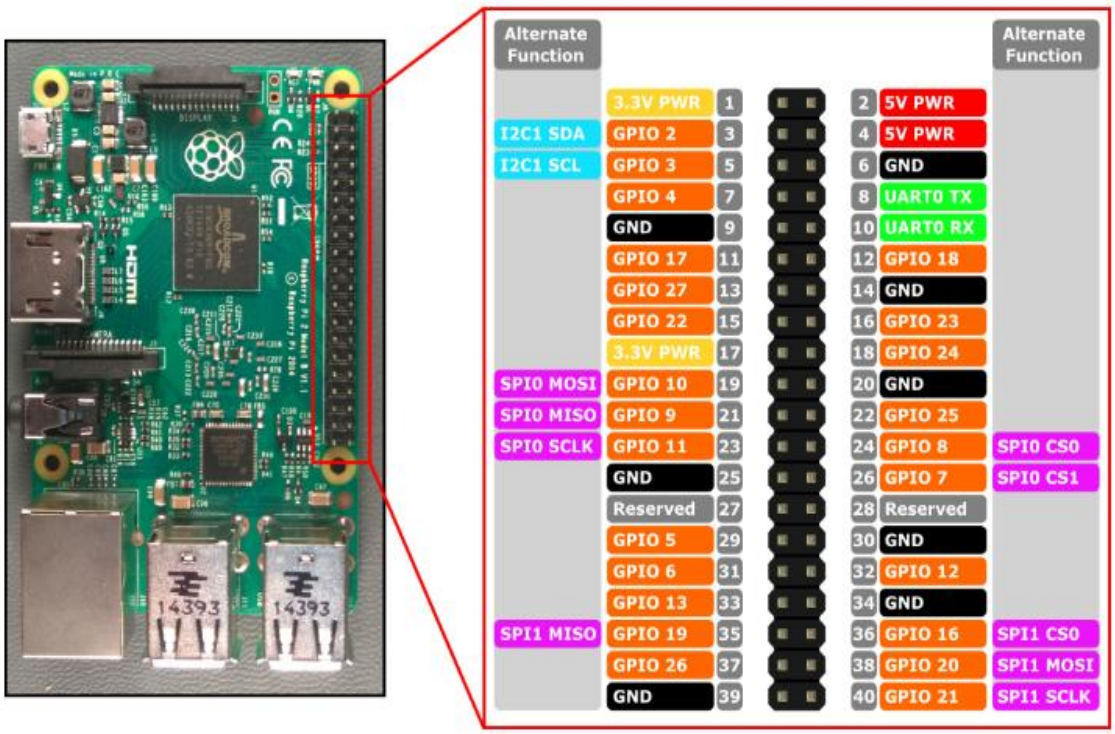


Figure 4-7: Les GPIO de la carte Raspberry Pi 3

4.2.4.2 la carte Arduino

Arduino fait référence à une plate-forme ou à un tableau électronique à source ouverte et au logiciel utilisé pour le programmer. Arduino est conçu pour rendre l'électronique plus accessible aux artistes, concepteurs, amateurs et à tous ceux qui sont intéressés par la création d'objets ou d'environnements interactifs. Une carte Arduino peut être achetée pré-assemblée, car la conception matérielle est open source, construite à la main. Dans les deux cas, les utilisateurs peuvent adapter les cartes à leurs besoins, mettre à jour et distribuer leurs propres versions.

Une carte Arduino pré-assemblée comprend un microcontrôleur programmé à l'aide du langage de programmation Arduino et de l'environnement de développement Arduino. En substance, cette plate-forme offre un moyen de créer et de programmer des composants électroniques. Le langage de programmation Arduino est un langage simplifié à partir du langage de programmation C / C ++ basé sur ce que Arduino appelle des "esquisses", qui utilisent des structures de programmation, des variables et des fonctions de base. Celles-ci sont ensuite converties en un programme C ++.

On peut l'associer à des capteurs (de lumière, de température, de position, …), des actionneurs (moteurs, pompe, …), des organes de sortie (lampe, chauffage, …), des circuits de puissance, une alimentation (piles, panneaux solaires, …) etc.

Il existe toute une gamme de carte Arduino chacune ses caractéristiques, utilisées selon le besoin de l'utilisateur. Dans ce projet on s'intéresse à la carte Arduino Méga et Uno:

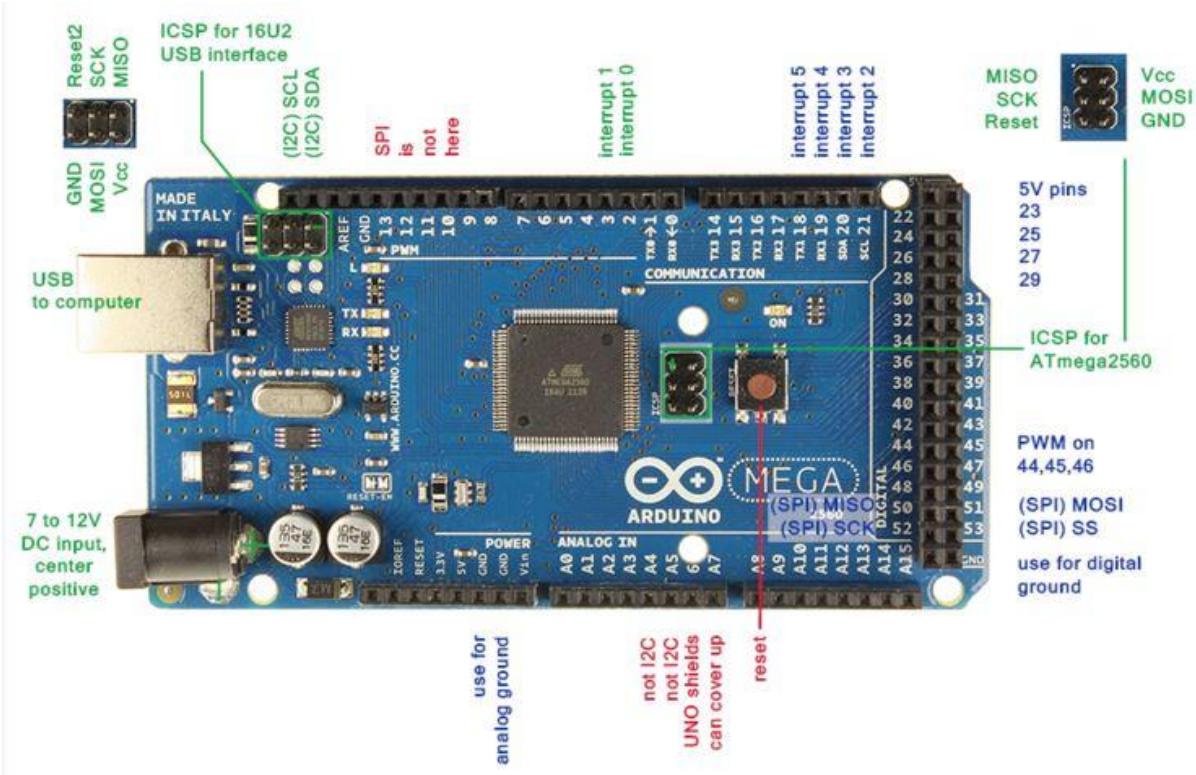


Figure 4-8: La carte Arduino Méga

Les caractéristiques techniques de la carte MEGA :

Microcontrôleur		ATmega2560
Tension de fonctionnement :		5 V
Tension d'entrés (limites)		7 à 12 V
Tension d'entrée (limites)		6 à 20 V
Broches E/S numériques		54 (dont 14 fournissent la sortie PWM)
Broches d'entrée analogiques		16
Courant alternatif par broche d'E/S		40 mA
Courant continu pour la broche de 3,3 V		50 mA
Mémoire Flash :		256 Ko (dont 8 Ko utilisés par le chargeur initial de programme)
SRAM		8 Ko
EEPROM :		4 Ko
Vitesse de l'horloge		16 MHz

Tableau 4-3: Les caractéristiques techniques de carte Arduino Méga

4.2.5 Carte de puissance

Nous avons réalisé une carte de puissance spécifique à notre projet. Cette dernière dispose d'un pont en H et de toutes les connexions nécessaires lui permettant d'être branchée directement sur une carte de type Arduino UNO ou Méga. Une carte de liaison servant d'interface entre le pont en H et le module Arduino dispose de quatre connecteurs mâles spécialement dédiés au branchement des servomoteurs. Une des deux sorties du pont en H est utilisée pour alimenter la pompe à vide constituant l'organe terminal da bras. Et en la reliant aussi à une carte Raspberry Pi 3 par le sérial en connectant le GND, TX, RX pour la transmission des données. Les figures ci-dessous illustres le schéma électrique de l'ensemble.

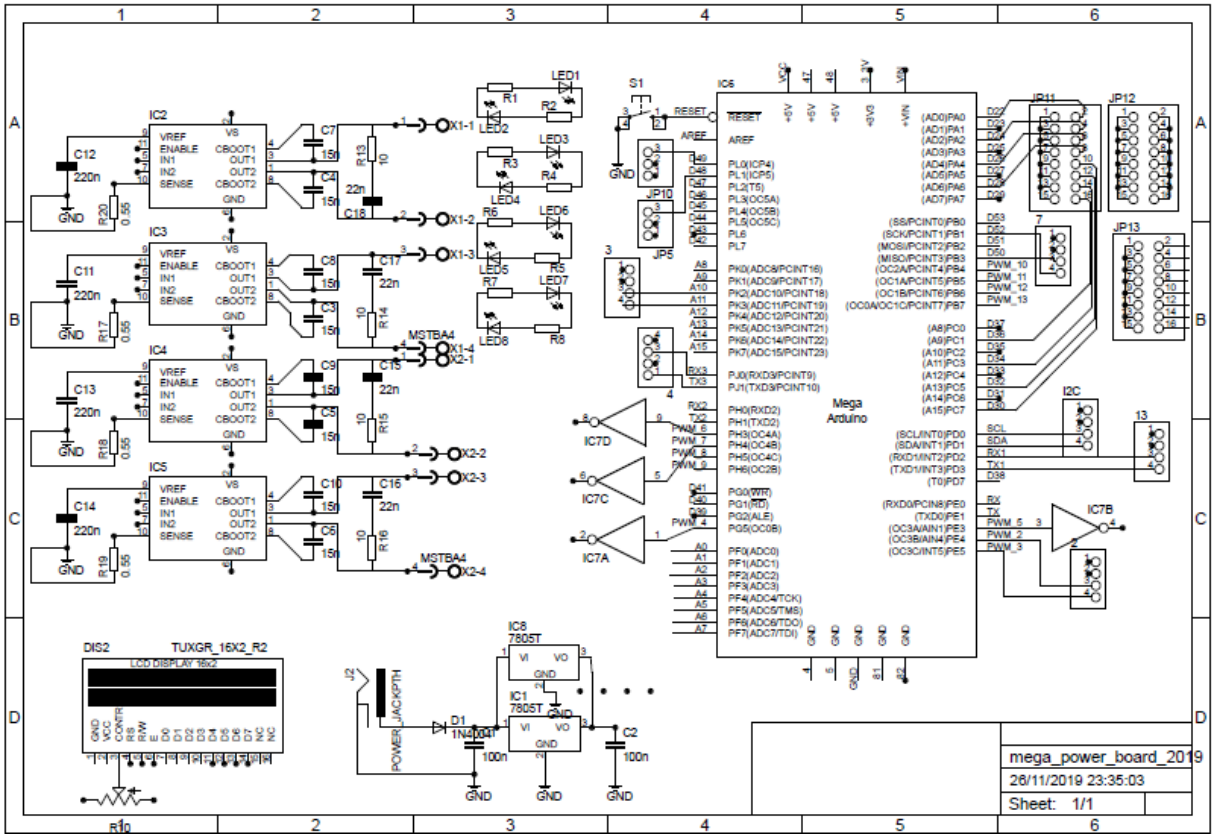


Figure 4-9 : Schéma électrique de la carte de puissance

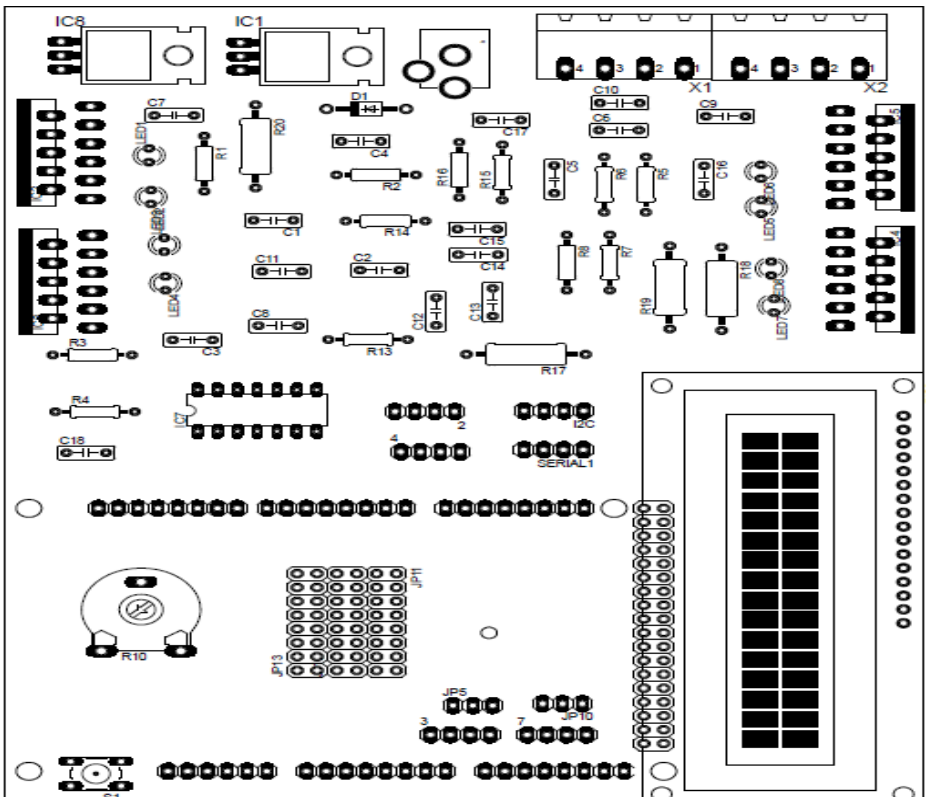


Figure 4-10 : Schéma électrique de la carte de puissance

4.3 Les logicielles

4.3.1 Le logicielle de la carte Raspberry Pi

4.3.1.1 Le langage programmation Python

Raspberry Pi fonctionne sous linux, alors n'importe quel langage est possible pour l'installer, mais finalement on a opté pour le python un langage lisible, simple et moins complexe, malgré que le langage compilé soit plus rapide qu'un langage interprété qui est le python, mais il peut s'adapter à tout type d'utilisation en l'utilisant dans de nombreux contextes.

4.3.1.2 OpenCV

OpenCV est une bibliothèque qui faut ajouter à ces langages de programmation et l'importer dans le programme, car ils ne disposent pas de fonctions de traitement des vidéos ni des photos.

OpenCV (Open Source Computer Vision Library), qui veut dire libre et spécialisée dans la vision par ordinateur, est une bibliothèque de logiciels open source de vision informatique et d'apprentissage automatique. OpenCV a été conçu pour fournir une infrastructure commune aux applications de vision par ordinateur et pour accélérer l'utilisation de la perception de la machine dans les produits commerciaux. OpenCV étant un produit sous licence BSD, il est facile pour les entreprises d'utiliser et de modifier le code.

La bibliothèque contient plus de 2500 algorithmes optimisés, qui incluent un ensemble complet d'algorithmes classiques et avancés de vision par ordinateur et d'apprentissage automatique. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer les actions humaines dans des vidéos, suivre les mouvements d'une caméra, suivre des objets en mouvement, extraire des modèles 3D d'objet, supprimer les yeux rouges des images prises au flash, suivre les mouvements des yeux, reconnaître les paysages et établir des repères pour les superposer à la réalité augmentée, etc.

OpenCV compte plus de 47 000 utilisateurs communauté et le nombre de téléchargements estimé à plus de 14 millions. La bibliothèque est largement utilisée dans les entreprises.

Il possède des interfaces C ++, Python, Java et MATLAB et prend en charge Windows, Linux, Android et Mac OS. OpenCV s'appuie principalement sur les applications de vision en temps réel et tire parti des instructions MMX et SSE.

La bibliothèque OPENCV sera utilisée dans notre projet pour détecter, suivre une couleur en mouvement en utilisant le module camera de la carte Raspberry.

4.3.2 Le logicielle de la carte Arduino

La carte Arduino n'a pas le choix de choisir le langage pour la programmer, on doit installer l'IDE Arduino sur notre ordinateur. Il fonctionne dans des différentes plates-formes Windows, Linux et Mac.

Le langage Arduino et proche du C/C++, avec des bases faciles à accès qui demande très peu de ressources, il nous permet de le programmer la carte via l'interface USB.

4.3.3 Le logicielle Matlab

4.3.3.1 La toolbox rvctools

La toolbox rvctool c'est un ensemble de routines développées sous Matlab, pour être exploitées dans le domaine de la robotique et la vision artificielle. Cet ensemble contient des fonctions utiles pour l'étude et la simulation d'un très Grand nombre de robots y compris leurs modèles cinématique, dynamique ainsi la génération de trajectoires.

L'utilisation du RVC Tools offre un grand nombre de fonctions exploitables à partir de l'environnement Matlab, pour permettre d'effectuer des calculs matriciels et surtout faciliter la représentation graphique des opérations relatives aux transformations rigides, nécessaire à l'étude des mouvements des articulations rotoides et prismatique dont sont munis les robots manipulateurs.

4.4 Communication entre les deux cartes Raspberry Pi et Arduino

4.4.1 Le protocole I²C

Pour profiter des avantages de chacune de ces cartes, on a pensé de faire travailler les deux ensembles en configurant le Raspberry Pi en tant que maitre et Arduino comme esclave.

Ils pourront communiquer de manière bidirectionnelle (half-dublex) les en relient avec un bus I²C. Semi duplex ou parfois appelée liaison à l'alternat, est une liaison dans laquelle les données circulent dans un sens ou l'autre mais pas les deux simultanément.

La transmission des données s'effectuera avec des signaux transportés soit directement en signale numérique comme dans les réseaux de données ou bien analogique mais après l'avoir converti sous forme numérique. Cette transmission est caractérisée par la synchronisation, le mode de transmission des données et le sens de leurs échanges.

4.4.1.1 Le sens des échanges de données

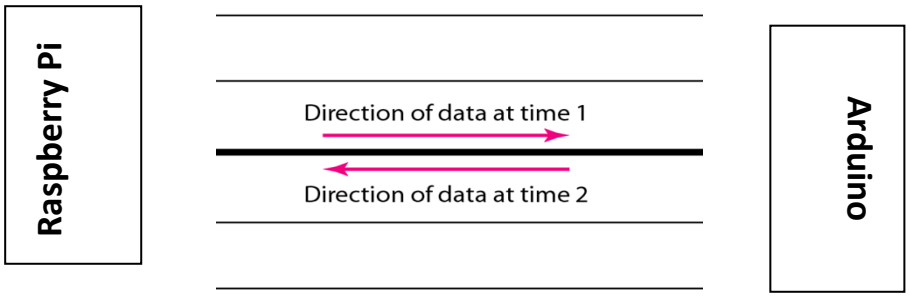


Figure 4-11: Le mode Half-Dublex

4.4.1.2 Transmission série et parallèle

On désigne par liaison parallèle la transmission simultanée de *N* bits. Ces bits sont envoyés simultanément sur N voies différentes (une voie étant par exemple un fil, un câble ou tout autre support physique).

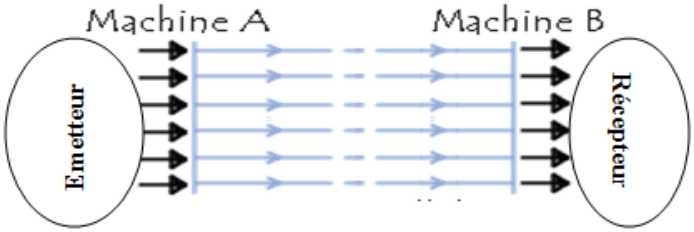


Figure 4-12: Transmission parallèle

Dans une liaison en série, les données sont envoyées bit par bit sur la voie de transmission. Toutefois, étant donné que la plupart des processeurs traitent les informations de façon parallèle, il s'agit de transformer des données arrivant de façon parallèle en données en série au niveau de l'émetteur, et inversement au niveau du récepteur.

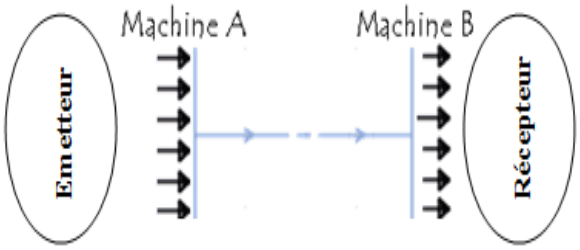


Figure 4-13: Transmission série

4.4.1.3 La synchronisation

Etant donné les problèmes que pose la liaison de type parallèle, c'est la liaison série qui est la plus utilisée. Toutefois, puisqu'un seul fil transporte l'information, il existe un problème de synchronisation entre l'émetteur et le récepteur, c'est-à-dire que le récepteur ne peut pas a priori distinguer les caractères (ou même de manière plus générale les séquences de bits) car les bits sont envoyés successivement. Il existe donc deux types de transmission permettant de remédier à ce problème :

La liaison asynchrone, Si qu'un seul bit soit transmis pendant une longue période de silence... le récepteur ne pourrait savoir s'il s'agit de 00010000, ou 10000000.

Pour éviter ce problème, chaque caractère est précédé d'une information indiquant le début de la transmission du caractère (bit *START*) et fin de transmission (bit *STOP*).

La liaison synchrone, dans laquelle émetteur et récepteur sont cadencés à la même horloge. Le récepteur reçoit de façon continue les informations au rythme où l'émetteur les envoie.

De plus, des informations supplémentaires sont insérées afin de garantir l'absence d'erreurs lors de la transmission.

4.4.1.4 Définition du protocole I²C

I²C : (Inter Integrated Circuit) développé au début des années 80, par Philips Semiconductor pour permettre de relier facilement à un microprocesseur les différents circuits d'un téléviseur moderne.

Son but est de faire la communication entre les composants électronique grâce à ces 3 fils :

- Signal de donnée : SDA (Serial DAta Line) c'est la ligne où passent les données dans les deux sens.
- Signal d'horloge : SCL (Serial CLock Line) c'est la ligne où passe le signal d'horloge pour la synchronisation.
- Signal de référence électrique : masse (GND) qui sert à référencer les deux signaux d'informations SCL et SDA.

Le protocole de la liaison est du type Maitre /Esclave comme on a pu citer précédemment qu'on va identifier la carte Raspberry Pi comme maitre et Arduino comme Esclave.

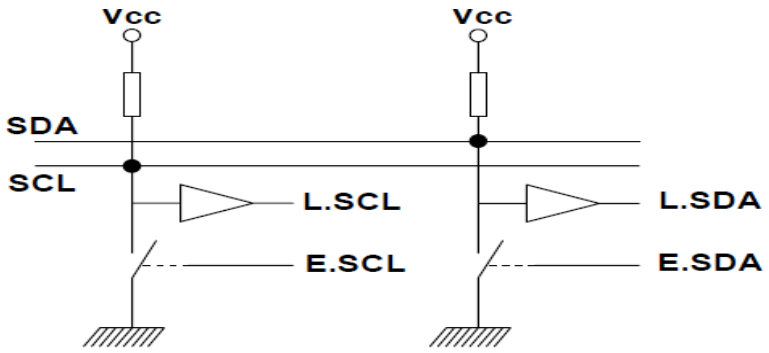


Figure 4-14: Entrées-Sortie de type collecteur ouvert

On peut forcer SDA et SCL uniquement au niveau bas et ne pas au niveau haut.

Chaque circuit (dans notre cas des Arduino) possède une adresse matérielle unique qui le distingue des autres sur 7 bits : possibilité de 128 adresses.

C'est un bus dit Maitre/esclave dans la mesure où tout échange sur le bus ne peut se faire qu'à l'initiative du maitre. Les données transitent exclusivement à la demande du maitre vers tous les esclaves (Arduino).

Avant de tenter de prendre le contrôle du bus I²C, un circuit doit vérifier que les lignes SDA et SCL sont au repos

Il faut surveiller la condition de départ : SDA passe à 0, SCL reste à 1 et la condition d'arrêt : SDA passe à 1, SCL reste à 1.

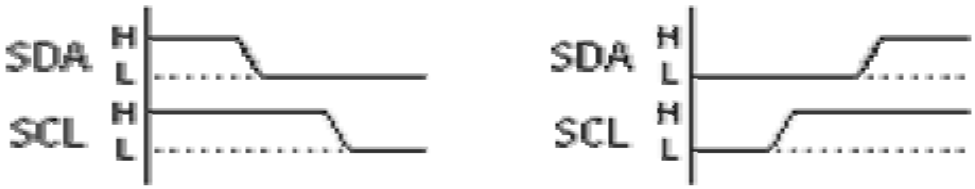
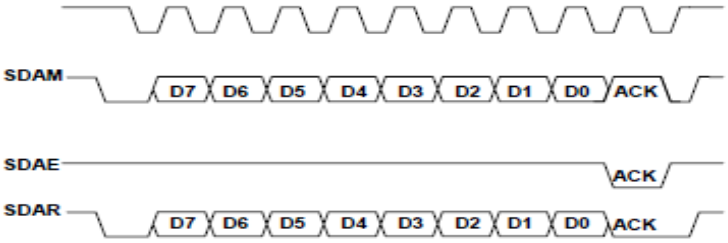


Figure 4-15 : Bit Start, Bit Stop



SCL : Horloge imposée par le maître
SDAM : Niveaux de SDA imposés par le maître
SDAE : Niveaux de SDA imposés par l'esclave
SDAR : Niveaux de SDA réels résultants

Figure 4-16 : Transmission d'un octet

Le maitre transmet le bit de poids fort D7 sur SDA puis il valide la donnée en appliquant un niveau 1 sur SCL, lorsque SCL retombe à 0, il poursuit avec D6 jusqu'à ce que l'octet complet soit transmis.

L'esclave doit imposer un niveau 0 pour signaler que la transmission s'est déroulée correctement puis quand le maitre voit 0 il passera à la suite.

Chaque octet de donnée transmis est suivi par l'émission d'un bit d'acquittement (ou ACK) qui sert d'un accusé de réception.

- ACK = 0 : Le maître (Raspberry Pi) peut continuer les opérations. (ACK)
- ACK = 1 : Une erreur de sélection le maitre génère la condition d'arrêt (NACK)

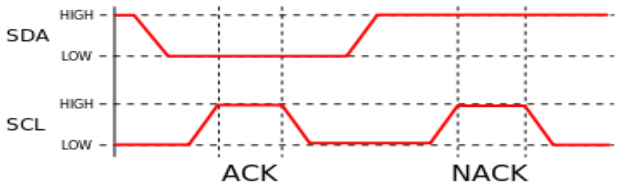


Figure 4-17: La condition d'acquittement ACK ou de non-acquittement NACK

4.4.1.5 Raccordement I²C

Il suffit juste de raccorder les broches SDA et SCL de ces cartes et partager la même masse pour crée un bus I²C

- SDA et SCL pour la carte Raspberry Pi sont les pins 3 et 5.

- SDA et SCL pour la carte Arduino sont les pins 20 et 21.

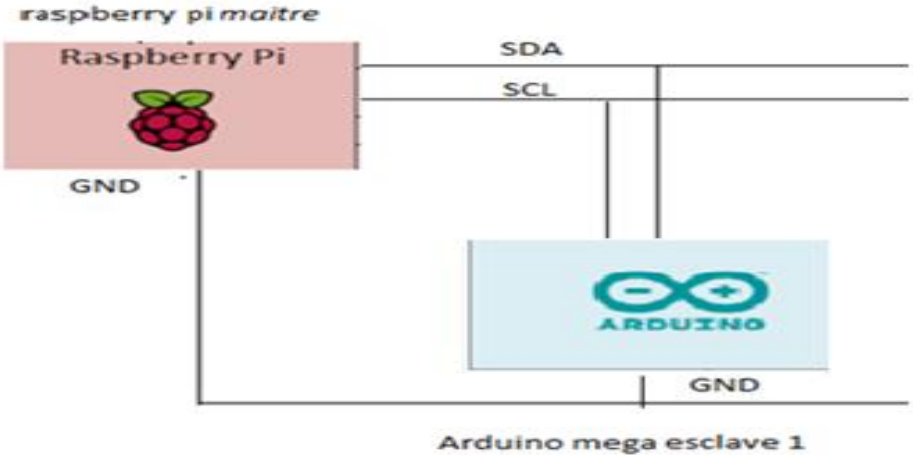


Figure 4-18: Raccordement I²C

4.4.2 Le protocole port série

En premier temps on a travaillé par la méthode i2c mais on a rencontré des difficultés lors des résultats du bras, donc on a opté pour ce protocole qui est le port série pour une meilleure fiabilité des résultats

Le protocole série asynchrone comporte un certain nombre de règles intégrées - des mécanismes permettant d'assurer des transferts de données robustes et sans erreur. Ces mécanismes, que nous obtenons pour éviter le signal d'horloge externe, sont les suivants:

- Bits de données,
- Bits de synchronisation,
- Bits de parité
- Taux de bauds

Grâce à la variété de ces mécanismes de signalisation, vous constaterez qu'il n'existe pas un seul moyen d'envoyer des données en série. Le protocole est hautement configurable. La partie critique est de s'assurer que les deux périphériques sur un bus série sont configurés pour utiliser exactement les mêmes protocoles.

4.4.2.1 Débit en bauds

Le débit en bauds spécifie **la rapidité avec laquelle les** données sont envoyées sur une ligne série. Il est généralement exprimé en unités de bits par seconde (bps).

Les taux de bauds peuvent être n'importe quelle valeur raisonnable. La seule exigence est que les deux appareils fonctionnent au même taux. L'un des débits en bauds les plus courants, en particulier pour les choses simples où la vitesse n'est pas critique, est de **9600 bps**. Les autres bauds "standard" sont 1200, 2400, 4800, 19200, 38400, 57600 et 115200.

4.4.2.2 Les bits des données, synchronisation et de parité

Chaque bloc (généralement un octet) de données transmises est en réalité envoyé dans un paquet ou une trame de bits. Les cadres sont créés en ajoutant des bits de synchronisation et de parité à nos données.

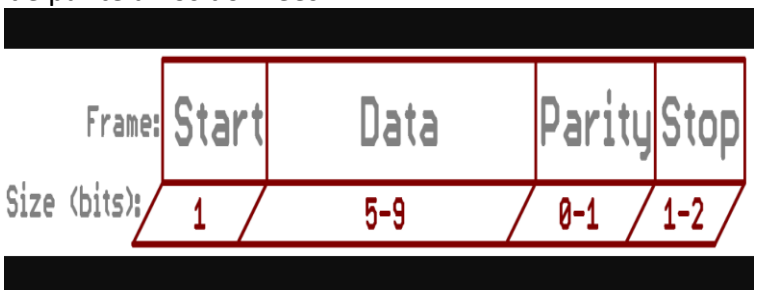


Figure 4-19: Encadrer les données

Les données sont transmises par un bus de 8 bits en fonctionnant de bits de synchronisation qui sont deux ou trois bits spéciaux transférés avec chaque bloc de données. Ce sont le bit de départ et le (Start) et le bit (Stop) d'arrêt.

Le bit de départ est toujours indiqué par une ligne de données inactive allant de 1 à 0, tandis que le ou les bits d'arrêt repassent à l'état inactif en maintenant la ligne à 1.

La parité est une forme de vérification d'erreur très simple et de bas niveau. Il vient en deux saveurs : impair ou pair.

En fait, pour chaque octet de données transmis, 10 bits sont envoyés : un bit de début, 8 bits de données et un bit d'arrêt. Ainsi, à 9 600 bits / s, nous envoyons en réalité 9 600 bits par seconde ou 960 (9 600/10) octets par seconde.

4.4.2.3 Câblage

Un bus série ne comprend que deux fils : un pour l'envoi de données et un autre pour la réception. En tant que tels, les périphériques série doivent avoir deux broches série : le récepteur, **RX**, et le transmetteur, **TX**.

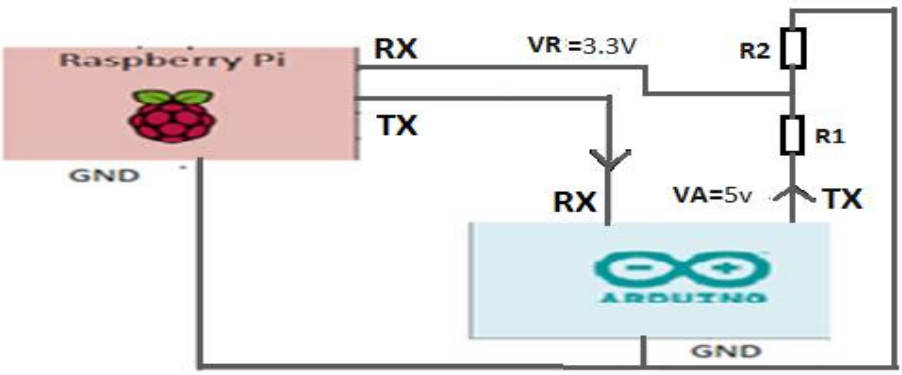


Figure 4-20 : Câblage d'un bus série

Comment il transmettre les données via un bus :

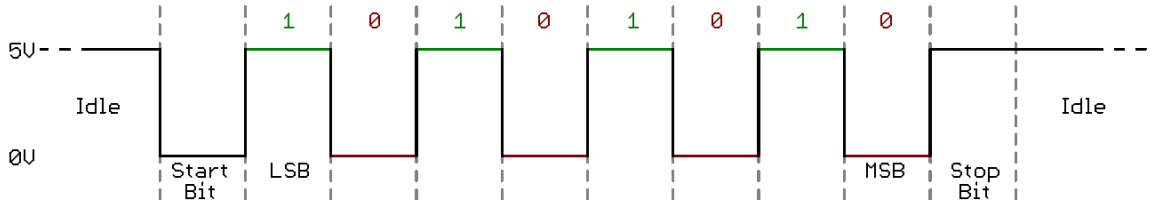


Figure 4-21 : Transmission des données via un port série

Des signaux série **TTL** (logique transistor-transistor) existent entre la plage d'alimentation en tension d'un microcontrôleur - généralement de 0V à 3,3V ou 5V. Un signal au niveau VCC (3,3 V, 5 V, etc.) indique une ligne inactive, un bit de valeur 1 ou un bit d'arrêt. Un signal 0V (GND) représente un bit de début ou un bit de données de valeur 0.

4.5 Conclusion

Dans ce chapitre on a représenté les différents matériels utilisés pour réaliser notre robot d'une manière générale, après nous avons expliqué comment se fais la transmission des coordonnées de la balle bleu par un bus i2c et ensuit par port série de Raspberry pi vers Arduino. Et dans le dernier chapitre nous allons vous montrer les différents résultats que nous avons obtenus.

Chapitre 5 :
Les résultats du projet

5.1 Introduction

Comme mentionné dans le chapitre précédent, les taches du robot manipulateur mobile se divisent en deux parties essentielles :

- La détection et le suivi d'objet de couleur bleu
- Le bras déplace cet objet d'un point P_1 de coordonnées (X_1, Y_1, Z_1) vers un point P_2 de coordonnées (X_2, Y_2, Z_2).

Nous allons montrer les résultats trouvés et les méthodes utilisées dans ce dernier chapitre.

5.2 Détection et suivi d'objet de couleur

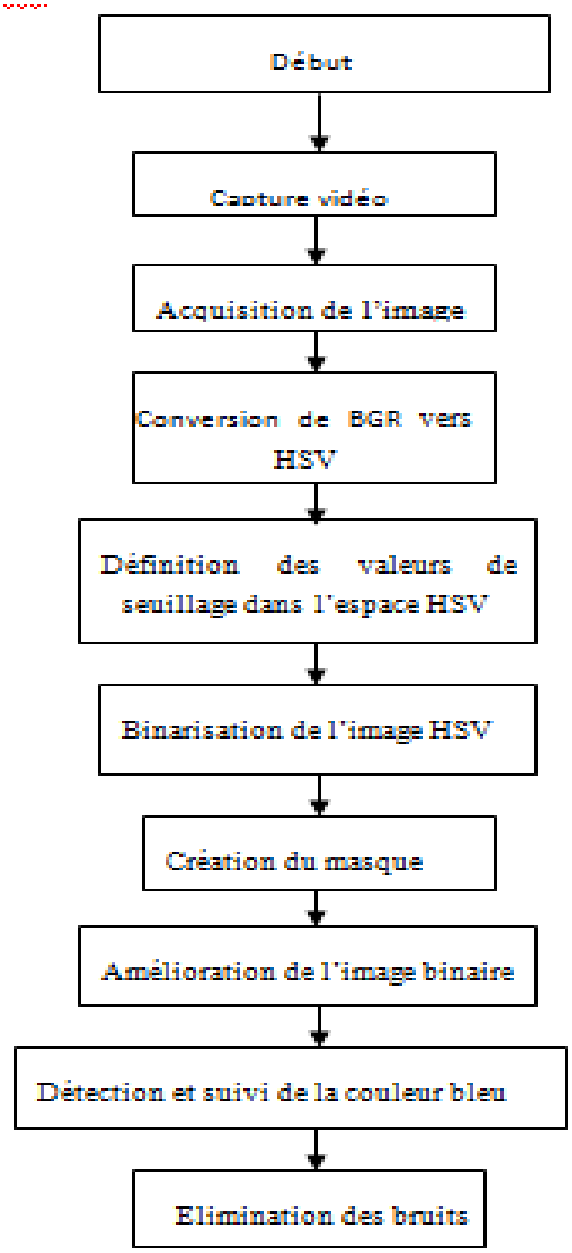
Plusieurs méthodes existent pour détection et suivi d'objet, on choisi la plus simple qui la détection d'objet selon sa couleur mais cela il faut connaitre les bordures inferieure et supérieures de cette couleur en utilisant un programme python avec bibliothèque OpenCV sur Raspberry Pi.

5.2.1 Détection d'objet avec OpenCV_Python

Les humains reconnaitre et détecte la couleur sans faire d'effort tout en contraire pour l'ordinateur, car la caméra Raspberry PI ne fait qu'envoyer des données brutes suit à ça, elles vont être lues et traitées d'une manière différente pour pouvoir enfin détecter la couleur voulue.

Nous allons effectuer une binarisation des pixels de la couleur bleu sur les images parce qu'on a qu'une seule couleur à détecter, c'est pour cela nous allons diminuer la quantité d'informations présentent dans l'image et garder que les pertinentes.

5.2.1.1 Organigramme de la détection de couleur



Organigramme 4.1 : Organigramme de la détection de couleur

5.2.1.2 Les résultats obtenus

Il à détecter que l'objet bleu entre les deux couleurs comme illustre a figure suivant :

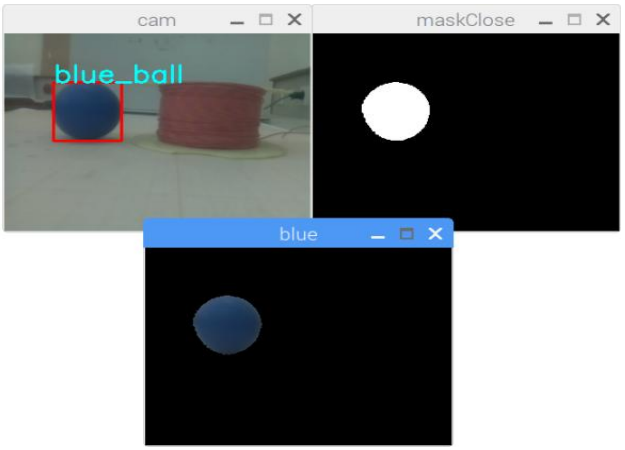


Figure 5-1: Détection d'objet selon ça couleur dans une image, et dans une image binaire (mask , mask bleu)

Malgré la détection d'objet en fonction de sa couleur on a rencontré plusieurs problèmes :

- la difficulté pour trouver l'intervalle des paramètres HSV de la couleur voulue.
- lorsque la taille des objets est similaire, la détection devient compliquée
- l'éclairage à un grand rôle pour la détection de couleur car les résultats peuvent changer au changement de l'éclairage
- l'image binaire (le mask) est légèrement bruitée. Sauf si on élimine les pixels en blanc éparpillés dans l'image qui n'appartiennent pas à l'objet et faire le contraire avec les pixels de ceux qui appartiennent pour avoir une meilleure précision.

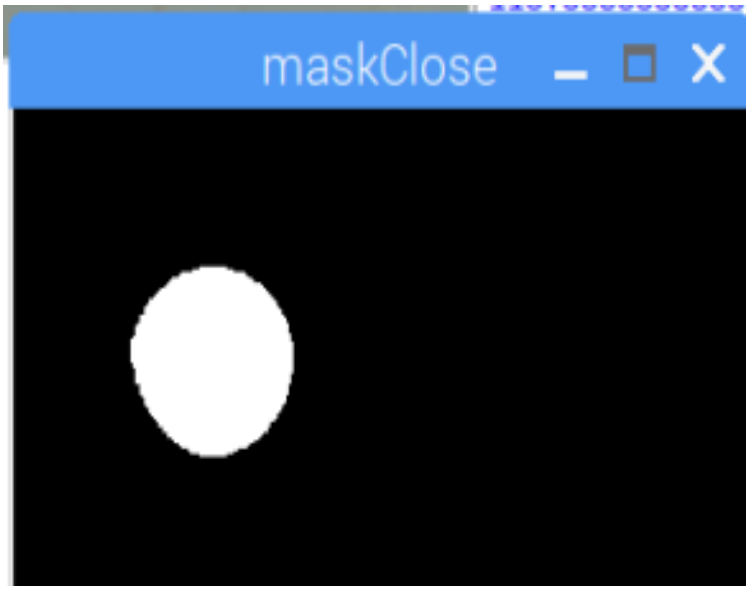


Figure 5-2: L'amélioration de l'image binaire

5.2.2 Calcule de la distance d'objet

On calculer la distance de la balle bleue manuellement en notent sa largeur à chaque fois, puis on effectuer sous Matlab un programme qui nous permet de trouve l'équation de cette opération, ensuite on fait rentrer cette équation d'ordre 5 sur python pour ce dernier puisse calculer la distance automatiquement quand il détecte la balle

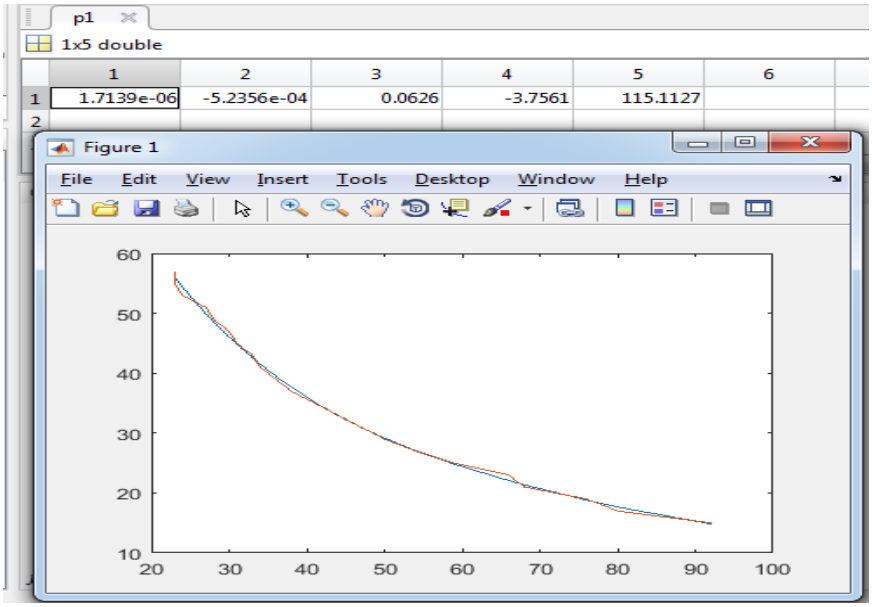


Figure 5-3: Simulation sous Matlab pour le polynôme

Les résultats obtenus par la carte Raspberry Pi comme suit :

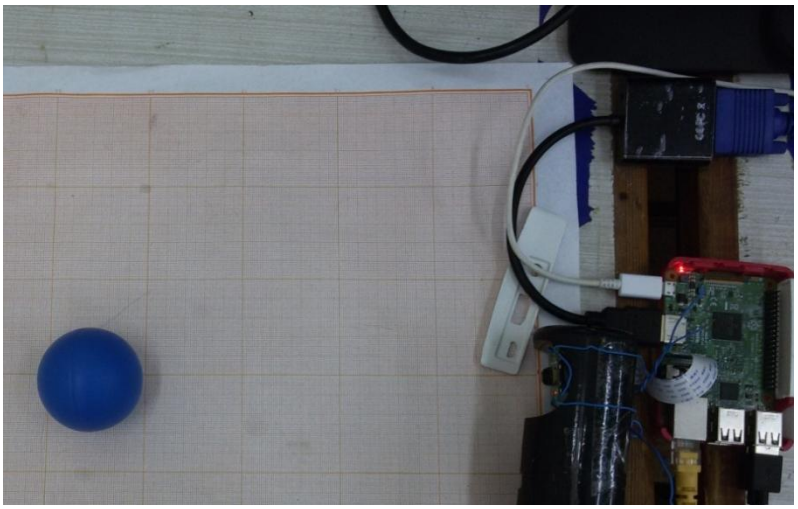


Figure 5-4: distance de la balle en réelle (z=20)

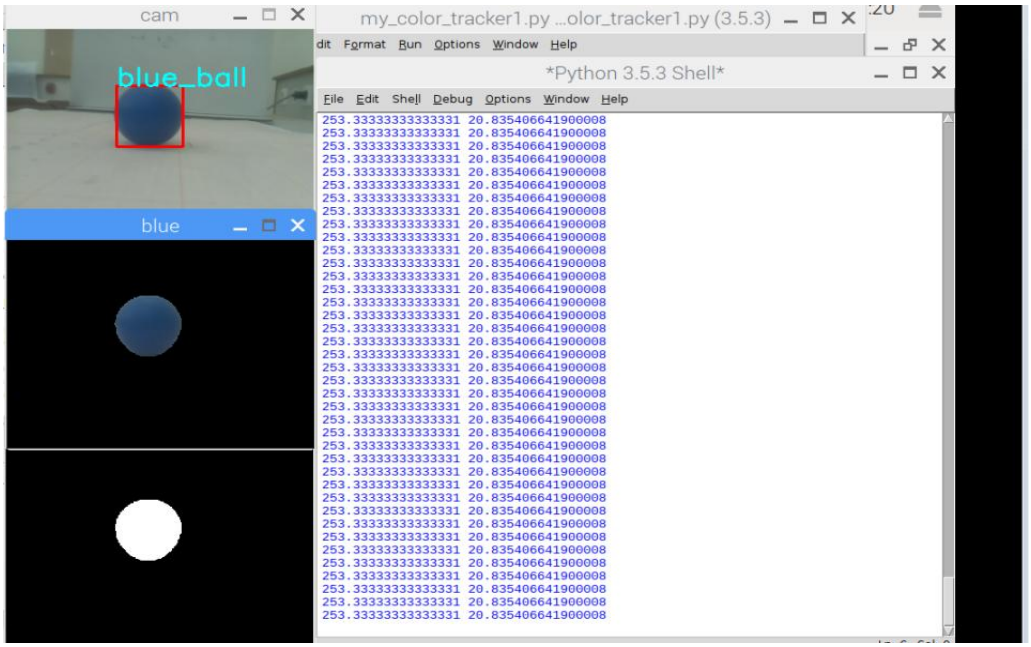


Figure 5-5: Distance de la balle calculée par la caméra X=253, Z=20.83

La figure en dessous présente le graphe de la distance de réelle en fonction de distance mesurée, c'est une droite cela veut dire que la valeur d'erreur st miniature

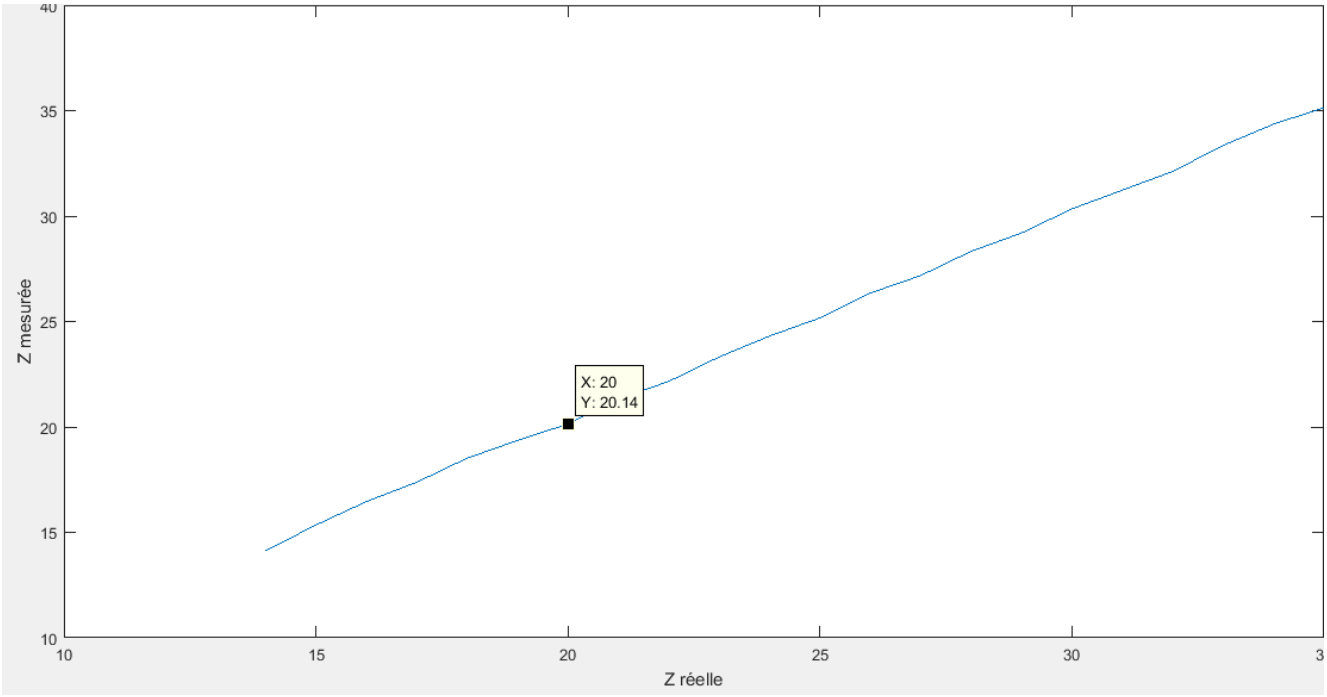


Figure 5-6 : De la distance de réelle en fonction de distance mesurée

D’après le tableau suivant, nous pouvons voir que la distance calculer par la caméra et presque exacte du celle de réelle.

Z réel le	14	15	16	17	18	19	20	21	22	23	24	25	26
Z mesurée	14.13	15.35	16.46	17.38	18.52	19.36	20.14	21.32	22.16	23.32	24.31	25.17	26.35

Tableau 5-1 : Les valeurs de la distance de balle réelle et mesurée

Alors pour le calcule de distance nous avons remarquée que si la balle est au centre de la caméra nous obtenons les bonnes valeurs voulus, mais si elle très a gauche ou a droite nous obtenons des valeurs fausse d’a peu prêt de 2 cm d’erreur et cela est a cause de coordonnée x et y de caméra, si on fixe la balle sur l’axe x et nous l’orientons sur l’axe z qui est la distance on remarque que la caméra pour elle la valeur x change alors qu’il est fixe

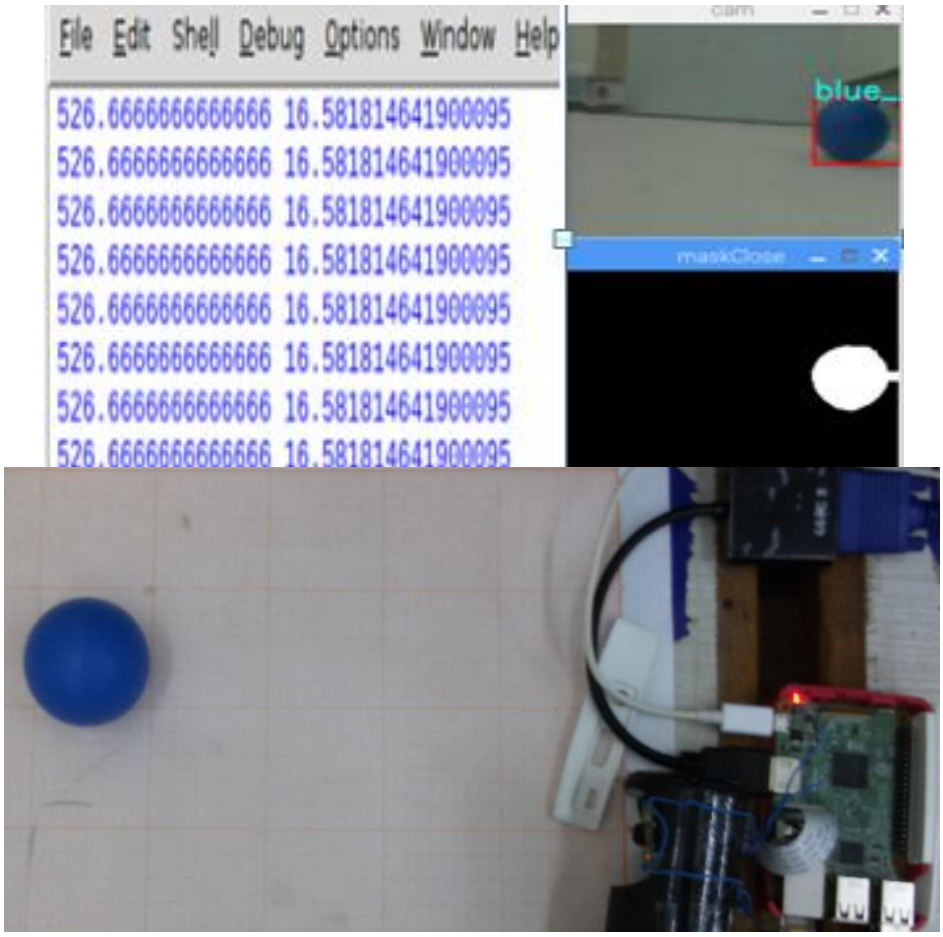


Figure 5-7 : Le Z=20 en réelle est fixe et varie sur l’axe X

Pour la caméra la distance de la balle est égale à 16 alors que c’est faux.

La figure en dessous montre que lorsque la balle se situe e centre de caméra elle donne la distance exacte

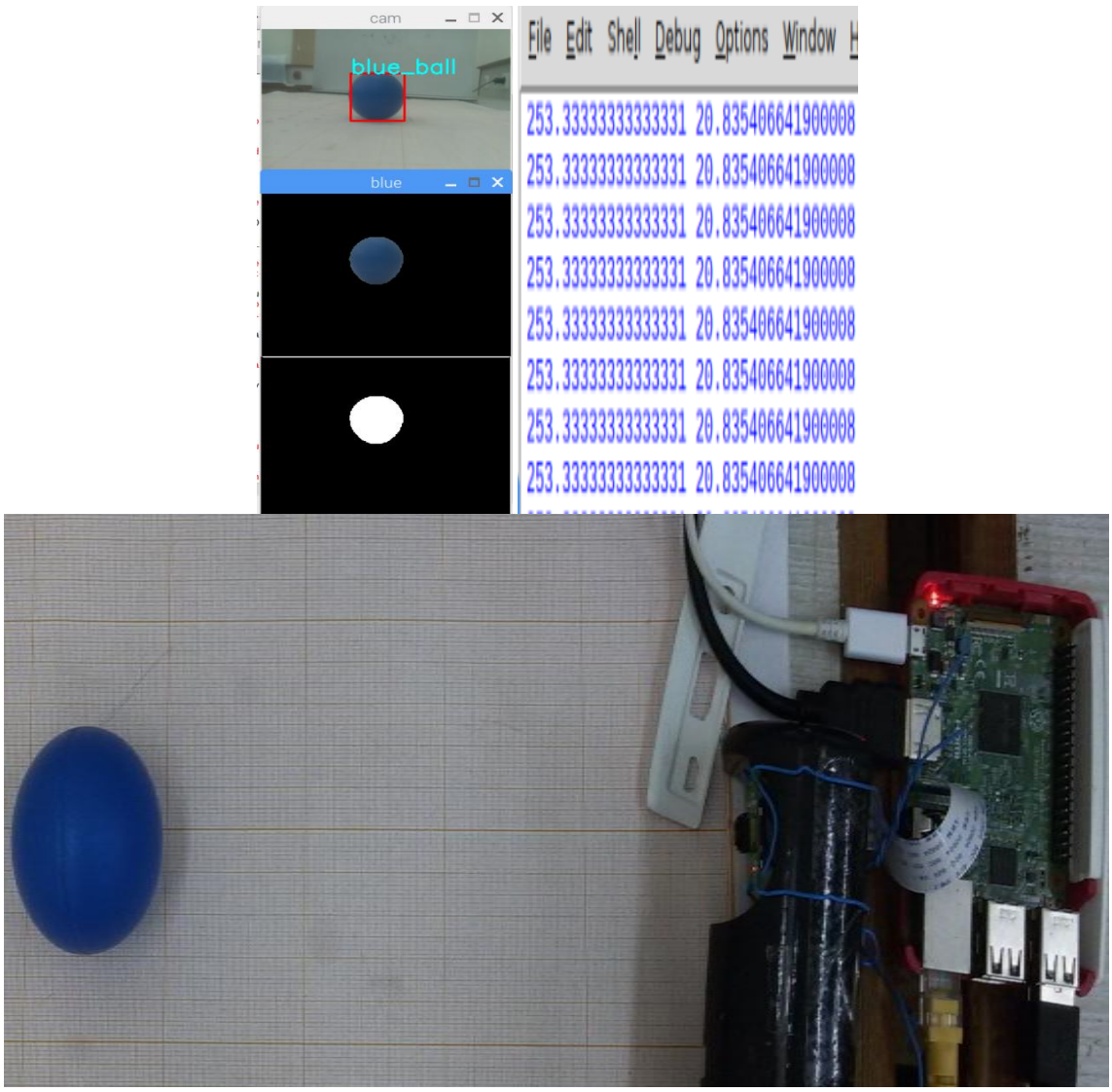


Figure 5-8 : Le Z=20 en réelle est fixe et varie sur l'axe X

D'après la figure en-dessous nous montre le réel problème pour que le X varie, plus la balle s'éloigne de la caméra plus la distance augmente plus nous reperçons que la balle deviens plus petite, donc pour la caméra la balle bouge non que sur l'axe Z mais aussi sur l'axe X mais en réalité la balle est fixe sur l'axe X

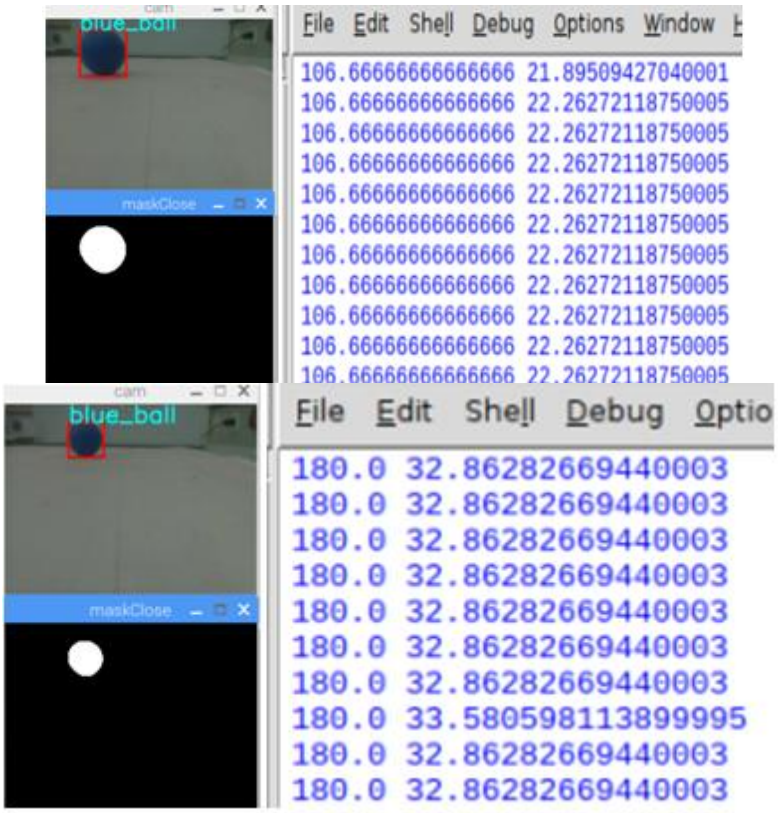


Figure 5-9 : La balle est fixe sur l'axe X et varie sur l'axe Z

Nous avons fait plusieurs testes quand la balle est fixe sur l'axe x et varie sur l'axe z et vice-versa après nous les avons traité par Matlab

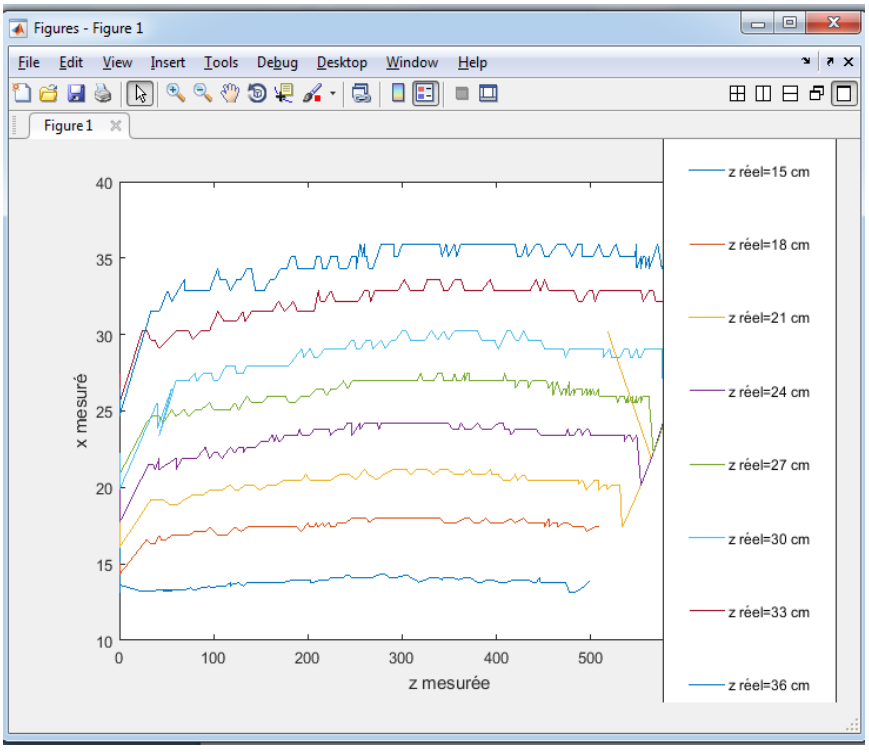


Figure 5-10 : La balle est fixe sur l'axe Z et varie sur l'axe X

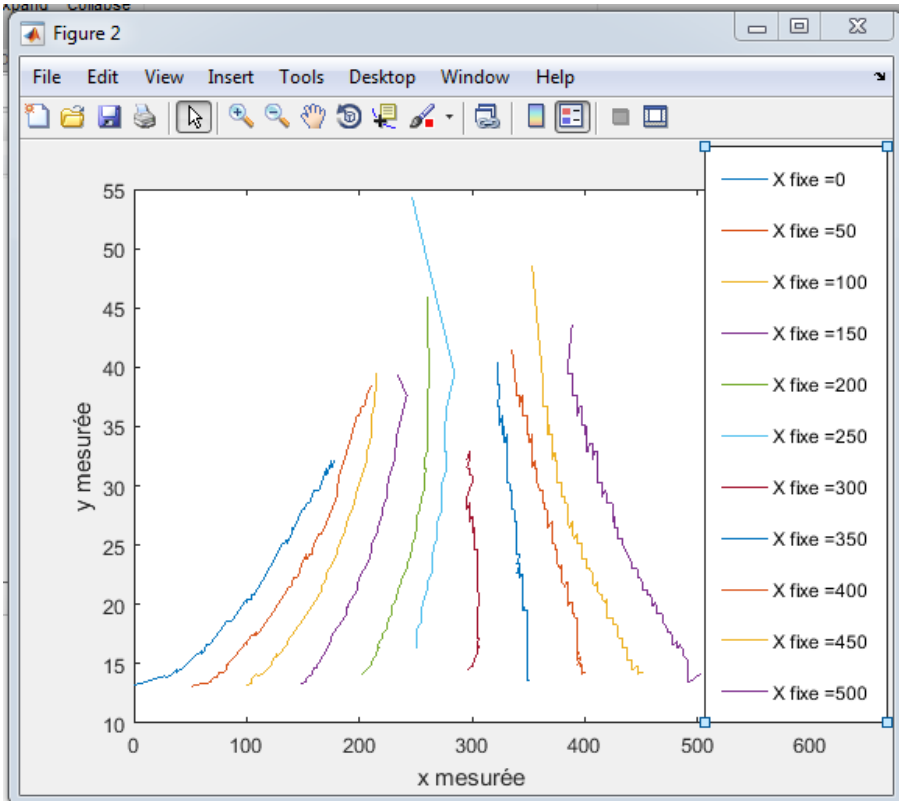


Figure 5-11 : La balle est fixe sur l'axe X et varie sur l'axe Z

Comme nous l'avons cité précédemment que juste la balle approche au centre de caméra la distance réelle est proche au distance mesurée par la caméra et la valeur de X devient stable comme celle de réel et c'est tout à fait normal par exemple la route est droite mais sur l'image nous voyions qu'elle varie vers le centre, c'est la même chose

5.3 Simulation de bras manipulateur Puma quatre ddl

5.3.1 Cinématique directe

La cinématique directe est le contrôle de notre bras robotique en fonction des positions que nous appliquons à nos servos moteurs. Nous contrôlons la totalité des servos moteurs qui correspondent aux articulations de notre bras, soit l'angle entre 2 os. C'est un contrôle manuel.

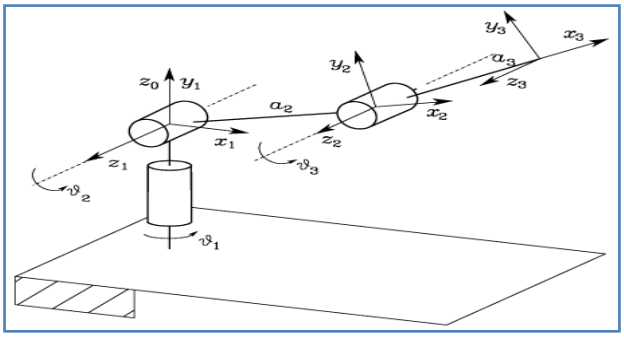


Figure 5-12: Les variables articulaires d'un bras Puma quatre ddl

Le tableau ci-dessous représente les transformations homogènes :

Articulations	Θ	d	a	α
Link1	Θ_1=0	d_1=14	0	Pi/2=90°
Link2	Θ_2=0	0	a_2=11	0
Link3	Θ_3=0	0	a_3=15	0
Link4	Θ_4=0	0	a_4=11	0

Tableau 5-2: Les quatre paramètres DH du robot Puma quatre ddl

Ces valeurs numériques de ce tableau servent à comparer par rapport aux résultats sur Matlab :

$$T_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 14 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; T_1^2 = \begin{bmatrix} 1 & 0 & 0 & 11 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; T_2^3 = \begin{bmatrix} 1 & 0 & 0 & 15 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; T_3^4 = \begin{bmatrix} 1 & 0 & 0 & 11 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Le produit des quatre matrices données :

$$T_4^0 = T_0^1 * T_1^2 * T_2^3 * T_3^4 ; \quad T_4^0 = \begin{bmatrix} 1 & 0 & 0 & x = 37 \\ 0 & 0 & -1 & y = 0 \\ 0 & 1 & 0 & z = 14 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donc on compare avec les résultats d'exécution du programme Matlab on a :

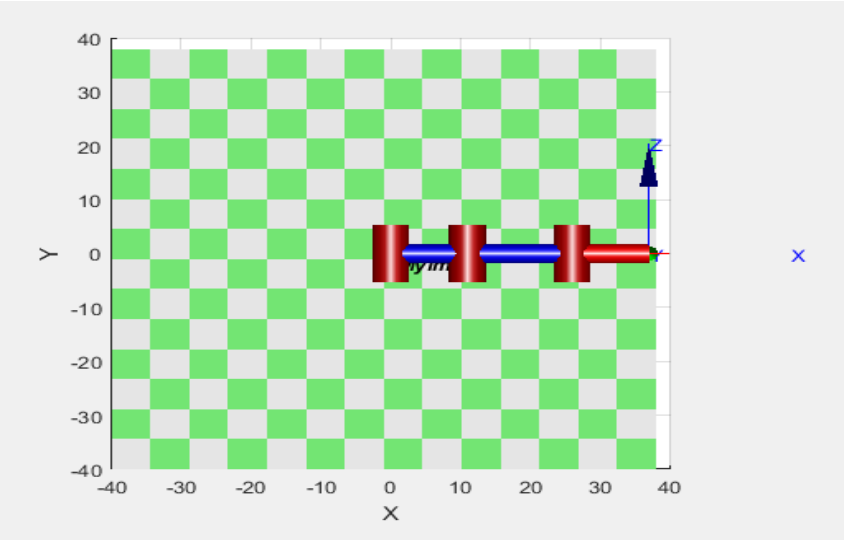


Figure 5-13: X=37

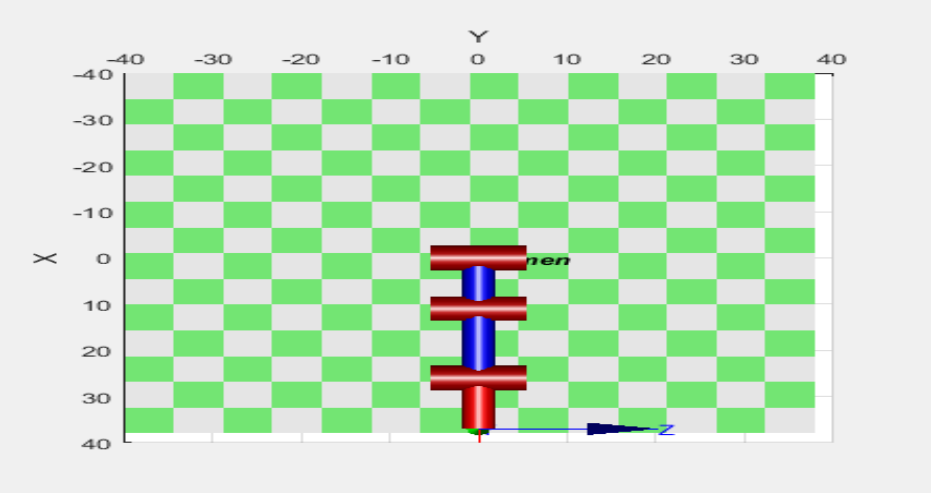


Figure 5-14: Y=0

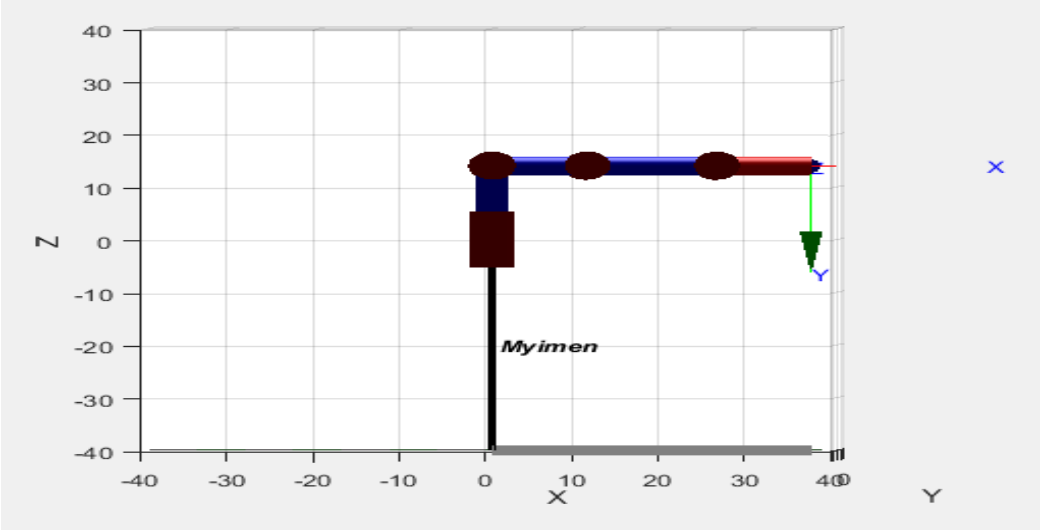


Figure 5-15: Z=14

5.3.2 Cinématique inverse

La cinématique inverse est le contrôle de notre bras robotique en fonction de la position d'arrivée. Cette position est donnée selon les axes x, y et z que nous avons définis et grâce à des algorithmes, les différents servos moteurs prennent des valeurs optimums afin d'atteindre la position souhaitée. Il est possible de poser une contrainte sur une partie du bras, généralement le poignet ; on peut souhaiter que celui-ci reste parallèle au sol ou alors qu'il saisisse l'objet par-dessus. La cinématique inverse nous permet de réaliser toutes ces contraintes.

5.3.2.1 Simulation Matlab

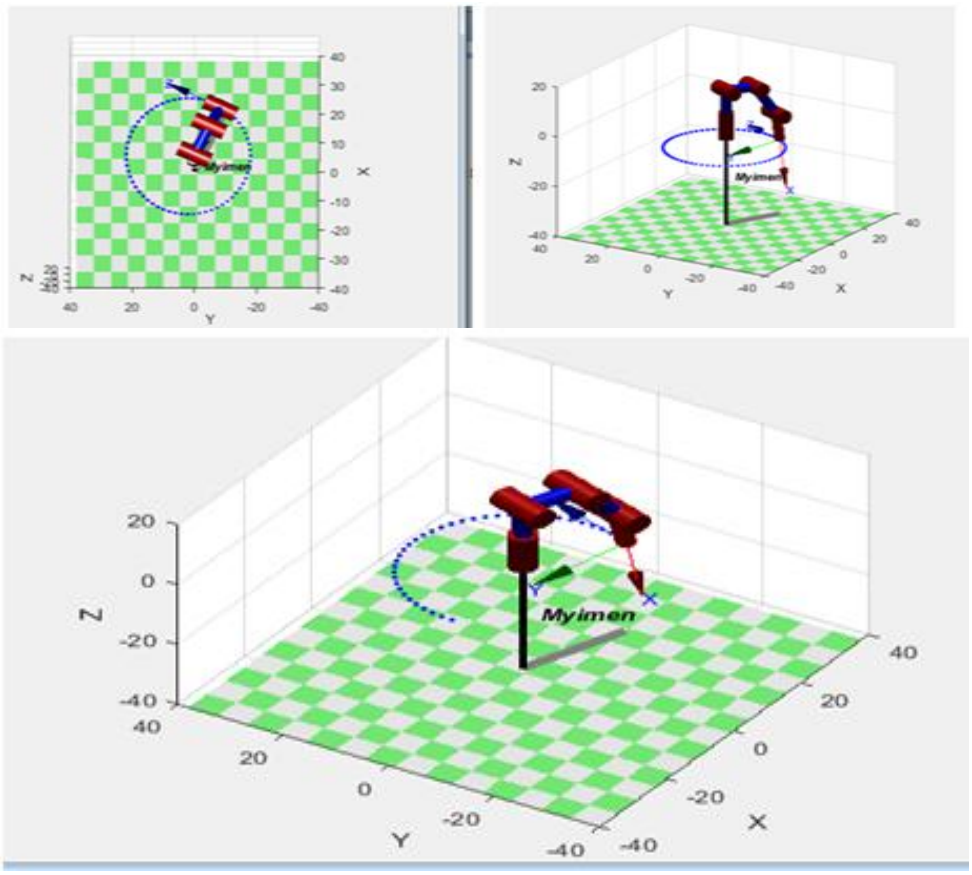


Figure 5-16: Simulation d'un robot Puma quatre ddl dessinant un cercle autour de lui

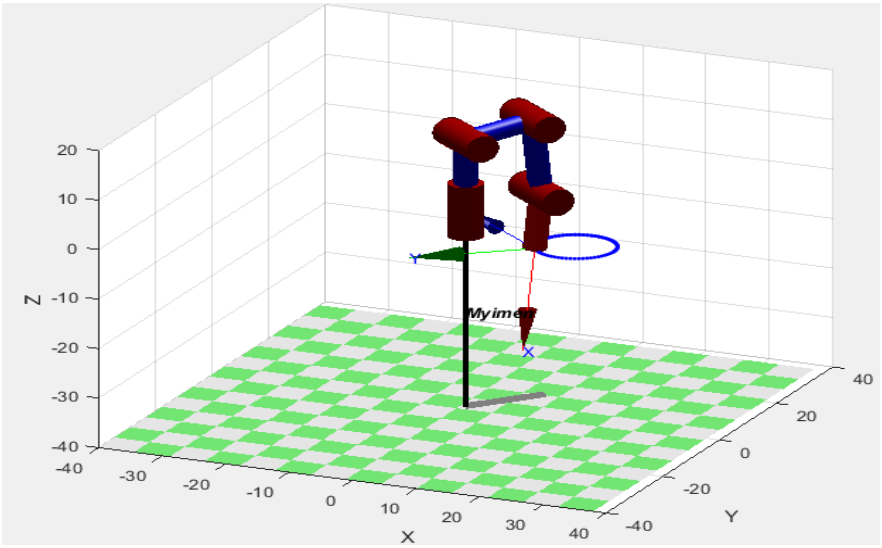


Figure 5-17: Simulation d'un robot Puma quatre ddl dessinant un cercle

5.4 Les résultats de la communication entre les deux cartes via i2c

Pour la communication des deux cartes nous avons utilisé le protocole i2c.

Figure 5-18 : Résultat de transmission des données x, y, z de la balle bleu via i2c

Lorsque la caméra détecte la balle bleu, elle transmit ces coordonnées x, y, z en boucle de Raspberry pi vers Arduino via un bus i2c, mais nous avons eux des soucis comme elle montre la figure au-dessus les coordonnées ne sont pas bien ordonnés à cause de temps le (Delay) entre les deux carte et cela est un danger pour le bras qui va commencer à faire n'importe quoi puisque il reçoit des données fausse, alors on opter pour une autre méthode qui est le port série , il est plus fiable que le i2c.

5.5 Les résultats réels de transformation homogène des repères

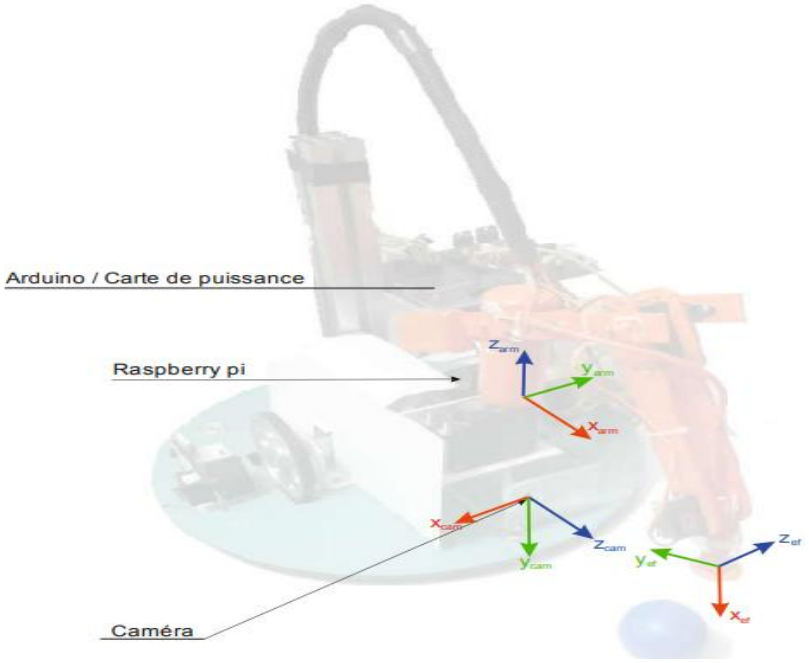


Figure 5-19 : Positions des repères sur le robot

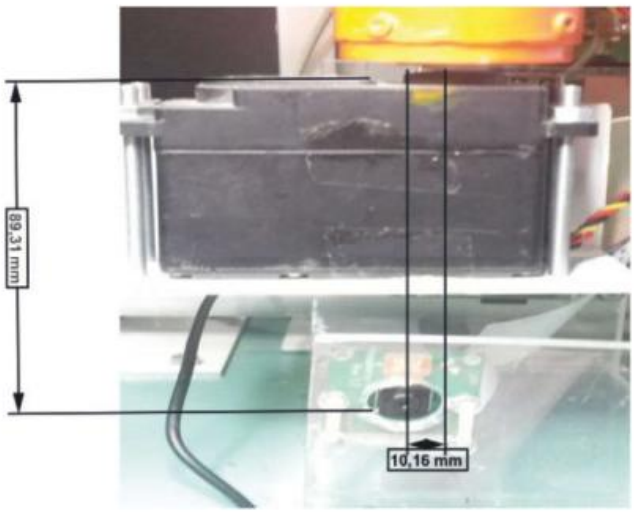


Figure 5-20 : Vue de face montrant les distances verticale et horizontale entre la caméra et la base du bras

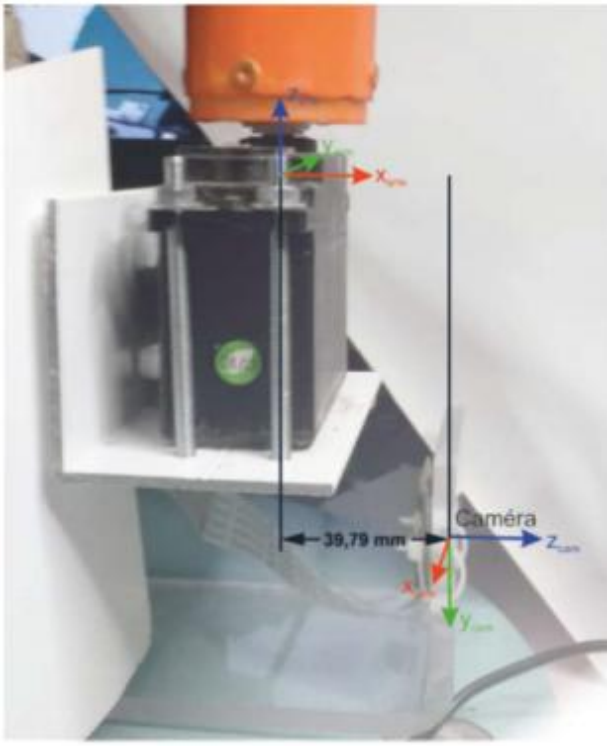


Figure 5-21 : Vue de côté montrant distance horizontale entre la caméra et la base du bras

Les figures ci-dessus montrent les positions relatives des repères liés à la caméra et celui lié à la base du bras manipulateur. En exploitant l'étude détaillée du paragraphe (3-4) et remplaçant les distances utilisées en simulation par les valeurs réelles du bras nous obtenons la matrice de Transformation Homogène suivante :

$$iHac = \begin{bmatrix} 0 & 0 & 1 & 40 \\ -1 & 0 & 0 & -10 \\ 0 & -1 & 0 & -90 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Cette matrice nous permettra donc de calculer les coordonnées de l'objet détecté par la caméra dans le repère du bras simplement réalisant une multiplication de cette matrice avec le vecteur des coordonnées générées par la caméra.

5.6 Conclusion

Ce chapitre a été consacré aux différentes étapes par lesquelles nous sommes passés pour la concrétisation des résultats escomptés. Nous avons montré la manière avec laquelle nous avons pu ajouter une troisième dimension à notre caméra en se basant sur la taille de l'objet détecté afin d'évaluer sa position en profondeur par rapport à la caméra. Ceci était indispensable car notre bras manipulateur évolue dans un espace en 3 dimensions. Nous avons aussi montré les difficultés inhérentes à la vision 3D en comparant les distances observées par la caméra avec celles mesurées réellement, le constat était que la caméra introduit des déformations au niveau de l'axe latéral qui a pour effet de générer une erreur sur cette dimension. Pour ce qui concerne notre cas cela n'a pas un grand impact sur le résultat final vue que l'espace de travail de notre bras est relativement limité.

Conclusion générale

Conclusion générale

Pour conclure ce mémoire nous pouvons considérer que l'objectif principal du sujet proposé a été atteint d'une manière très satisfaisante. En affrontant toutes les difficultés rencontrées tout au long de la réalisation de ce projet nous avons appris que, lorsqu'il s'agit de travailler sur un projet de robot avec des capacités de vision, chaque détail pose problème pour lequel il faut trouver une solution. Pour ce qui nous concerne la première étape était de se familiariser avec la partie vision et son utilisation avec le langage de programmation Python sur la carte Raspberry pi, cela nous a pris une partie assez importante du temps consacré à ce projet.

Tel que précisé au niveau des différents chapitres de ce mémoire il y avait des parties où nous avons fait appel à des simulations sur Matlab pour tester et vérifier les solutions proposées, avant d'être implémentées sur les modules Arduino et Raspberry pi. La communication entre ces deux cartes a finalement été assurée à travers le port de communication série disponible sur ces deux cartes. Cette manière de procéder nous a permis de réaliser une plateforme sur laquelle nous pouvons envisager différents types d'expériences qui permettraient, par exemple de l'installer sur une chaine de production dans laquelle notre bras, une fois fiabilisé, pourrait ramasser des objets de couleur préprogrammée, pour les mettre dans un conteneur à sa proximité. Une autre utilisation potentielle de notre plateforme serait de l'exploiter dans un milieu pédagogique pour des travaux pratiques entrant dans l'enseignement de la robotique.

Tel que c'est le cas pour tout travail de recherche ce projet se prête à de multiples améliorations et notamment en ce qui concerne l'utilisation de servomoteurs de meilleures qualités. Etant donné que ce bras est fixé sur une plateforme mobile, l'étape naturelle qui devra suivre sera celle d'ajouter une mobilité à notre bras pour lui permettre d'atteindre des objets distants. Le reste du travail sera essentiellement axé sur la programmation de la carte Arduino qui est en charge de gérer tous les actionneurs installés sur le robot, la partie matérielle est tout à fait prête pour une telle extension.

Références bibliographique

Références bibliographique

[1] **KHACHOUCHE Samir,** ''Conception et réalisation d'un bras manipulateur de type PUMA à 4 degrés de liberté '', mémoire de Master en Automatique, option Automatique et Systèmes, Département d'Electronique, Université de Blida, Session 2018/2019.

[2] **OUCHEFOUN Nassima, SID-AHMED Yasmine**, ''Réalisation d'un robot mobile avec vision intelligente embarquée'' mémoire de Master en Automatique, option Automatique, Département d'Electronique, Université de Blida, Session 2017/2018.

[3] **O. Patrouix,** '' Automatisation Robotique Modélisations de mécanismes Génération de trajectoires '', version 1.2 / 2012.

[4] notes de cours commandes de Robots de Manipulations /master2/automatique et systèmes/année 2017/2018, université de Blida, Mr Kazed Boualem.

Les sites Web :

[5]http://www.map.toulouse.archi.fr/works/panoformation/imagenum/imagenum.htm

[6] https://fr.wikipedia.org/wiki/Traitement_d%27images

[7]https://patrick-bonnin.developpez.com/cours/vision/Bases-du-Traitement-Image/Chap1/#signetNoteBasPage2

[8] https://webdevdesigner.com/q/why-do-we-convert-from-rgb-to-hsv-386476/

[9] http://operationpixel.free.fr/traitementbinaire_binarisation.php