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

Mention Électronique
Spécialité Électronique des Systèmes embarqués

présenté par

Chaouch Ibrahim

&

Semmar Hocine

Commande d'un bras de robot par vision artificielle

Proposé par : Kabir Yacine & Naceur Djamila

Année Universitaire 2017-2018

Remerciements

Avant tout, nous remercions dieu tout puissant pour nous avoir donné la force et le courage d'accomplir ce travail avec abnégation.

Nos vifs et sincères remerciements s'adressent spécialement à, Monsieur KABIR YACINE et Madame NACEUR DJAMILA dont on a eu la chance de les avoir comme encadreurs et qui ont bien voulu nous confier ce travail riche d'expériences et nous guider dans chaque étape de sa consécration. Vous nous avez toujours réservé un chaleureux accueil, malgré vos obligations et les contraintes professionnelles. Vos talents ainsi que vos compétences et votre sens du devoir nous ont marqué à jamais. Vos encouragements inlassables, votre amabilité, votre gentillesse et votre patience méritent toute notre attention. Veuillez trouver ici l'expression de notre estimes et notre considération.

Nous remercions également tous nos professeurs du département électronique

nous remercions également, les membres du jury de nous avoir fait l'honneur de juger ce mémoire. Veuillez accepter l'expression de nos vives gratitudes

Enfin, A nos parents et à toutes les personnes qui ont contribué de prés ou de loin, d'une manière directe ou indirecte à l'élaboration de ce travail de fin d'études.

ملخص:

نظرا للافتقار إلى المرونة في الأنظمة الآلية بسبب نقص معلومات الدخول المتعلقة بالروبوتات فإن الرؤية الاصطناعية هي البديل الأفضل لحل هذه المشكلة. يتكون هذا المشروع من تصميم و تصنيع نظام تحكم عبر الرؤية قائم على رازبيري لمعالجة الصور بمساعدة المكتبة أوبنسيڤي بلغة الرمجة سي بلاس بلاس و بطاقة أردوينو لتكييف الإشارات. المحرك هو ذراع بثلاث درجات من الحركة. يجب أن يكون النظام مستقلا.

كلمات المفاتيح: راسبيري، أوبنسيڤي ، أردوينو

Résumé :

Vue le manque de la flexibilité des systèmes robotisés par manque d'informations en entrée des robots, la vision artificielle est la meilleure alternative pour résoudre ce problème. Ce projet consiste à faire la conception et la réalisation d'un système de commande par vision , basé sur un Raspberry pour le traitement d'image ,sur la librairie Opencv en c++ , et sur une carte Arduino pour le conditionnement du signal , l'actionneur est un bras à trois degrés de liberté , le système doit être autonome .

Mots clés : Raspberry ; OpenCV ; Arduino.

Abstract :

Regarding the lack of flexibility of robotic systems due to lack of entering robotic informations, artificial vision is the best alternative to solve this problem. This project consists in designing and implementing a Raspberry-based vision control system for image processing using the c ++ OpenCV library and an Arduino card for signal conditioning. The actuator is an arm with three degrees of freedom. The system must be autonomous.

Keywords : Raspberry ; OpenCV; Arduino.

Liste des acronymes et abréviations

IDE : integrated development environment .

Os : operating system

IOS : iphone operating system

CSI : Camera Serial Interface

DSI : Display Parallel Interface

Soc : system on chip

HDMI : High Definition Multimedia Interface

ARM : Reduced Instruction Set Compute

BCM : brodcom

GPIO : General Purpose for Input and Output

SDRAM : Synchronous Dynamic Random Acces Memory

Opencv : Open Source Computer Vision Library

LSB : least significant bit

MSB : most significant bit

QT : environnement de développement

[AC] : la distance entre la première articulation est la deuxième

[AB] : la distance entre la deuxième et la troisième articulation

[WE] : distance entre la webcam et le bras

[WJ] : la distance entre l distance entre la webcam et objet

[JK] : l'adjacent du triangle JEK

[WK] : l'adjacent du triangle WJK

[EK] : l'opposé du triangle JEK

δ : l'angle du moteur lié a la webcam

ζ : l'angle du moteur du bras

S3003 : référence du servo moteur

[X1] : la distance d'une articulation

SDHC : Secure Digital High Capacity

SDXC : Secure Digital eXtended Capacity

LPDDR : Low Power Double Data Rate

Cd : change directory

AMD : Advanced Micro Devices

IoT : Internet of things

EEPROM : Electrically-Erasable Programmable Read-Only Memory

ISP : Interface Serial Port

CAN : Convertisseur Analogique Numérique

AVR : famille de microcontrôleur développer par ATmel

PWM : Pulse-width modulation

SPARK : Scalable Processor Architecture, is a reduced instruction set computing

MIPS : Million instructions per second

Table des matières

Introduction générale 1

Chapitre 01 : Généralités sur la commande par vision 2

2.1 Introduction 2

2.2 Problématique 2

2.3 Solution apportée 2

2.4 Les robots mobiles 3

2.5 Les robots industriels 4

2.5.1 Définition de la vision industrielle 4

2.5.2 Technique d'asservissement visuelle 5

2.6 Conclusion 6

Chapitre 2 : Environnement de travail 7

2.1 Introduction 7

2.2 Hardware utilisé 8

2.2.1 Raspberry 8

2.2.2 L'Arduino 11

2.3 Software sur la carte Raspberry 15

2.3.1 Le noyau Linux 15

2.3.2 Implémentation sur la carte 15

2.3.3 Opencv 17

2.3.4 Compilation par ligne de commande 20

2.3.5 Utilisation d'un IDE 20

2.3.6 WirigPi 24

2.4 Conclusion 25

Chapitre 3 : conception et réalisation 26

3.1 Introduction 26

3.2 Conception du bras 26

3.2.1 Modèle géométrique du bras 27

3.2.2 Montage du bras par rapport à la webcam 29

3.2.3 Calcul de distance et géométrie 30

3.2.4 Calcul de la distance (Camera-Objet) 30

3.3 Montage général 32
3.3.1 Schéma synoptique 32
3.3.2 schéma électronique (carte de puissance) 32
3.4 Algorithme et traitement d'images 34
3.4.1 Traitement d'images et seuillage (binarisation) 34
3.4.2 Binarisation par la méthode HSV 35
3.5 Exploitation de l'image binaire pour la commande 38
3.5.1 Test de présence d'un objet 38
3.5.2 Calcul de la largeur 39
3.5.3 Centrer l'objet dans l'image 40
3.5.4 Calcul de la distance focale 41
3.5.5 Conversion et transmission de la distance 42
3.5.6 Reconnaissance des formes 43
3.6 Algorithme sur Arduino 45
3.6.1 Initialisation 46
3.6.2 Présence d'objet 48
3.6.3 Centrer l'objet dans l'image 49
3.6.4 Lecture et conversion 50
3.6.5 Application des formules géométriques 51
3.6.6 Conversion de la distance en angles 52
3.7 Photo de réalisation 52
3.7.1 Conception de la pincette 52
3.7.2 Réalisation du bras 53
3.8 Taux de réussite 56
3.9 Conclusion 57
Conclusion générale 58
Annexes 59
Bibliographie 62

Liste des figures

Figure 1: chaine du système de commande par vision 1
Figure 1-1 : système de commande par vision détaillé 3
Figure 1-2 : Exemples de deux robots a gauche se déplace avec des chenilles et a droite avec des roues. 4
Figure 1-3 : A droite la configuration œil dans la main, a gauche l'œil regarde la scène 6
Figure 2-1 : diagramme du système global 7
Figure 2-2 : les E/S du Raspberry pi 2 8
Figure 2-3 : GPIO (general purpose for input and output) du Raspberry pi2 10
Figure 2-4 : les éléments principaux de la carte Arduino mega 11
Figure 2-5 : IDE (integrated devolopment environement) d'Arduino 12
Figure 2-6 : environnement de programmation 13
Figure 2-7 : Bouton de compilation du programme. 13
Figure 2-8 : Gestionnaire des périphériques sous Windows 14
Figure 2-9 : Bouton de téléverssement 14
Figure 2-10 : accueil du système d'exploitation ubuntu. 16
Figure 2-11 : terminal du système. 18
Figure 2-12 : interface de du logicielle QT. 21
Figure 2-13 : choix du kits. 22
Figure 2-14 : ajout du path du compilateur. 22
Figure 2-15 : ajout du chemin des libraires. 23
Figure 3-1 : schéma explicatif du positionnement d'objet. 27
Figure 3-2 : dessin explicatif de la géométrie du bras. 28
Figure 3-3 : dessin explicatif de la géométrie du bras. 28
Figure 3-4 : vue de haut du système. 29
Figure 3-5 : géométrie et calcul de distance. 30
Figure 3-6 : géométrie et calcul de distance. 31
Figure 3-7 : schéma synoptique du système. 32
Figure 3-8 : schéma électronique (carte de puissance). 32
Figure 3-9 : a gauche le négatif a droite le circuit développé avec TCI. 33
Figure 3-10 : Résultat de la gravure du PCB. 33
Figure 3-11 : a gauche le plan HSV à droite le plan HSL. 34
Figure 3-12 : Histogramme d'une image a niveau de gris. 35
Figure 3-13 : l'interface graphique (rectangle rouge : introduction des paramètres de seuillage). 36
Figure 3-14 : Démonstration des fonctionnalités de l'interface graphique. 37
Figure 3-15 : Démonstration des fonctionnalités de l'interface graphique. 38
Figure 3-16 : Organigramme (fonction de détection). 39
Figure 3-17 : Organigramme de calcul de la largeur en pixel. 40
Figure 3-18 : image binaire. 41
Figure 3-19 : organigramme d'asservissement du moteur de la webcam. 41
Figure 3-20 : Diagramme explicatif de la distance focale. 42

Figure 3-21 : Organigramme de conversion décimale-binaire. 43
Figure 3-22 : variation de la largeur d'objet selon la forme. 44
Figure 3-23 : organigramme de reconnaissance des formes 45
Figure 3-24 : organigramme qui se déroule dans l'Arduino 46
Figure 3-25 : Modèle géométrique du réducteur 47
Figure 3-26 : Organigramme de l'initialisation 48
Figure 3-27 : organigramme de recherche 49
Figure 3-28 : Organigramme de mise en face de la camera 50
Figure 3-29 : organigramme de lecture et conversion de la distance 51
Figure 3-30 : Organigramme de calcul des données pour le bras 52
Figure 3-31: Mécanisme de la pincette 53
Figure 3-32 : axe de rotation du bras 53
Figure 3-33 : en bas dessous du bras (moteur pas a pas) en haut le bras avec les trois servo-moteurs 54
Figure 3-34 : vue de profil prototype 55
Figure 3-35 : vue de haut du prototype 55
Figure 3-36 : vue de bas du prototype 56
Figure 3-37 : performance du système (Taux de réussite) 57
Figure 2 : géométrie de la stéréoscopie 60

Liste des tableaux

Tableau 2-1 : liste des Arduino avec leurs microcontrôleurs et nombres E/S.....11
Tableau 2-2 : spécifications technique de la carte Arduino mega.....12

Introduction générale

La robotique permet d'aider l'homme dans les tâches difficiles, répétitives ou pénibles. De plus, elle constitue le rêve de substituer la machine à l'homme dans ces tâches. Les facultés de perception et de raisonnement des robots progressent chaque jour actuellement et plus encore dans l'avenir. Ils sont appelés à jouer un rôle de plus en plus important dans notre vie [1]. Parmi les facteurs qui influencent l'intelligence d'un robot on trouve le taux d'information en entrée à traiter. Ces informations sont souvent récoltées par des capteurs (magnétique, capacitif, ultrasons, infrarouges…etc.). Plus ces informations sont nombreuses, plus le robot aura une meilleure flexibilité. Mais il y a un domaine plus récent, qui utilise le traitement d'image comme moyen d'information, appelé vision par ordinateur.

La vision par ordinateur, aussi appelée vision artificielle, est une branche de l'intelligence artificielle dont le but est de permettre à une machine d'analyser, traiter et de comprendre une ou plusieurs images prises par un système d'acquisition [2] tel qu'une caméra.

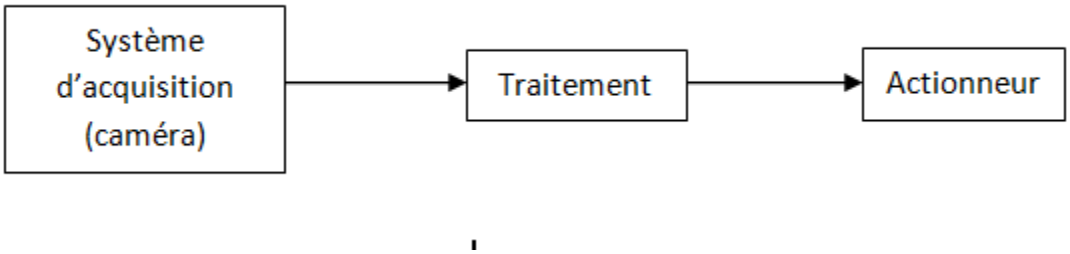

Figure 1 :commande par vision

Ce projet consiste à faire la conception et la réalisation d'un prototype basé sur ce principe. Toutefois, il faut noter que la contrainte majeure du système est le temps. Pour qu'un système de vision échappe à la contrainte temps réelle ,il est impératif d'utiliser un hardware plus au moins puissant et d'optimiser les algorithmes de traitement le mieux possible tout en gardant les performances du robot car une scène vidéo est un flux de données énorme qui doit être traiter par le processeur dans un temps bien déterminé sans qu'il y ait un disfonctionnement du système. le manuscrit est divisé en trois chapitres , le premier est une généralité sur la commande ,le deuxième regroupe l'environnement de travail et le troisième porte sur la conception et la réalisation .

Chapitre 01 : Généralités sur la commande par vision

1.1 Introduction

La robotique est un ensemble de disciplines (électronique, automatique, informatique, mécanique), qui se scindent en deux types : les robots industriels et les robots mobiles.

Les robots industriels sont généralement fixes et sont utilisés dans de nombreuses applications industrielles : l'assemblage mécanique, la soudure, la peinture, etc.

Les robots mobiles ne sont pas fixes, ils sont classifiés selon la locomotion ; en robots marcheurs, à roues, à chenilles...etc., et selon le domaine d'application, en robots militaires, de laboratoire, industriels et de services. [1]

L'imagerie se divise en deux grandes parties : le traitement d'image haut niveau qui consiste à améliorer, modifier ou changer l'image et le traitement bas niveau qui lui consiste à extraire l'information de l'image, dans ce cas, on parle de vision artificielle.

Après l'extraction de l'information, il est temps de l'utiliser pour commander le robot selon les taches préprogrammées.

1.2 Problématique

Malgré l'évolution des systèmes automatisés, la machine reste toujours incapable de prendre des décisions d'elle-même et par conséquent plusieurs problèmes se posent :

- Les accidents dans l'interaction homme-machine.
- Le manque de flexibilité des robots ordinaires.
- La complexité des algorithmes par raison de manque d'information en entrée
- Taches limitées

1.3 Solution apportée

La solution pour ces problèmes est de concevoir un système de commande par vision artificielle donc on n'a pas besoin de programmer énormément de tâches ni de prédire des

actions. Grâce à ce couplage, les robots sont plus intelligents car ils peuvent désormais effectuer des tâches précises et à grande vitesse sur un objet. C'est le robot qui va s'adapter à son milieu, et non pas l'inverse.

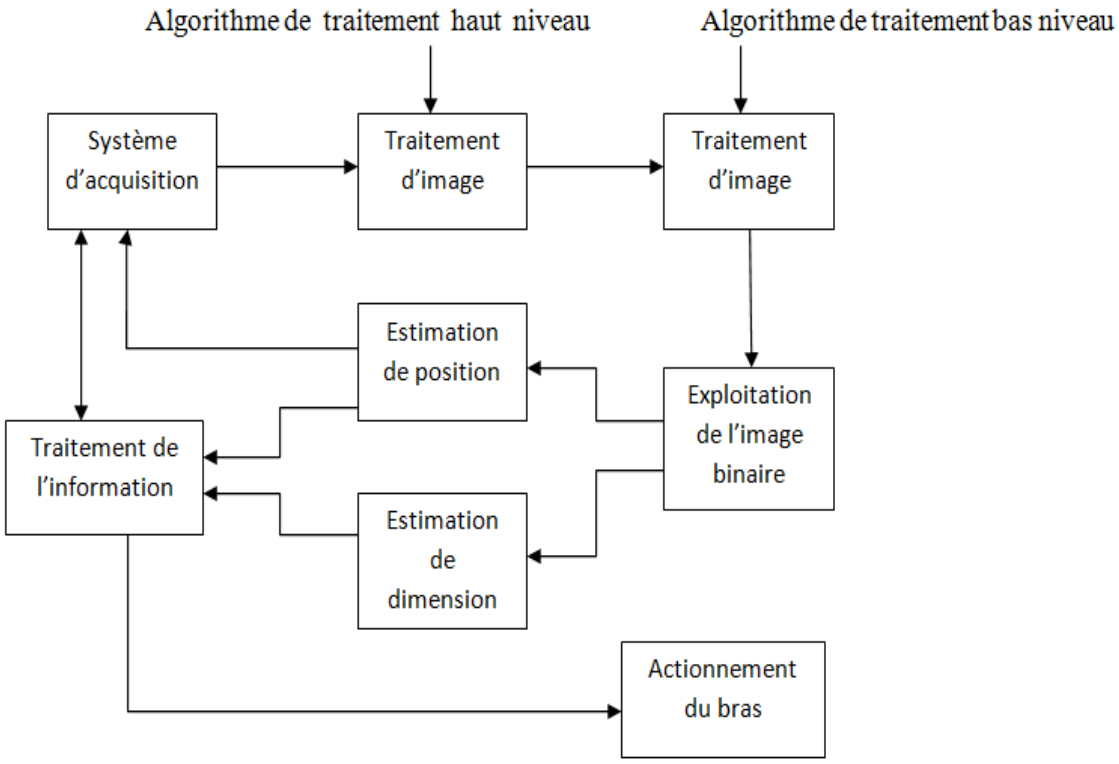

Figure 1-1 : système de commande par vision détaillé.

1.4 Les robots mobiles

Le robot mobile est doté de moyens qui lui permettent de se déplacer dans son espace de travail. Suivant son degré d'autonomie ou degré d'intelligence, il peut être doté de moyens de perception et de raisonnement. Certains sont capables, sous contrôle humain réduit, de modéliser leur espace de travail et de planifier un chemin dans un environnement qu'ils ne connaissent pas d'avance.

Actuellement, les robots mobiles les plus sophistiqués sont essentiellement orientés vers des applications dans des environnements variables ou incertains, souvent peuplés d'obstacles nécessitant une adaptabilité à la tâche. [1]

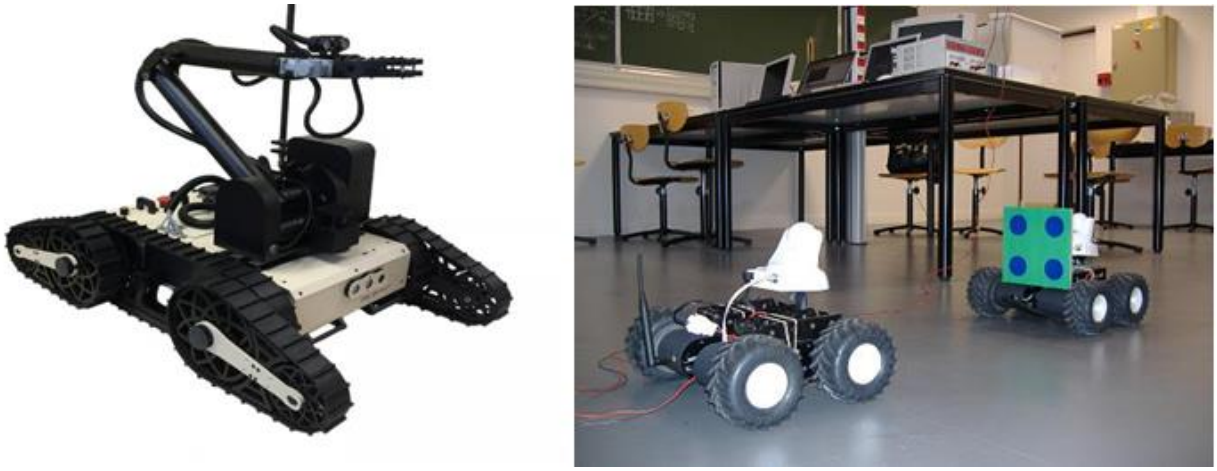

Figure 1-2 : Exemples de deux robots a gauche se déplace avec des chenilles et a droite avec des roues. [web 1].

1.5 Les robots industriels

Contrairement au robot mobile, les robots industriels sont généralement fixes.

Tout comme les unités de fabrications modernes, ce sont des systèmes d'automatisme de haut niveau qui utilisent des ordinateurs comme partie intégrante dans leur chaine d'asservissement. Actuellement ils constituent la pièce maitresse de toute automatisation industrielle.

La robotique est utilisée aussi en technologie spatiale et dans le domaine médical [1].

Les robots industriels ont été créés pour accroitre la productivité et améliorer la qualité au sein des unités industrielles. Ils s'avèrent particulièrement utiles dans de nombreuses applications industrielles en particulier la peinture, la soudure, le contrôle et l'assemblage mécanique... [1]

Les recherches actuelles portent en effet sur des robots évolués capables d'entendre, de voir, de toucher et de prendre des décisions.

1.5.1 Définition de la vision industrielle.

On définit la vision industrielle comme étant l'application de la vision assistée par ordinateur aux domaines industriels de production et de recherche.

Les domaines d'applications de la vision dans le monde industriel sont très nombreux et variés. Elle peut être utilisée dans la production de masse à haute cadence, le souci constant d'amélioration de la qualité et la recherche de gain économique, ce qui pousse de plus en plus les industriels à automatiser les moyens de production. La vision industrielle est une réponse à ces préoccupations pour les opérations de contrôle de la production. En effet les machines de vision industrielle permettent un contrôle de la production à haute cadence, à la différence d'un opérateur, une machine n'est jamais fatiguée et ses critères de décisions ne varient pas.

La vision industrielle peut aussi être utilisée pour gérer des flux d'objets importants, par exemple la lecture optique d'un code à barres ou d'une adresse postale sur un colis pour l'orienter dans un centre de tri, ou encore le tri de pommes par couleurs différentes avant emballage.

Enfin la vision industrielle est un moyen de guidage pour un système mobile autonome (comme un robot) lorsque ses mouvements ne peuvent pas être déterminés par avance comme, par exemple la capture d'objets sur un tapis roulant. Une caméra est alors embarquée sur la tête du robot et permet le positionnement de celui-ci au point désiré. [3]

1.5.2 Technique d'asservissement visuelle

Un asservissement visuel consiste à commander un dispositif physique ou virtuel, le plus souvent actionné mécaniquement, à partir d'un capteur visuel quelconque, le plus répandu étant la caméra matricielle. Le dispositif d'acquisition/traitement est un système capable d'analyser l'information visuelle numérisée, autrement dit un ordinateur. Souvent c'est le même système qui est chargé du traitement de l'image et de la commande. Cette commande, qui résulte du retour visuel, est envoyée au dispositif.

Comme pour tout asservissement, les données issues du capteur sont comparées avec la consigne. Pour exprimer cette comparaison, on a l'habitude de distinguer plusieurs espaces de représentation des données. C'est cette classification que nous avons retenue pour la suite. Les asservissements visuels peuvent aussi être distingués par le fait que le capteur soit embarqué sur le dispositif (L'œil est dans la main et observe la scène), soit déporté (l'œil regarde la scène (éventuellement la main)). voir figure 1-3

Figure 1-3 : A droite la configuration œil dans la main, a gauche l'œil regarde la scène [web 2].

Le choix de la configuration dépend de la tâche à exécuter ou de l'encombrement du capteur visuel. On peut également distinguer les asservissements visuels par leurs comportements dynamiques. Ainsi, la commande d'un déplacement du dispositif peut être effectuée, par séquence, par mouvements discontinus, à l'issue de l'analyse des images (look and move). Ce type de commande ne nécessite généralement pas de modéliser le comportement dynamique des éléments entrant en jeu et a été mis en place à l'origine. La commande cinématique a permis plus tard d'élaborer une commande en vitesse pour obtenir une séquence continue des mouvements du dispositif. Plus récemment, les asservissements visuels rapides ont été mis en œuvre, grâce aux capacités grandissantes des calculateurs, des capteurs, mais aussi des technologies des actionneurs.

1.6 Conclusion

Ce chapitre introductif a été consacré essentiellement à la présentation de la robotique en générale et aux méthodes et techniques utilisées pour l'étude du projet. nous avons aussi mis a l'accent sur ses objectifs et sa planification.

Chapitre 2 : Environnement de travail

2.1 Introduction

Avec le développement des nouvelles technologies la gestion des robots devient plus ou moins facile, on trouve également plusieurs alternatives proposées comme la gestion assistée par ordinateur, Raspberry, Arduino ou d'autres cartes dédiées. Mais le choix du type de gestion diffère d'un robot à un autre, par exemple si on prend le cas d'un petit robot suiveur de ligne il est préférable de prendre comme carte de commande la plus basique comme Arduino ou tout autre carte à microcontrôleur, car c'est un gâchis de ressources d'utiliser un ordinateur. Généralement ce n'est pas pratique. Mais lorsqu'on cherche à commander un bras de peinture industrielle par exemple il est préférable de choisir une plateforme robuste pour plus de fiabilité et de précision.

Dans ce projet, nous avons une partie traitement d'images et une autre pour la commande des moteurs, pour cela nous divisons la partie hardware en trois parties :

- un Raspberry pour le traitement d'image
- un Arduino pour la gestion du bras
- une petite carte de puissance comme driver pour les moteurs

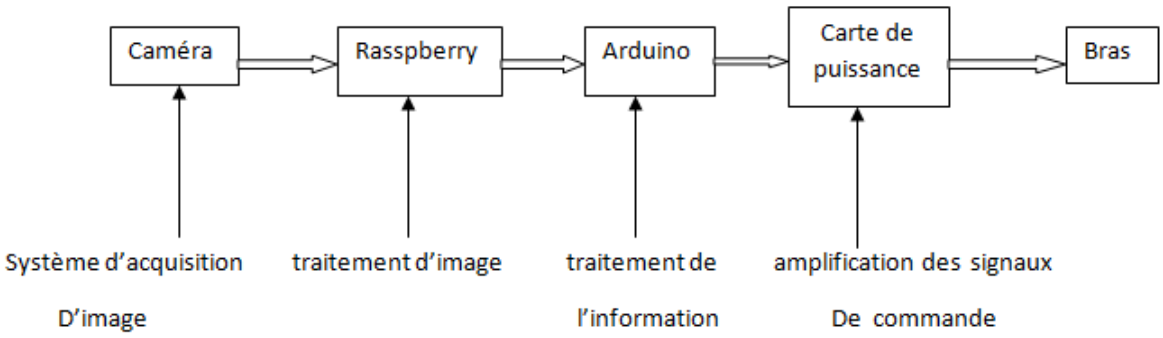

Figure 2-1 : diagramme du système global.

2.2 Hardware utilisé

2.2.1 Raspberry

Le Raspberry Pi est un nano-ordinateur mono carte à processeur ARM conçu par le programmeur britannique de jeux vidéo David Braben, dans le cadre de sa fondation Raspberry Pi [4].

Cet ordinateur, qui a 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-Debian et des logiciels compatibles. Mais également avec les OS Microsoft Windows : Windows 10 IoT Core [5] et Android Pi [6].

Il est fourni nu (carte mère seule, sans boîtier, ni alimentation, ni clavier, ni souris, ni écran) dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération. Néanmoins des kits regroupant le tout en un, existent.

En 2006, les premiers prototypes du Raspberry Pi sont développés sur des micro-Atmel ATMēga 644. Le schéma et le plan du circuit imprimé sont rendus publics [5]. Cet ordinateur s'inspire du BBC Micro d'Acorn Computer (1981) destiné à encourager laprogrammation. Le premier prototype ARM est intégré dans un boîtier de la même taille qu'une clé USB avec un port USB d'un côté et un port HDMI de l'autre.[6][7][8].

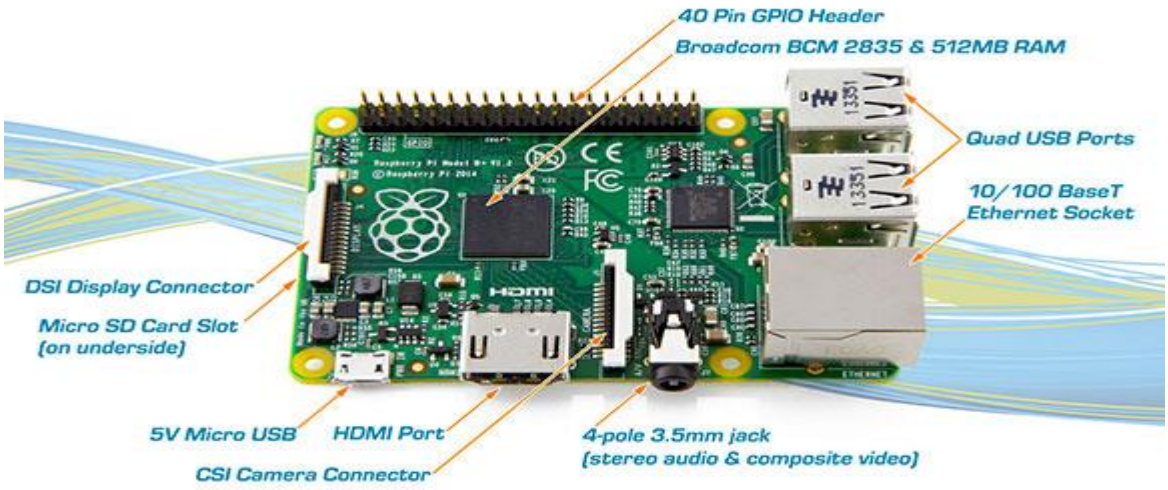

Figure 2-2 : les E/S du Raspberry pi2 [web 3]

Plusieurs modèles sont disponibles sur le marché, la figure 2 .2 représente le Raspberry Pi 2 modèle B basé sur le processeur Broadcom **BCM2835** avec architecture ARM cortex 7 quad-core cadencé à 900 Mhz avec une mémoire volatile de 1 Go il possède :

- 4 x 2 ports USB
- 4 sorties stéréo et vidéo composite
- Port HDMI
- Port caméra CSI pour connecter la caméra Raspberry Pi
- Port affichage DSI pour connecter l'écran capacitif Raspberry Pi
- Port micro SD pour lancer le système d'exploitation et la sauvegarde des données.

a) Spécifications techniques

- **Puce (SoC)** : Broadcom BCM2835
- **Processeur** : 900MHz quadricoeur ARM Cortex-A7 (jeu d'instructions ARM v7)
- **Processeur graphique** : Broadcom VideoCore IV63, OpenGL ES 2.0, MPEG-2 et VC-1 (avec licence), 1080p30 h.264/MPEG-4 AVC High-profile décodeur et encodeur
- **Mémoire (SDRAM)**: 1GB LPDDR2
- **Nombre de ports USB 2.0** : 4
- **Sorties vidéos** : HDMI et Jack (via convertisseur Composite PAL/NTSC)
- **Sorties audio** : stéréo Jack 3,5 mm (sortie son 5.1 sur la prise HDMI) et Composite
- **Sauvegarde des données** : Carte MicroSD
- **Connexion réseau** : 10/100 Ethernet
- **Périphériques** : 40 × GPIO
- **Alimentation** : 5 volt via Micro-B USB
- **Dimensions** : 85,60 mm × 53,98 mm × 17 mm
- **Poids** : 45 g

b) les GPIO

GPIO est l'acronyme anglais de (*General Purpose for Input and Output),* comme toute carte de développement il existe sur le Raspberry des port E /S qui possède une adresse définie dans la mémoire, elle sert à interagir avec le monde extérieure.

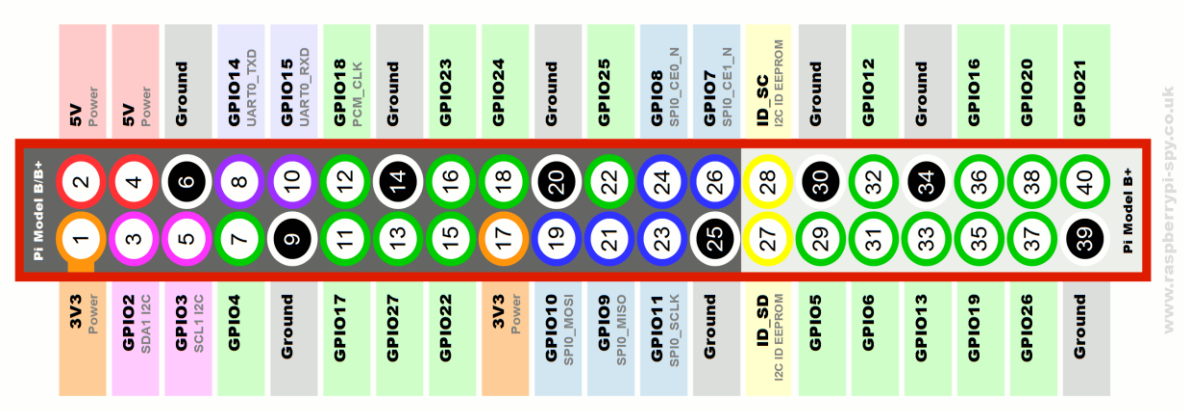

Figure 2-3 : GPIO (general purpose for input and output) du Raspberry pi2 [web 4].

Comme illustré dans la figure 2-3 ci dessus, il ne faut pas confondre entre le nombre de pin et le nombre attribué à la pin par le nombre de GPIO, on distingue 7 types GPIO dans le modèle B.

- les pins (2 ,4) : 5 volts
- les pins (6, 9, 14, 20, 25, 30, 34,39) : grounds
- la pin 1 :3.3 volts
- les pins (3,5) : la communication série synchrone (I2C)
- les pins(8,10) : la communication série asynchrone
- les pins(27,28) : le même protocole I2C mais dédié à l'EPROM c'est une nouveauté par rapport aux versions précédentes.
- les pins (24, 26, 19, 21,23) : communication synchrone (ISP)

Les autres pins peuvent être programmés en entrées / sorties logiques et en sortie analogique, mais pas d'entrées analogiques, donc pour lire une valeur analogique comme (la température, l'humidité, la pression …) il faut utiliser un convertisseur analogique numérique CAN comme le MCP3208 de chez Microchip avec une liaison SPI.

2.2.2 L'Arduino

L'Arduino est une carte de développent open source basée sur des microcontrôleurs d'architecture AVR pour certains modèles et ARM pour d'autres.

Elle est conçue par un groupe de développeurs italiens [9], mais les premiers prototypes ne ressemblent pas à ceux d'aujourd'hui par exemple (le port de communication série RS232 qui est remplacé par le port USB)

De nos jours en trouve plusieurs modèles sur le marché, le tableau ci dessous montre quelques modèles [10].

Arduino	**Microcontrôleurs**	**Nombres de broches**
Diecimila	ATmega168	26
Duemilanove	ATmega168/328P	26
Uno	ATmega328P	26
Leonardo	ATmega32U4	39
Mega	ATmega1280	85
Mega2560	ATmega2560	85
Due	Atmel SAM3X8E	78
Nano	ATmega168 ou ATmega328	28

Tableau 2-1 **:** liste des Arduino avec leurs microcontrôleurs et nombres E/S

Nous utiliserons dans ce projet la carte Mega 2560.

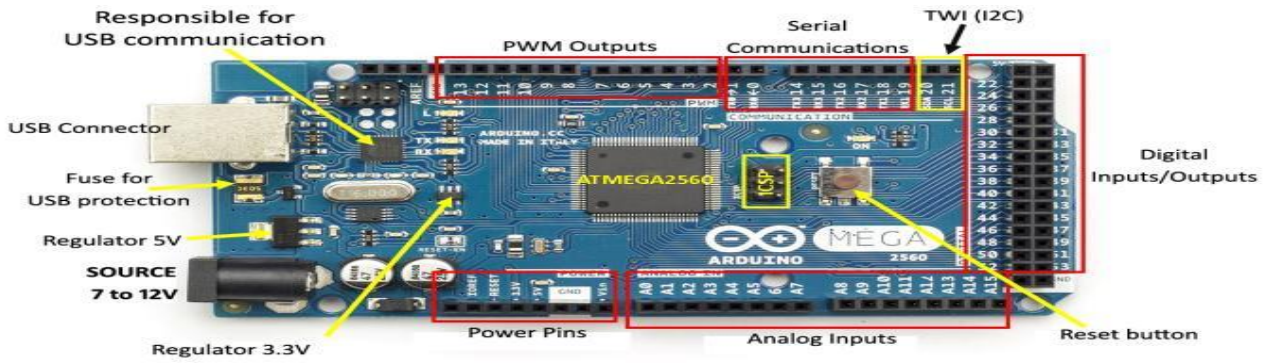

Figure 2-4 : les éléments principaux de la carte Arduino mega [web 5]

Le tableau suivant représente ses spécifications techniques

Micro-contrôleur	Flash Ko	EEPROM Ko	SRAM Ko	Broches E/S/N	Avec PWM	Broches E/A	Courant (mA) Vin=9v	Type USB	Dimensions mm
ATmega 2560	256	4	8	54	15	16	58	ATmega 8U2	101,6 x 53,3

Tableau 2-2 : spécifications technique de la carte Arduino mega

La programmation et l'implémentation d'un code sur n'importe quelle carte Arduino reste basée sur le même principe car tous les microcontrôleurs utilisés dans ces cartes sont programmés par un *boot loader* pour ne pas utiliser un programmateur dédié à chaque carte [11]. De plus un environnement de travail multi plateforme développé sous java, simple à utiliser et qui contient toutes les catégories des cartes existantes comme illustré en dessous Figure 2-5

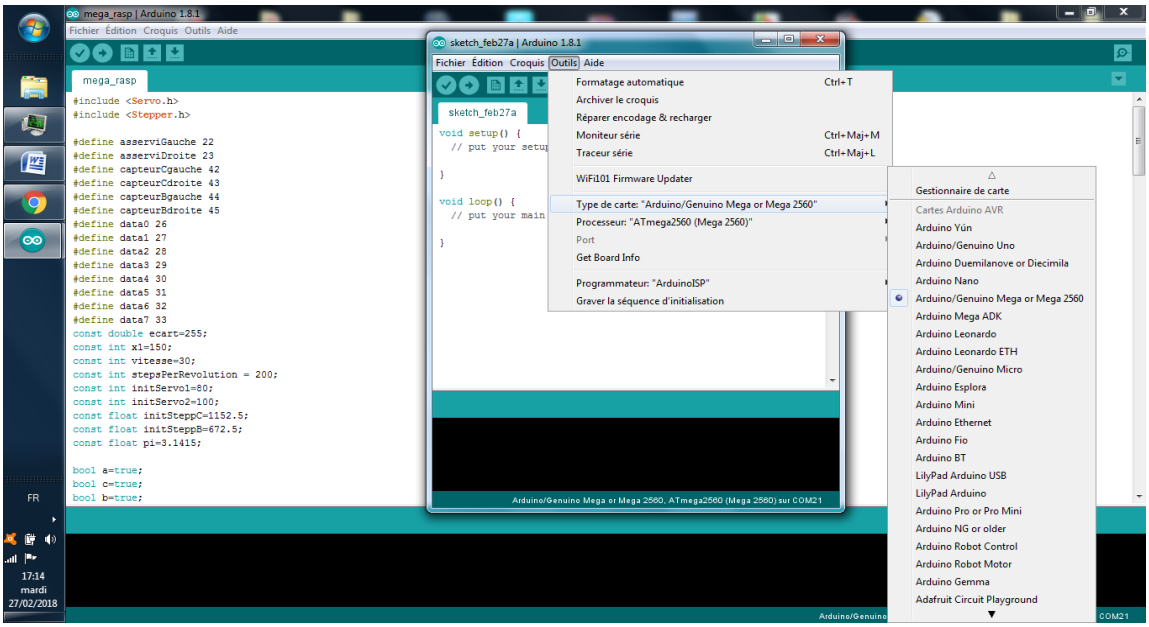

Figure 2-5: IDE (integrated devolopment environement) d'Arduino.

Ce logiciel est téléchargeable sur le site officiel Arduino.cc, on trouve deux versions portables qui n'ont pas besoin d'être installées.

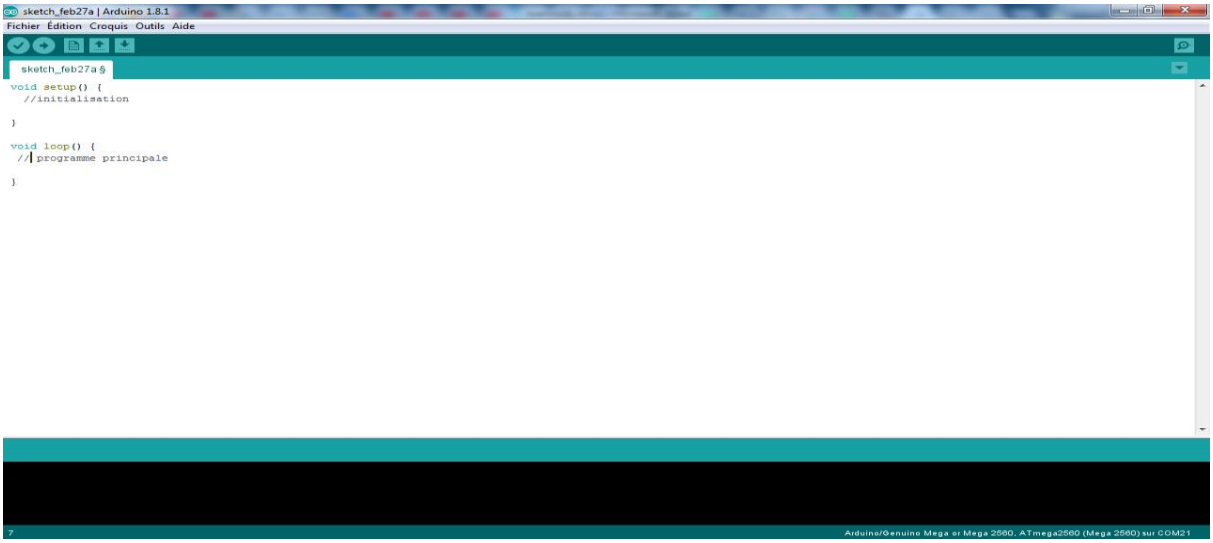

Figure 2-6 : environnement de programmation.

Le langage de programmation utilisé est le C/C++, compilé avec avr-g++ [12] pour générer le fichier binaire, est lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino, à toute personne maîtrisant le C ou le C++.

Reste à compiler à l'aide du bouton encerclé par le rouge dans la Figure 2-7 ci-dessus

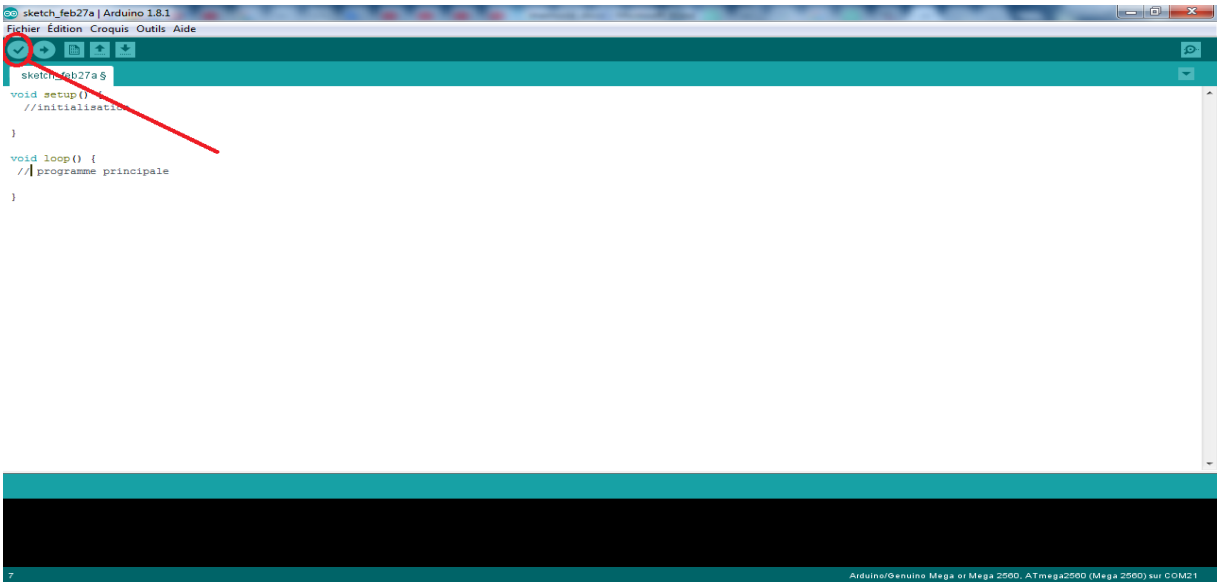

Figure 2-7 : Bouton de compilation du programme.

Il suffit de choisir le type d'Arduino dans l'onglet outils >> type de carte comme illustré dans la figure 2-5 après avoir branché la carte, dans le même onglet outils, si le champ port ne s'affiche pas c'est que notre ordinateur n'a pas bien installé les driver de la carte. sous Windows il suffit d'aller dans panneau de configuration >>gestionnaire de périphériques pour installer le driver Figure 2-8.

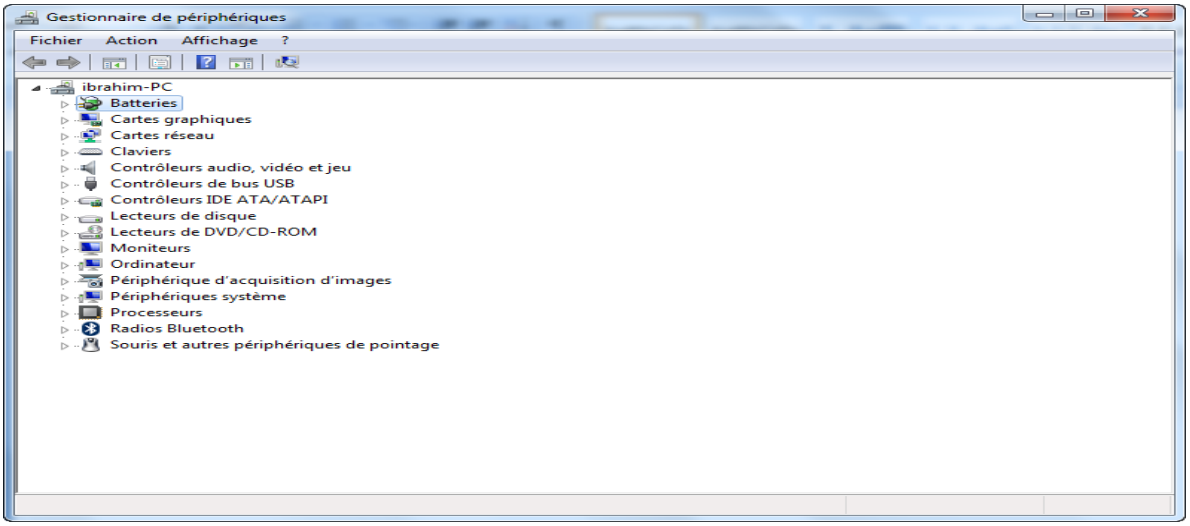

Figure 2-8 : Gestionnaire des périphériques sous Windows .

Si la carte est branchée et qu'elle n'est pas installée, il suffit de mettre à jour le driver [12]. Une fois le driver installé, il ne reste plus qu'à envoyer le code sur la carte en appuyant sur le bouton televerser Figure 2-9

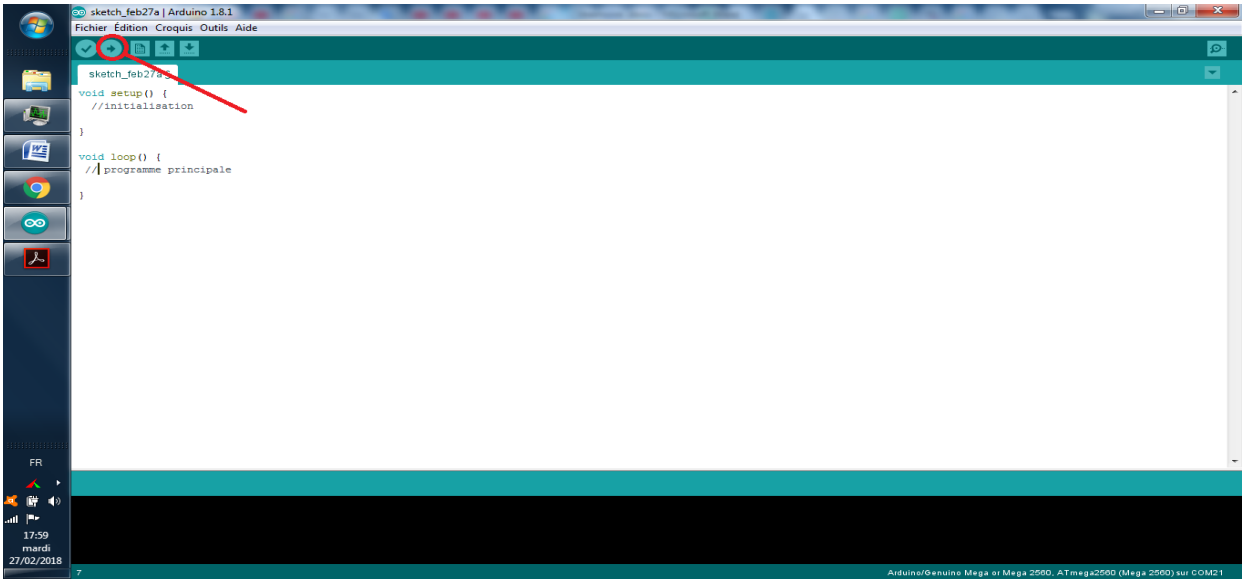

Figure 2-9 : Bouton de téléverssement.

2.3 Software sur la carte Raspberry

Il existe plusieurs versions de systèmes d'exploitation compatibles incluant diverses distributions linux et Windows, le choix de l'OS repose souvent sur les besoins de l'utilisateur (média-center, education, serveur, test et hacking, rétro-gaming et , système Raspbian classique,...). La plupart des OS sont des distributions basées sur le noyau linux.

2.3.1 Le noyau Linux

Le noyau Linux est un noyau de système d'exploitation de type UNIX. C'est un logiciel libre développé essentiellement en langage C par des milliers de bénévoles et salariés contribuant sur Internet.

Le noyau est le cœur du système, c'est lui qui s'occupe de fournir aux logiciels une interface de programmation pour utiliser le matériel. Il a été créé en 1991 par Linus Torvalds pour les compatibles PC. Initialement conçu pour l'architecture de processeur x86, il a ensuite été porté sur de nombreuses autres, dont m68k, PowerPC, ARM, SPARC et MIPS. Il s'utilise dans une très large gamme de matériel, des systèmes embarqués aux superordinateurs, en passant par les téléphones mobiles et ordinateurs personnels.

Ses caractéristiques principales sont d'être multitâches et multi utilisateurs. Il respecte les normes POSIX ce qui en fait un digne héritier des systèmes UNIX. Au départ, le noyau a été conçu pour être monolithique. Ce choix technique fut l'occasion de débats enflammés entre Andrew S. Tanenbaum, professeur à l'université libre d'Amsterdam qui avait développé Minix, et Linus Torvalds. Andrew Tanenbaum arguant que les noyaux modernes se devaient d'être des micro-noyaux et Linus répondant que les performances des micronoyaux n'étaient pas bonnes. Depuis sa version 2.0, le noyau, bien que n'étant pas un micro-noyau, est modulaire, c'est-à-dire que certaines fonctionnalités peuvent être ajoutées ou enlevées du noyau à la volée (en cours d'utilisation). [14][15][16][17]

2.3.2 Implémentation sur la carte

Il existe quatre-vingt-dix distributions qui tournent autour du noyau Linux, mais trois grandes d'entre elles, sont les plus populaires et les plus essentielles. On trouve (Debian, Slackware et Redhat). Mais le nombre des héritiers de ses trois grande familles est

incomptable, on donne comme exemple Ubuntu est l'héritier de Debian , Draco de Slackware et Fedora de Redhat. Parmi les quatre-vingt-dix distributions il y en a qui n'ont pas beaucoup de famille comme Android pour les téléphones portable et Gee XBOX pour la console de jeux XBOX .

Mais le problème qui se pose c'est que pour booter un système embarqué avec l'une des distributions il faut chercher l'image dédiée au processeur si elle existe, par exemple si on cherche à installer une distribution sur un PC populaire il faut chercher la distribution dédiée au processeur AMD ou bien à Intel, pour le Raspberry pareil il faut chercher la distribution dédiée à l'architecture ARM.

Nous avons choisi pour ce projet la distribution Ubuntu MAT qui est disponible sur le site officielle *ubuntu-mat.org*. Un système d'exploitation pour Raspberry doit être booter sur une carte mémoire , il faut utiliser un logiciel qui s'occupera de le monter sur une carte SD en mode bootable , pour cela on peut utiliser le programme open source Win32DiskImager disponible sur le site *sourceforage.net*.

Une fois la carte SD préparée, il suffit de l'insérer sur Raspberry et démarrer , le système demande d'introduire quelques paramètres comme le mot de passe d'administrateur figure 2 -10

Figure 2-10 : accueil du système d'exploitation ubuntu.

2.3.3 OpenCV

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels de vision artificielle et d'apprentissage automatique. La bibliothèque dispose de plus de 2500 algorithmes optimisés qui comprennent un ensemble complet d'algorithmes de vision artificielle et d'apprentissage automatique classiques et à la pointe de la technologie. Ces algorithmes peuvent être utilisés pour détecter et reconnaitre (des visages, identifier des objets, classer des actions humaines …etc)

Opencv est une librairie open source développée en C/C++, elle a été conçu pour l'efficacité de calcul et avec un fort accent sur les applications en temps réel, elle dispose d'interfaces C ++, Python et Java et prend en charge Windows, Linux, Mac OS, iOS et Android [19]

c) Installation d'Opencv

L'installation d'OpenCV dépend de deux facteurs essentiels :

- le système d'exploitation sur lequel on travaille.
- le langage de développement qu'on souhaite utiliser.

Dans le cas du système linux avec C/C++ comme langage il faut d'abord 'installer le compilateur gcc qui s'occupe de la génération du fichier binaire. et aussi un logiciel qui peut gérer le processus de construction d'un logiciel tel que cmake.

- **Cmake**

CMake est un logiciel libre et multiplateforme pour gérer le processus de construction d'un logiciel en utilisant une méthode indépendante du compilateur. Il prend en charge les hiérarchies de répertoires et les applications qui dépendent de plusieurs bibliothèques. Il est utilisé conjointement avec des environnements de génération natifs tels que make , AppleXcode et Microsoft Visual Studio . Il a des dépendances minimales, ne nécessitant qu'un compilateur C ++ sur son propre système de construction [20]

d) Les étapes à suivre pour l'installation de OpenCV

D'abord on clique simultanément sur ctrl+t pour accéder au terminal comme illustrée dans la figure 2-13

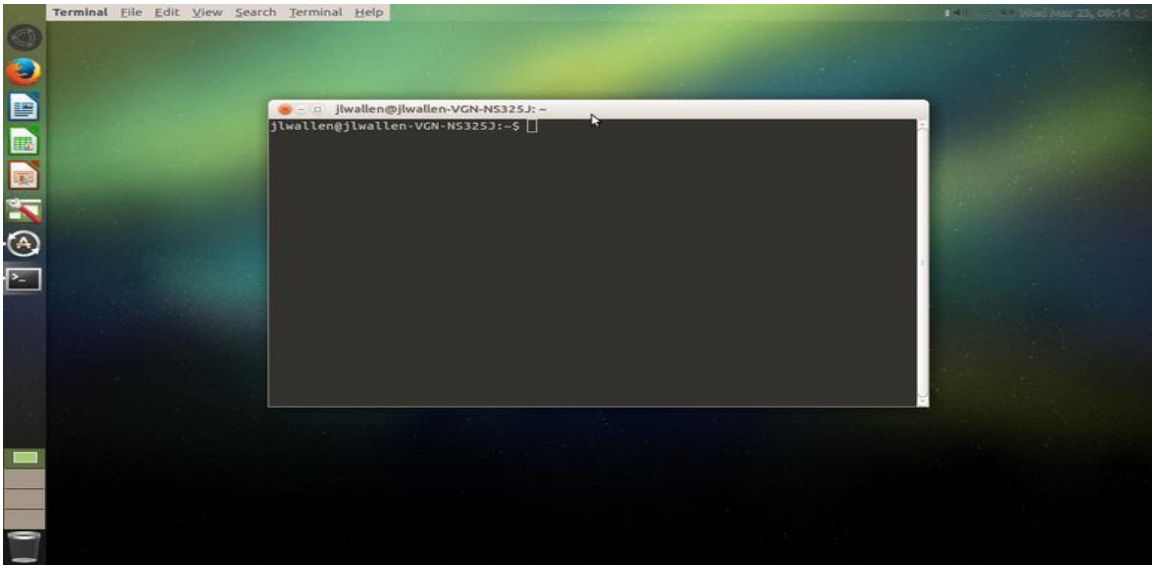

Figure 2-11: terminal du système.

L'installation se déroule en quatre étapes et fonctionne sur presque toutes les distributions de Debian.

- Etape 1

La première étape consiste à faire un update et un upgrade du système d'exploitation

- Sudo apt-get update - Sudo apt-get upgade

- Etape 2

Installation des dépendances

- sudo apt-get install build-essential cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev - sudo apt-get install python3.5-dev python3-numpy libtbb2 libtbb-dev - sudo apt-get install libjpeg-dev libpng-dev libtiff5-dev libjasper-dev libdc1394-22-dev libeigen3-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev sphinx-common libtbb-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libopenexr-dev libgstreamer-plugins-base1.0-dev libavutil-dev libavfilter-dev libavresample-dev

- Etape 3

Cette étape consiste a faire un clone d'OpenCV depuis la source github mais il faut passer au mode privilège (root).

- Sudo –s
- cd/opt
- git clonehttps://github.com/Itseez/opencv.git
- git clonehttps://github.com/Itseez/opencv_contrib.git

- Etape 4

Après avoir téléchargé OpenCV depuis source file il faut maintenant faire un build et l'installer.

- cd opencv
- mkdirrelease
- cd release
- cmake -D BUILD_TIFF=ON -D WITH_CUDA=OFF -D ENABLE_AVX=OFF -D WITH_OPENGL=OFF -D WITH_OPENCL=OFF -D WITH_IPP=OFF -D WITH_TBB=ON -D BUILD_TBB=ON -D WITH_EIGEN=OFF -D WITH_V4L=OFF -D WITH_VTK=OFF -D BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D OPENCV_EXTRA_MODULES_PATH=/opt/opencv_contrib/modules /opt/opencv/
- make -j4

Cette commande dépend directement de Raspberry, si la machine contient quatre cœur comme le Raspberry pi 2 modèle B, il faut exécuter avec –j4, si non si nous disposons par exemple d'un Raspberry pi zéro qui contenant un seul cœur il faut changer et utliser –j.

- make install
- ldconfig
- exit
- cd ~

Maintenant que tout s'est bien déroulé, il faut passer un test si l'installation a réussi, pour cela, il suffit de taper cette commande sur le terminal.

- pkg-config--modversion opencv

Si les étapes on été bien respectées, nous aurons sur le terminal la version d'OpenCV installé comme suit: 3.2 .x

Pour faire fonctionner un programme avec C/C++ sur cet environnement de travail on a deux choix :

Soit on crée un fichier avec extension .cpp et on utilise le compilateur g installé avec gcc, avec des lignes de commande précises, ou bien on utilise un IDE (*integrated devolopment environement),* un environnement de développement comme (éclipse, Qt, …ect) sauf qu'un IDE nécessite le chemin du compilateur à utiliser.

2.3.4 Compilation par ligne de commande

D'abord il faut créer un fichier avec extension .ccp en tapant ces lignes de commande sur le terminal.

```
- mircpp_test
- cd cpp_test
- touch test.cpp
```

Ces lignes créent un fichier test .cpp dans un fichier cpp_test, après l'avoir ouvert et écrit un programme, il faut ouvrir à nouveau le terminal et suivre le chemin du fichier en question, en utilisant la commande cd (change directory), puis pour faire la compilation tout en utilisant la bibliothèque OpenCV il faut ajouter le paramètre d'OpenCV [21]

```
g++ test.cpp –o tst `pkg-config --cflags --libs opencv`
```

Pour l'exécution

```
- ./tst
```

2.3.5 Utilisation d'un IDE

Comme déjà cité, il existe plusieurs IDE sur linux pour démarrer un projet en C /C++, pour notre cas on a utilisé Qt Creator version quatre.

- Installation de Qt v4 :

Après avoir ouvert le terminal on tape

- sudo apt-get install qtcreator

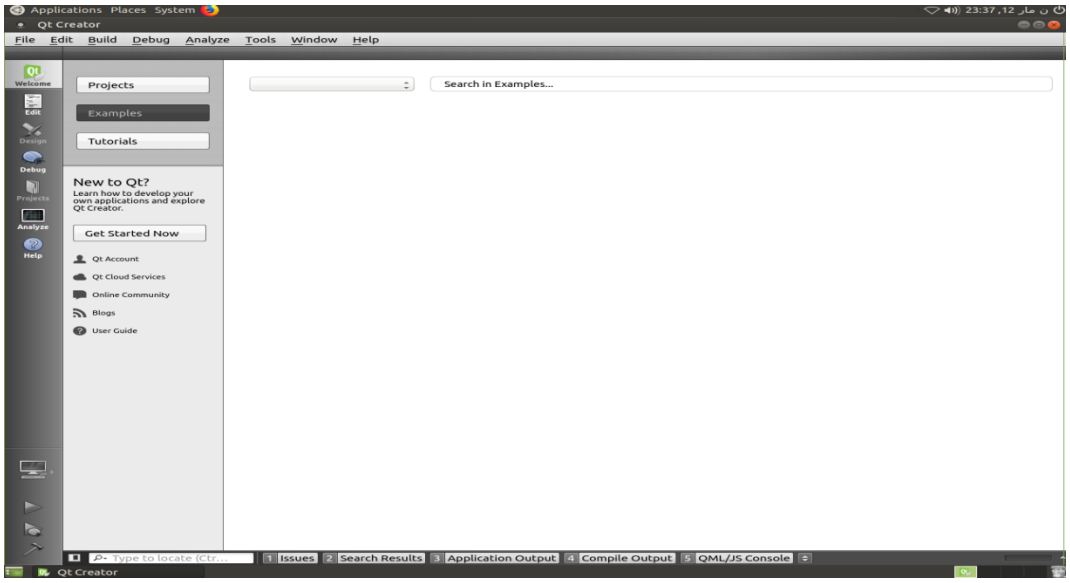

Figure 2-12 : interface de du logicielle QT.

Il faut savoir qu'un IDE nécessite le chemin du compilateur gcc, on va sur l'onglet Tools =>options, une fenêtre s'affiche, on choisit l'onglet build &run puis sur l'onglet Kits puis ajouter votre kits. Puis sur l'onglet compilé, donner le path du compilateur la figure 2 - 13 et 2-14 .

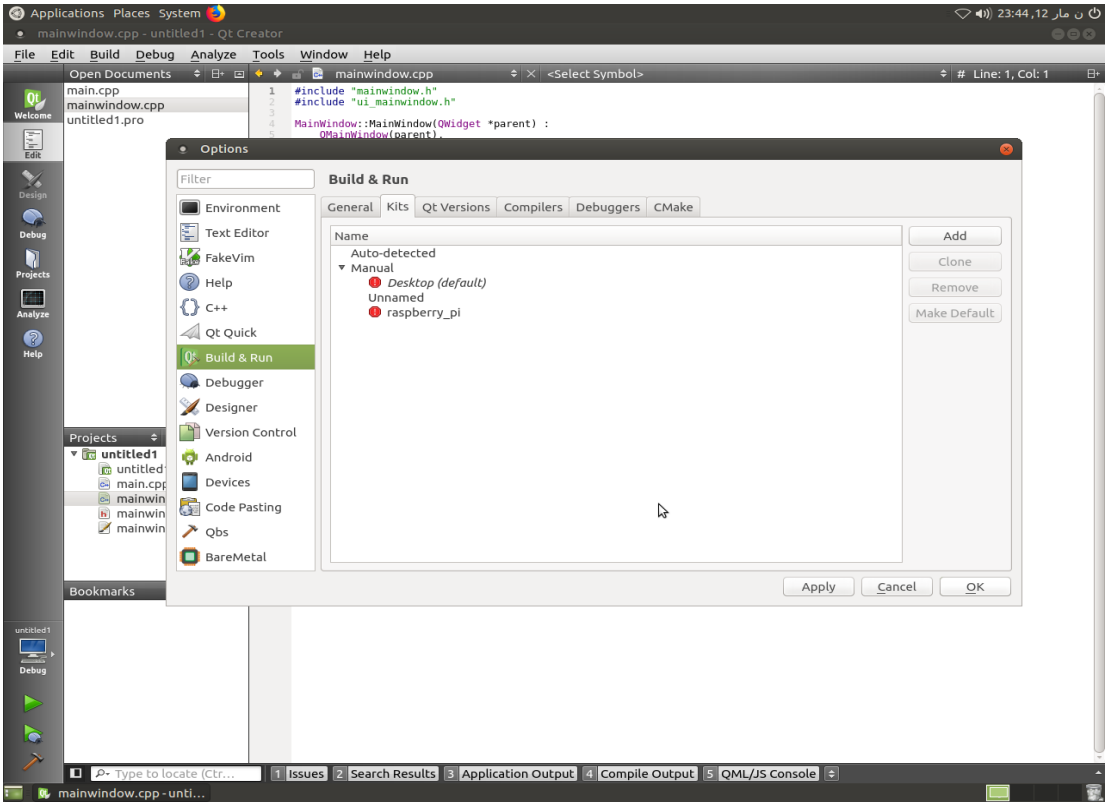

Figure 2-13 : choix du kits.

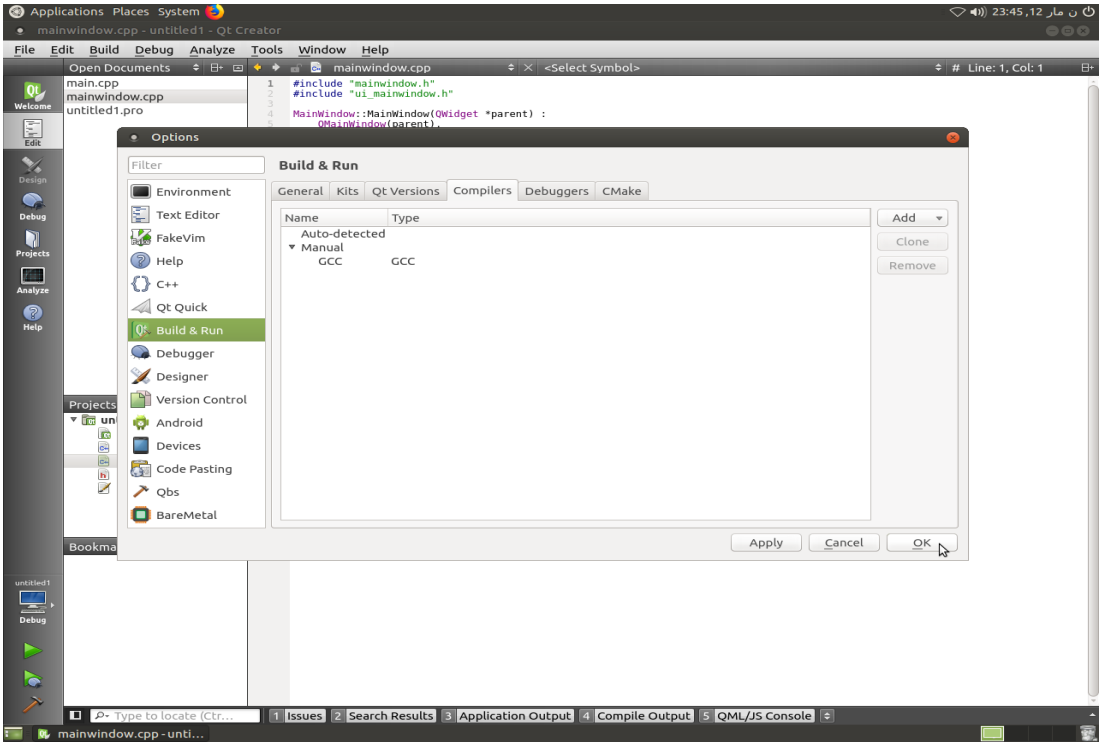

Figure 2-14 : ajout du path du compilateur.

Maintenant que le compilateur est en marche, on ouvre un nouveau projet, quatre fichiers se créent comme illustrée sur la figure 2-15

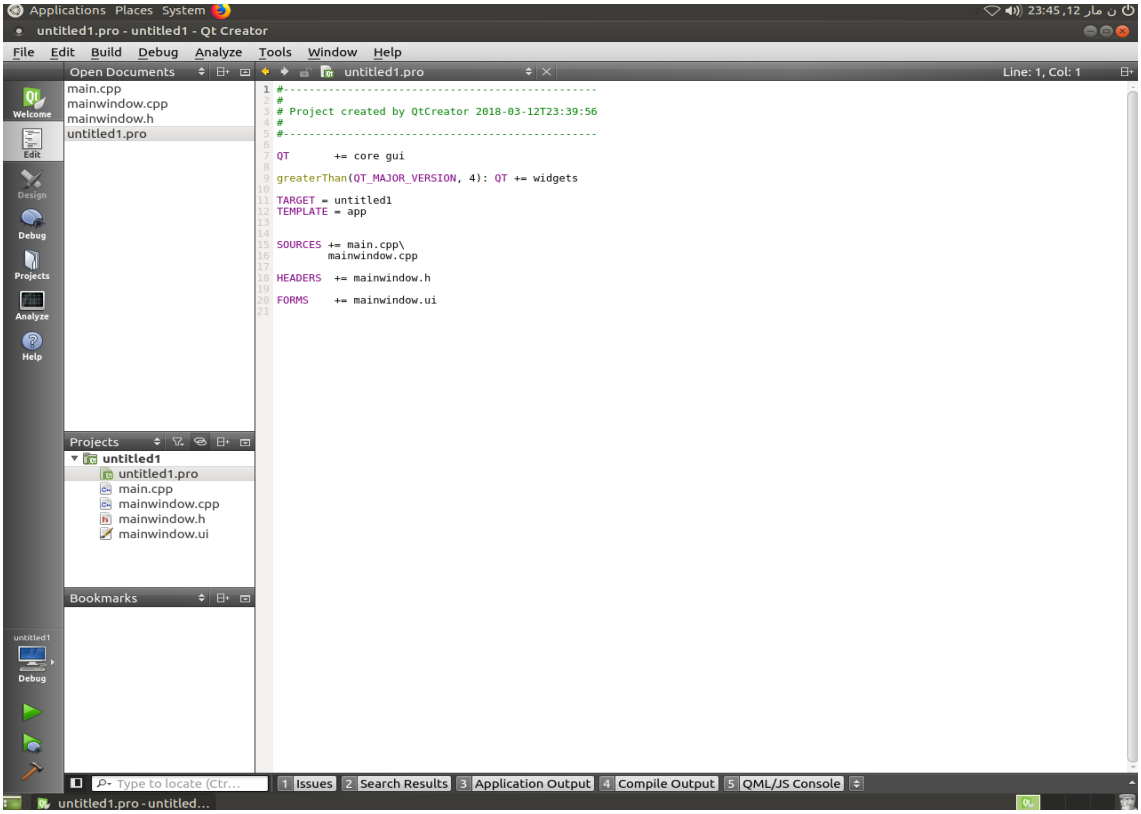

Figure 2-15 **:** ajout du chemin des libraires.

A gauche on voit main.cpp (programme principal), mainwindow.cpp (la classe de la fenêtre principale), mainwindow.h (fichier entête).

le fichier utititled.pro apparait sur la figure 2-15, dans lequel il faut ajouter les chemins de la librairie à utiliser en tapant ces lignes de code

- INCLUDEPATH+= ‘’/usr/local/include/’’
- INCLUDEPATH += /usr/lib
- LIBS += `pkg-config --cflags --libs opencv` // le paramètre utilisé pour la compilation d’OpenCV
- LIBS += -lwiringPi // le paramètre utilisé pour la compilation de wiringPi
- LIBS += -lcrypt

Avant d’ajouter la librairie WirinPi, il faut d’abord l’installer.

2.3.6 WirigPi

Lors de la conception de certains systèmes embarqués à base d'un Rapsberry, on a souvent besoin d'utiliser les GPIO de ce dernier , et puisque le Raspberry est un système plus complexe par rapport à un Arduino ou un microcontrôleur qui sont conçus justement à base d'entrés sorties, le Raspberry présente donc une difficulté pour la gestion des gpio vue que l'architecture de son processeur n'est pas conçue que pour cette tâche .

Heureusement des bibliothèques open source existe pour faciliter le contrôle des GPIO, comme la BCM2835 par Mike McCauley[24], mais elle est dédiée au processeur brodcom 2835 qui est le cœur du Raspberry pi 1, et elle nécessite une mise a jour car elle ne fonctionne que sur une partie des pins pour le Raspberry PI 2 et 3.

Il y a aussi la bibliothèque wiringpi crée par drogon [18] sous langage C /C++, elle peut être utilisé sur toutes les versions du Raspberrypi vue qu'elle prend l'architecture de BCM 2835, BCM2836 et BCM2837. [22]

Pour les développeurs qui ont l'habitude sur la plateforme Arduino, ils n'auront pas beaucoup de problèmes avec la syntaxe, vu que presque toutes les commandes ont été inspirées via cette dernière

e) Installation de wiringpi

Avant de commencer l'installation il faut s'assurer que wiringpi n'est pas installé, pour cela

```
- gpio –v
```

Si vous obtenez quelque chose cela veut dire qu'elle est déjà installée ou une bibliothèque similaire, alors il faut la supprimer avec :

```
sudo apt-get purge wiringpi
hash –r
```

Si vous n'avez pas le GIT vous pouvez l'installer avec :

```
sudo apt-get install git-core
```

Si vous rencontrez des erreurs, assurez -vous que votre rasspbery est à jour avec :

```
sudo apt-get update
sudo apt-get upgrade
```

Pour obtenir wiringpi avec git

```
cd
git clone git://git.drogon.net/wiringPi
cd ~/wiringPi
git pull origin
cd ~/wiringPi
./build
```

Maintenant vérifier en tapant :

```
Gpioreadall
```

Si vous voulez utiliser wiringpi sans IDE, la compilation avec ligne de commande nécessite un paramètre comme suit [22] :

```
gcc -Wall -o tst tst.c -lwiringPi
sudo ./blin
```

2.4 Conclusion

La coordination entre software et hardware représente la clé d'un bon agissement d'une conception électronique, dans ce chapitre nous avons décrit le hardware utilisé pour le projet ainsi que les softwares adoptés, ainsi que leurs installations et leurs configurations par étapes afin d'assurer un bon fonctionnement.

La mise en œuvre du bras industriel nécessite une géométrie et un calcul exact , cette dernière fera l'objet du chapitre suivant.

Chapitre 3 : conception et réalisation

3.1 Introduction

Il existe deux méthodes de brancher la caméra dans le système .

- Caméra dans le bras
- Caméra hors le bras

Ces deux configurations doivent être choisies au début de la conception , car ce choix va définir les méthodes et les algorithmes pour le traitement et l'asservissement .

Dans le cas de ce projet , la deuxième configuration a été choisie pour les raisons suivantes :

- Nous considérons le cas du bras utilisé en industrie (non mobile) .
- Simple à entretenir
- Plus de précision
- Moins d'encombrement sur le bras.

Le système proposé doit être capable de déplacer un objet d'une position donnée vers une autre en utilisant une camera. Cette camera fournit les informations nécessaires à l'asservissement des moteurs du bras de façon précise.

3.2 Conception du bras

La conception du bras est une partie très importante, elle doit respecter une géométrie bien définie comme la longueur de l'articulation, les angles maximaux et minimaux que chaque articulation peut supporter. Ces mesures vont tous être pris en compte lors de la conception des algorithmes de contrôle du bras.

Le bras est conçu avec trois degrés de liberté, deux pour les articulations avec des servo-moteurs et un troisième de rotation (rotation autour de l'axe Z) avec un moteur pas a pas.

3.2.1 Modèle géométrique du bras

Le positionnement d'un objet dans un plan peut se traduire de deux manières : coordonnées cartésiennes ou polaires. Pour un mouvement de rotation du bras , il est plus pratique d'utiliser les coordonnées polaires, Voir figure 3-1.

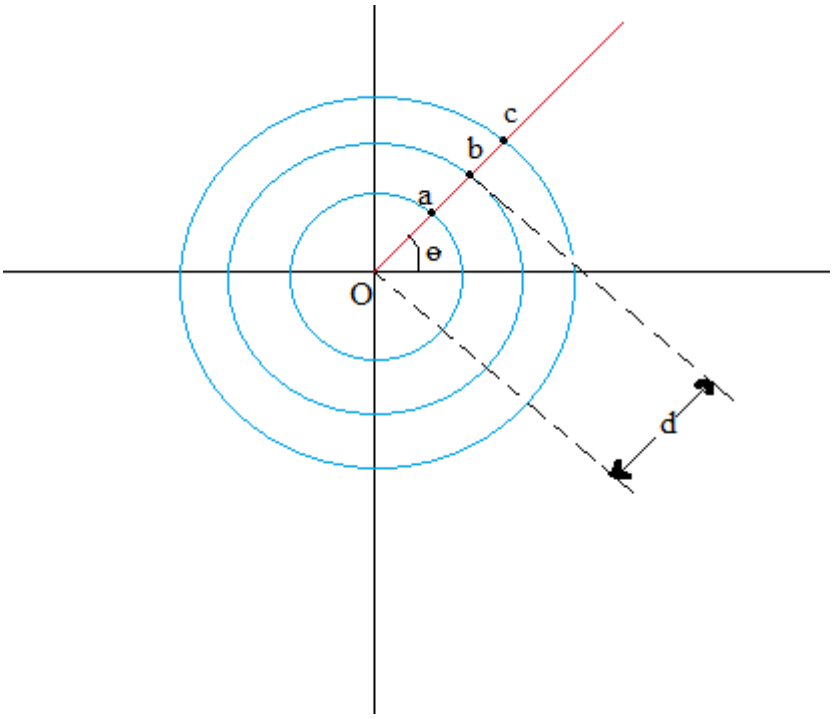

Figure 3-1 : schéma explicatif du positionnement d'objet.

Il reste un petit défi à relever ; imaginons que le bras ce trouve au centre 'O' et l'objet au point 'a' , d'abord le bras fait un angle 'ϴ' pour se positionner face à l'objet.

Il faut calculer la distance pour atteindre l'objet. cette distance dépendra des angles de rotation de chaque servo-moteur. La solution consiste à concevoir le bras comme un triangle isocèle. Voir figure 3-2 et figure 3-3

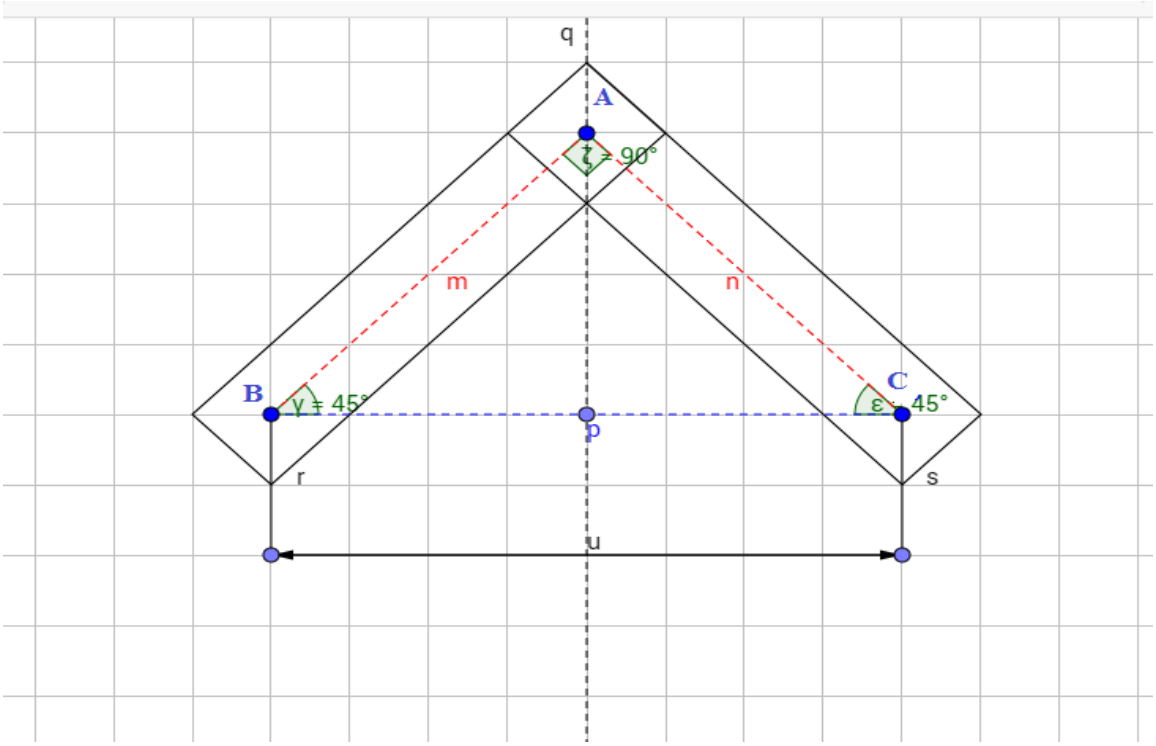

Figure 3-2 : dessin explicatif de la géométrie du bras.

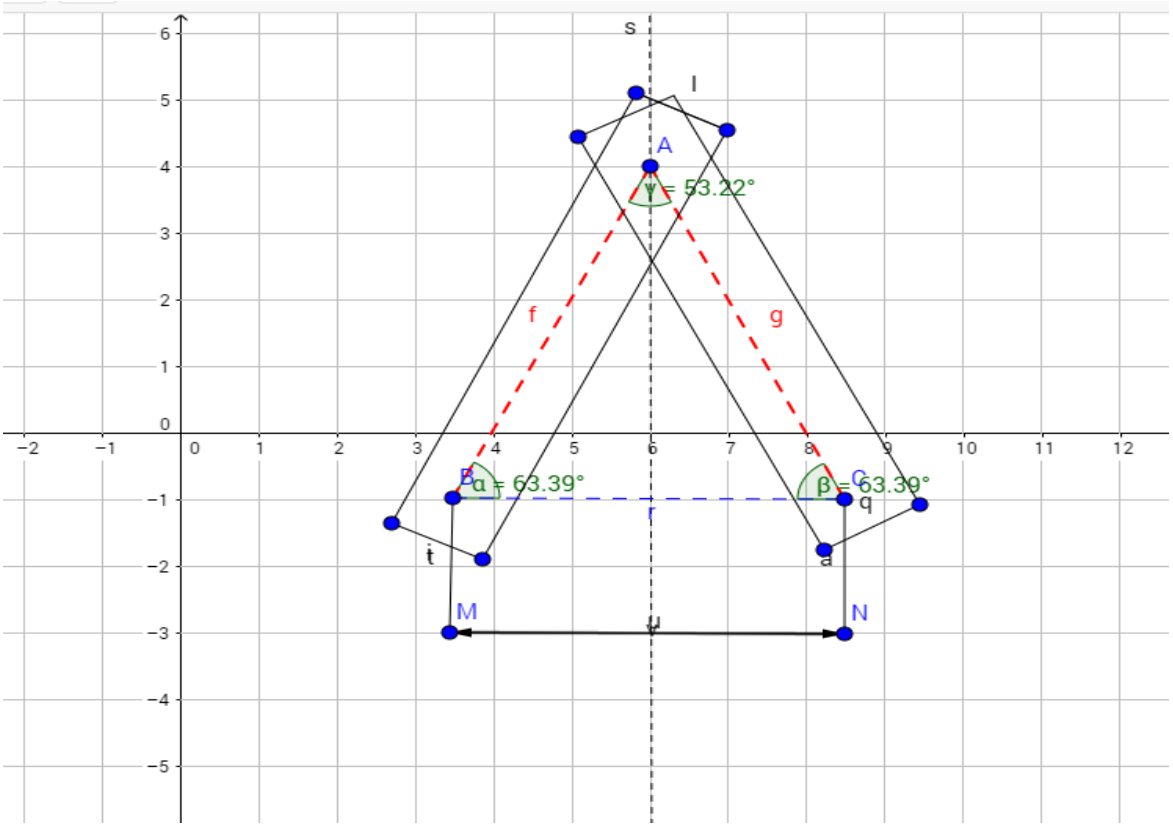

Figure 3-3 : dessin explicatif de la géométrie du bras.

Pour que le bras atteigne l'objet voulu il suffit de convertir la distance [u] en des angles bien précis. Les valeurs des angles seront transmis aux servo-moteurs.

Le théorème de Pythagore permet de faire ce calcul .

$\beta = \alpha = 180 * \arccos((u/2)/AC))/\pi$

$\gamma = 180 - (2 * \beta)$ (la somme des angles d'un triangle égale a 180 °)

Ainsi, il est possible d' atteindre l'objet à n'importe quelle position sur le champ de travail du bras .

3.2.2 Montage du bras par rapport à la webcam

Le bras est monté face à la camera avec une distance de 300 mm , la camera est muni d'un moteur pas a pas pour lui permettre de balayer un champ de vision équivalent à un demi cercle (figure 3-4).

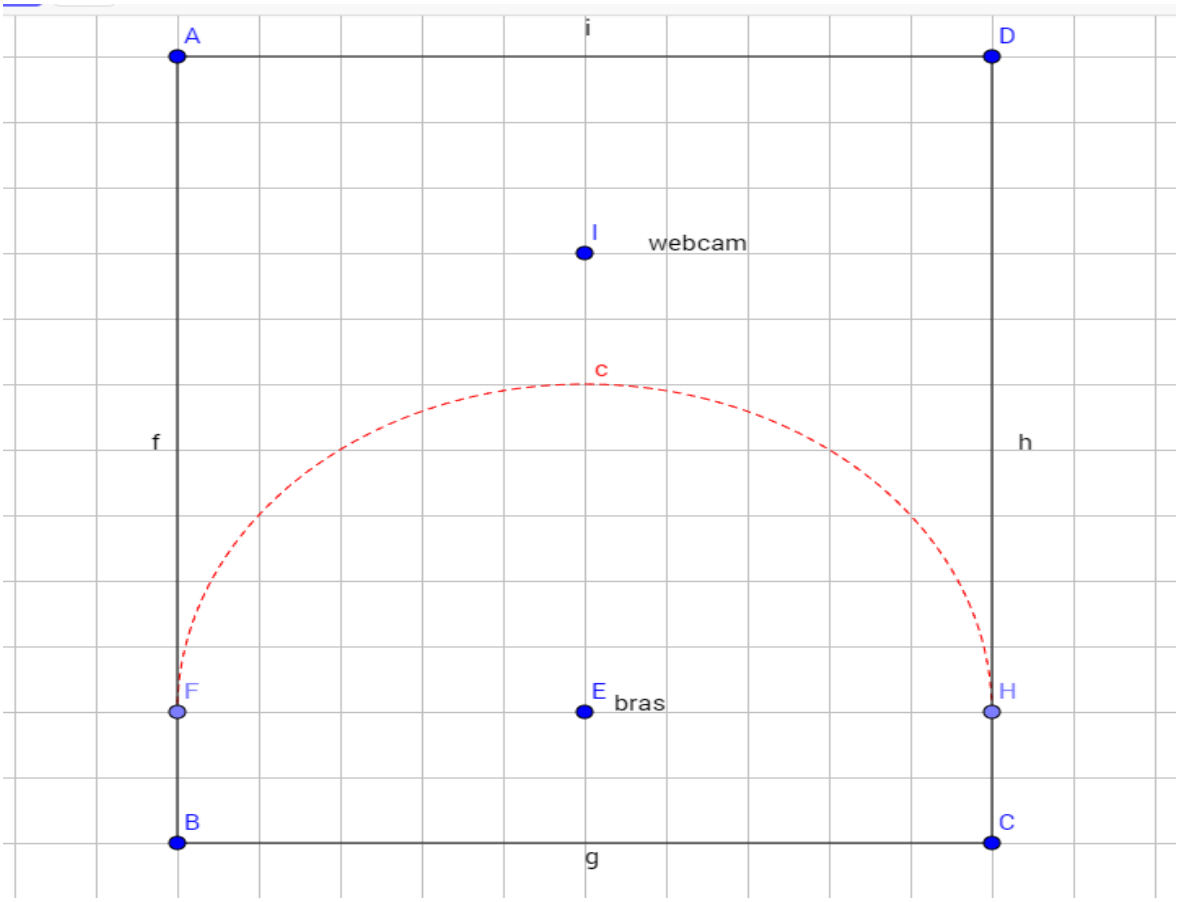

Figure 3-4 : vue de haut du système.

3.2.3 Calcul de distance et géométrie

Dans la figure 3-4 l'objet est positionné dans l'arc (c) , la prise de l'objet passe par deux étapes :

- Détection de l'objet et calcul de la distance camera- objet .
- Calcul de la distance entre le bras et l'objet (déduite à partir de l'image camera) .

3.2.4 Calcul de la distance (Camera-Objet)

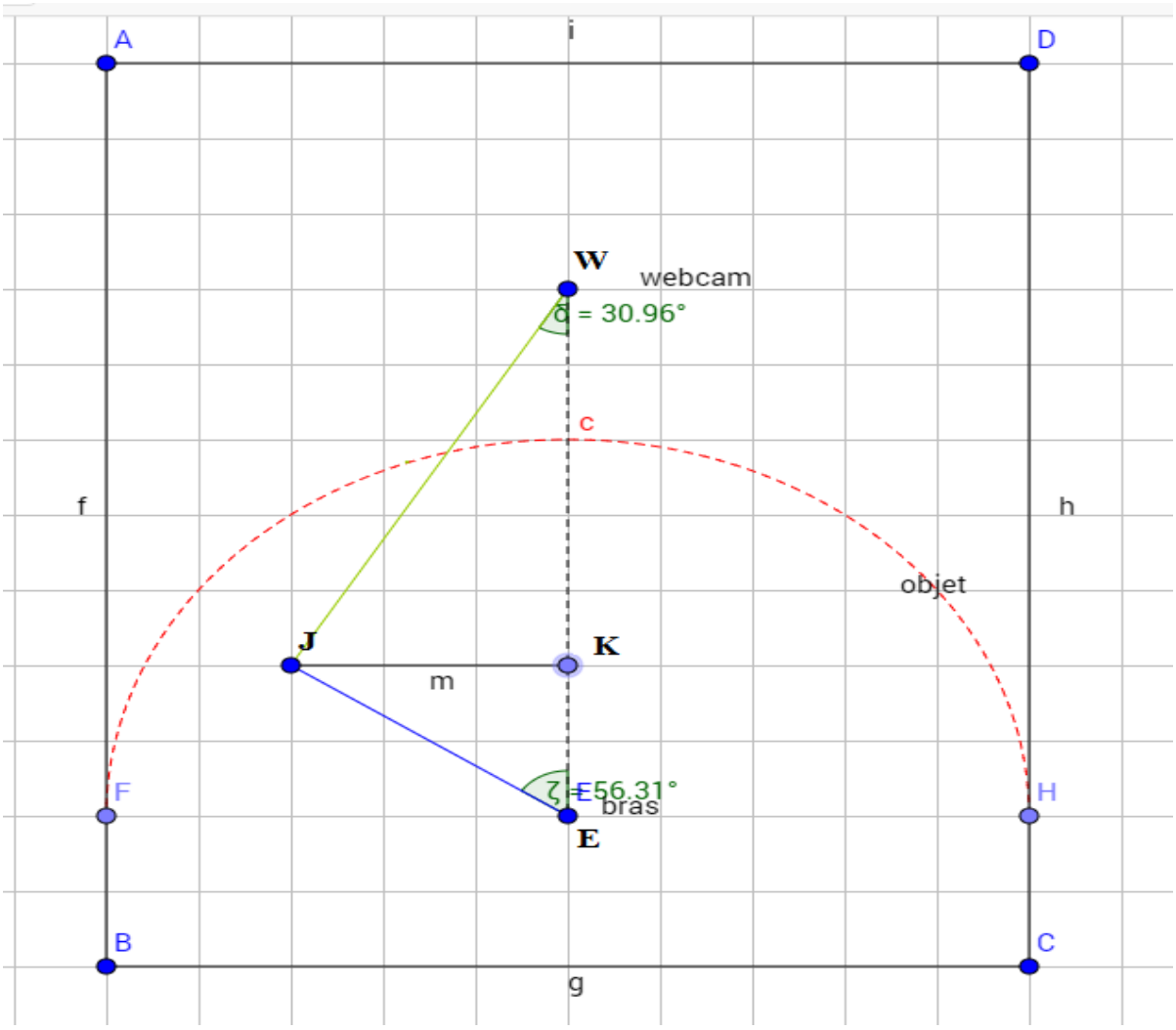

Figure 3-5: géométrie et calcul de distance.

L'objet se trouve au point 'J', les données disponibles sont :

- La distance objet camera [WJ].
- L'écart bras-camera [WE] (une constante introduit au début de la conception).

- L'angle du moteur pas a pas de la camera quand l'objet et au centre de l'image 'δ', (cet angle est calculé par un algorithme sur l'Arduino qui fait le comptage des pas et les converti en angle à l'aide d'une fonction qu'on va abordé dans la section (centrer l'objet dans l'image) .

En utilisant le théorème de Pythagore on peut déduire :

$$WK = WJ * \cos(\delta)$$

$$JK = WJ * \sin(\delta)$$

$$EK = WE - WK$$

$$\zeta = \tan(\frac{JK}{KE})$$

$$JE = \sqrt{EK^2 + JK^2}$$

Ou enfin :

$$JE = \sqrt{(WJ * \sin(\delta * \pi/180))^2 + (WE - \left(\sqrt{WJ^2 - (WJ * \sin\left(\delta * \pi/180\right))^2}\right))^2}$$

Et par raison de symétrie cette fonction s'applique sur tout le demi disque (c), voir figure 3-6.

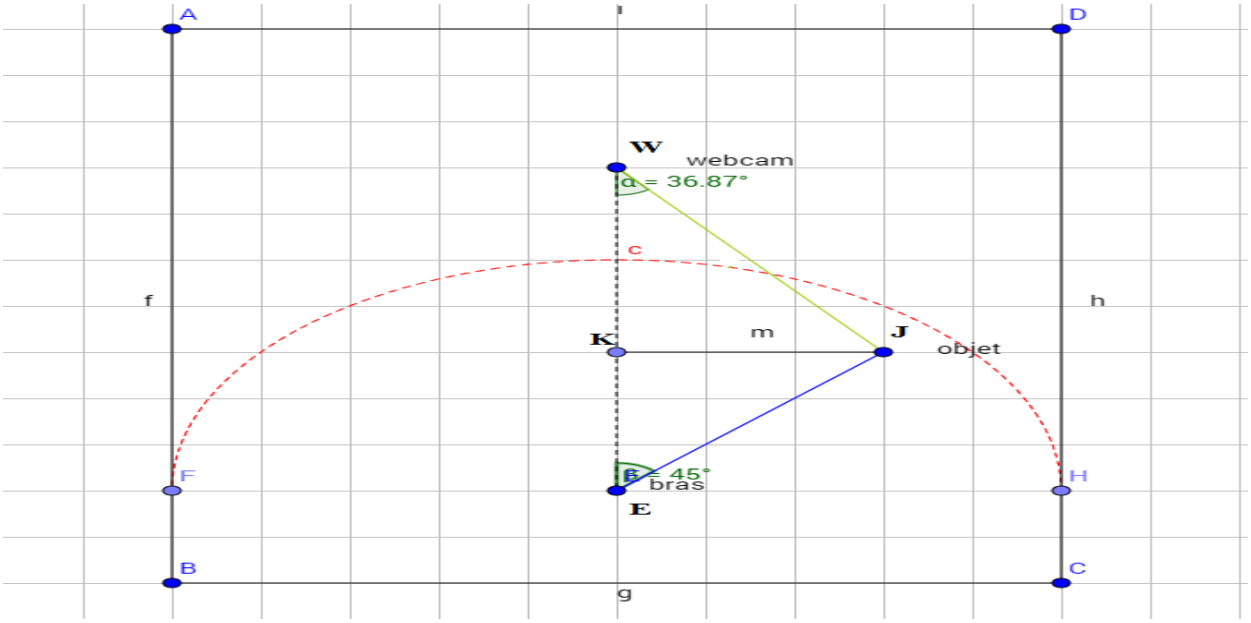

Figure 3-6 : géométrie et calcul de distance.

3.3 Montage général

3.3.1 Schéma synoptique

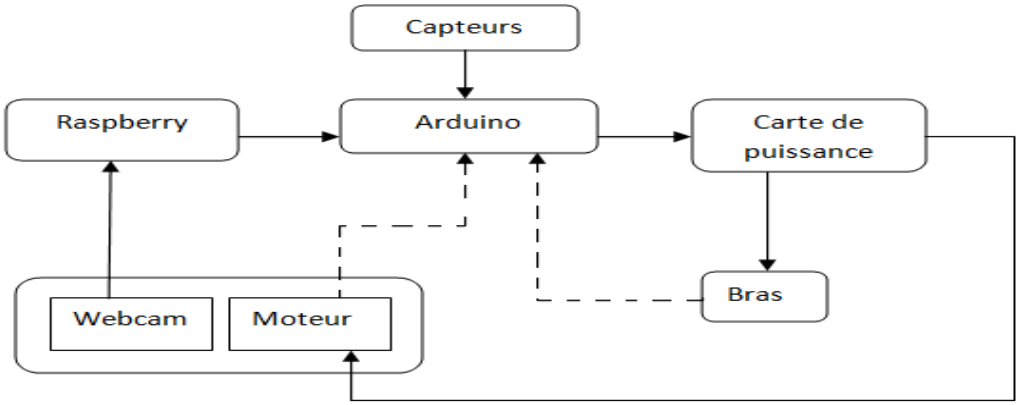

Figure 3-7 : schéma synoptique du système.

la carte de puissance permet de fournir les tensions de commande venant de l'Arduino pour attaquer les moteurs .

3.3.2 schéma électronique (carte de puissance)

la carte est à base de circuit intégré L293D de chez Texas instrument.

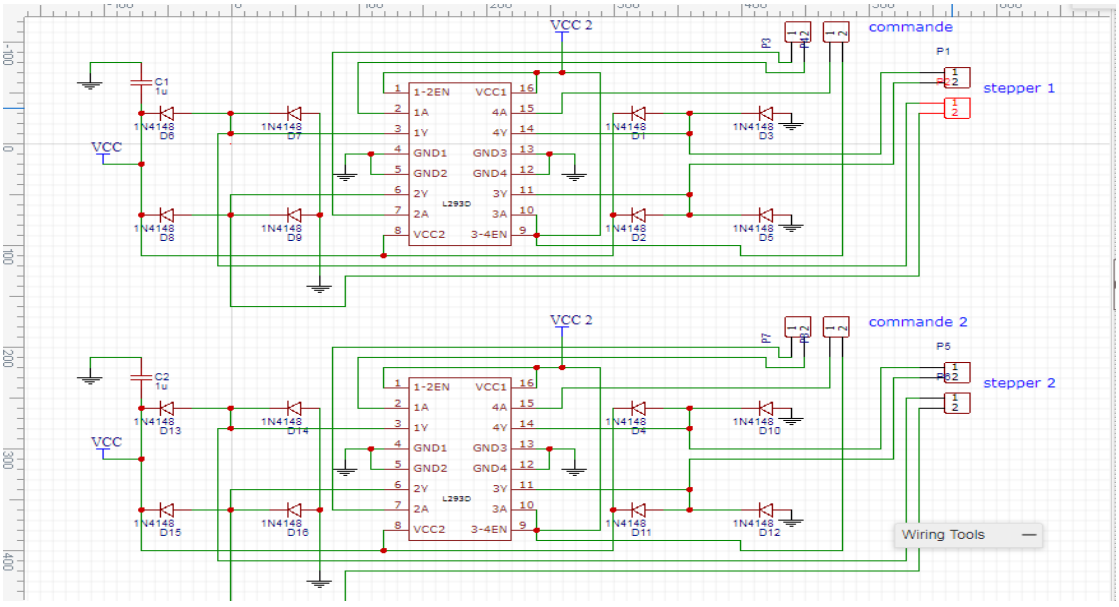

Figure 3-8 : schéma électronique (carte de puissance).

Le schéma a été développé sous easy EDA, qui est un logiciel de développement en ligne, après le test et vérification du circuit sur la breadboard, le circuit imprimé est réalisé à l'aide du logiciel TCI (figure 3-9).

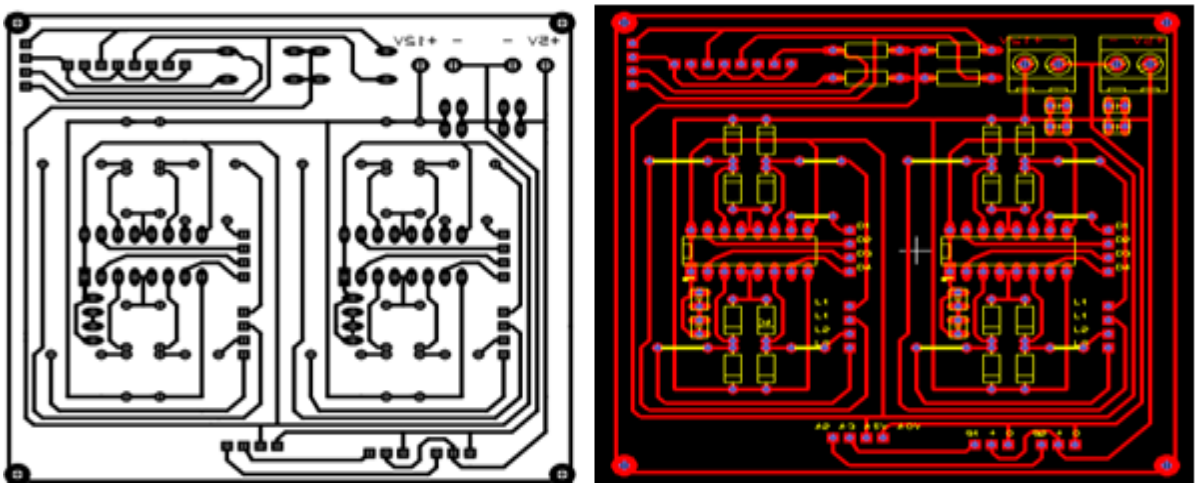

Figure 3-9 : a gauche le négatif a droite le circuit développé avec TCI.

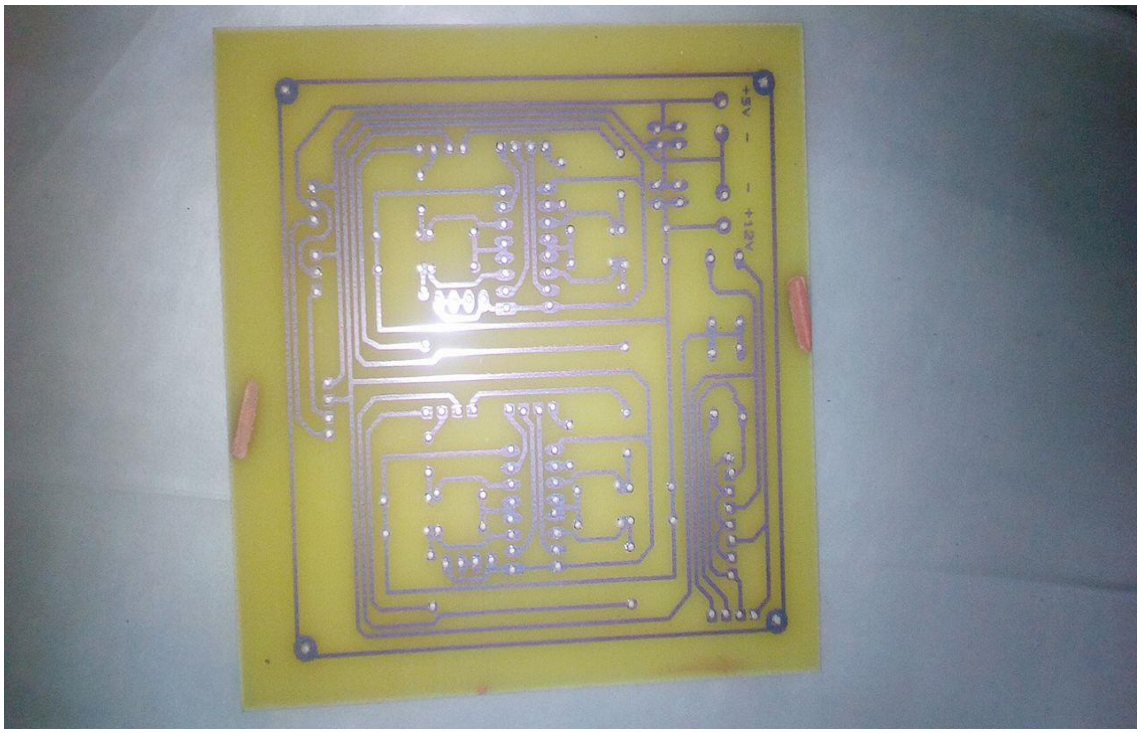

Figure 3-10 : Résultat de la gravure du PCB.

3.4 Algorithme et traitement d'images

3.4.1 Traitement d'images et seuillage (binarisation)

On peut faire un seuillage à partir d'une image en niveau de gris sinon à partir du plans HSV(*hue, saturation, value*) .

Les plans HSV et HSL(*hue, saturation, lightness*) , sont deux représentation alternatives du modèle RGB , conçues dans les années 1970 par des chercheurs en infographie pour s'aligner plus étroitement sur la vision de la couleur . Dans ces modèles , les couleurs de chaque teinte sont disposées en une coupe radiale , autour d'un axe central de couleurs neutres allant du noir en bas au blanc en haut . voir figure 3-11 .

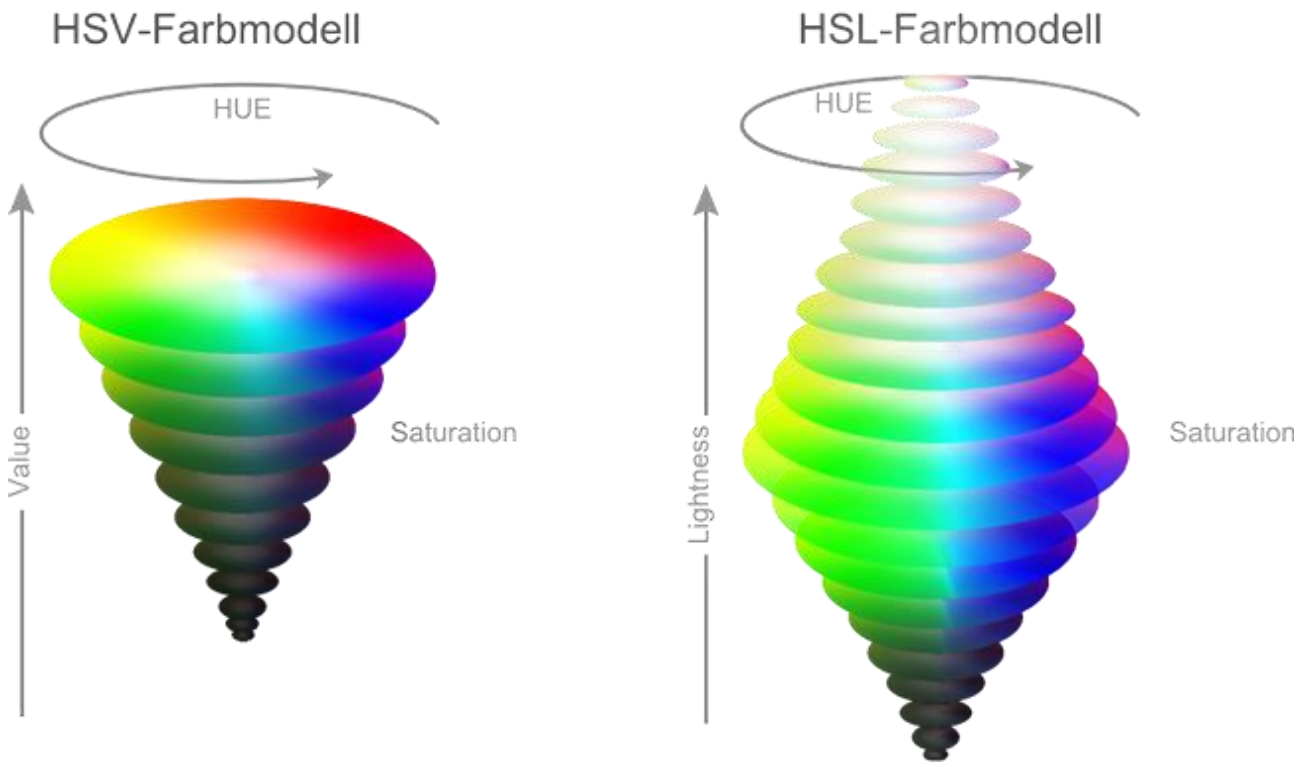

Figure 3-11 : a gauche le plan HSV à droite le plan HSL. [web 6].

Mais un seuillage du type HSV ou HSL reste toujours limité lors de la conception d'un système intelligent.

Il existe des algorithmes de seuillage à partir de l'étude d'histogramme comme l'algorithme d'otsu , qui choisi le seuil de façon automatique (figure 3-12).

La méthode de seuillage d'Otsu (voir annexe A) repose sur la sélection d'un minimum entre deux *modes*

- Fréquence $\omega = \sum_{i=0}^{T} P(i)$ $P(i) = n_i / N$ N: Nombre total de pixels
- Moyenne $\mu = \sum_{i=0}^{T} iP(i) / \omega$ **ni:** nombre de pixels dans le niveau *i*

T : nombre de niveaux de gris total (peut correspondre au seuil courant)

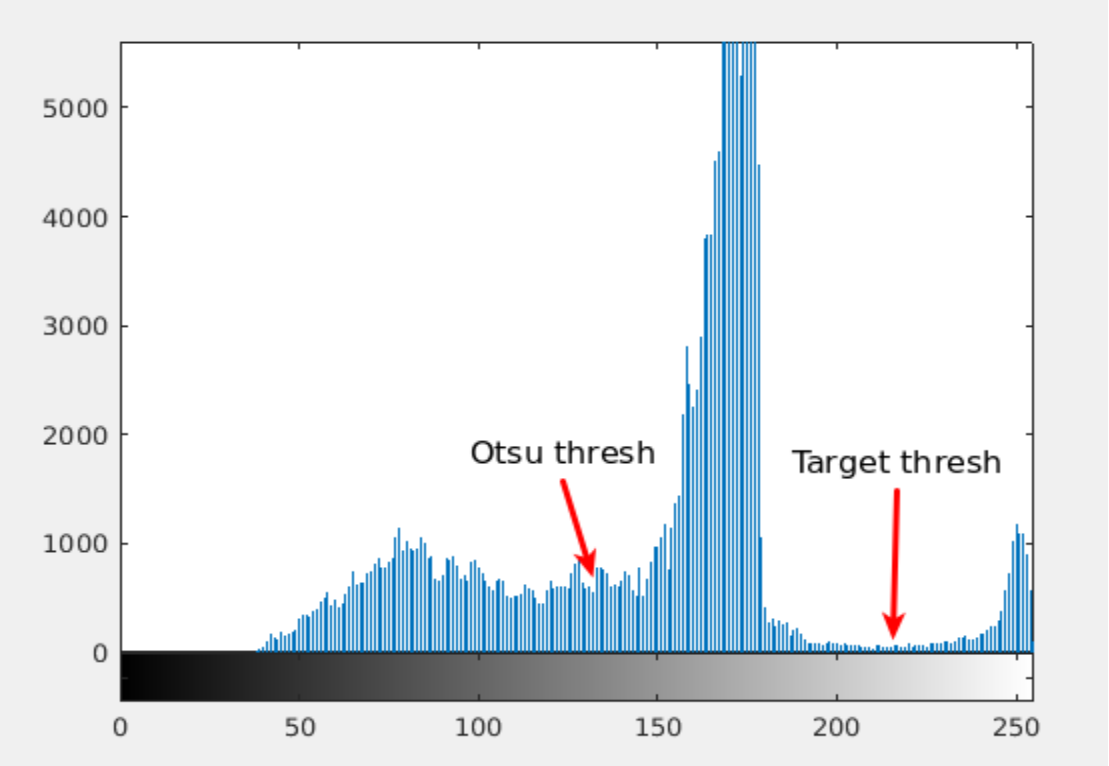

Figure 3-12 : Histogramme d'une image a niveau de gris.[web 7]

3.4.2 Binarisation par la méthode HSV

voici la partie du code depuis l'acquisition de l'image jusqu'a la binarisation.

```
cap.open(0);
if(cap.isOpened()==false){
  ui->txtXYRadius->append("error: webcam not accesed succefully");
   return ;}
cap.read(matOriginale);
```

```
if(matOriginale.empty()==true)
return;
-----------------------------------------------------------------------------------------------------------
cv::cvtColor(matOriginale, matProcessed, COLOR_BGR2HSV);
 cv::inRange(matProcessed, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS,
iHighV),matProcessed);
```

la première partie réalise l'acquisition de l'image sur l'objet 'matOriginale ' .

la deuxième partie :

la fonction 'cvtColor' avec la méthode COLOR_BGR2HSV réalise la conversion d'une image RGB a une image HSV .

'inRange ' permet de faire le seuillage de l'image HSV à l'aide des paramètres minimales et maximales introduits .

L'introduction des paramètres minimales et maximales se fait par l'interface graphique développé avec Qt (figure 3-13).

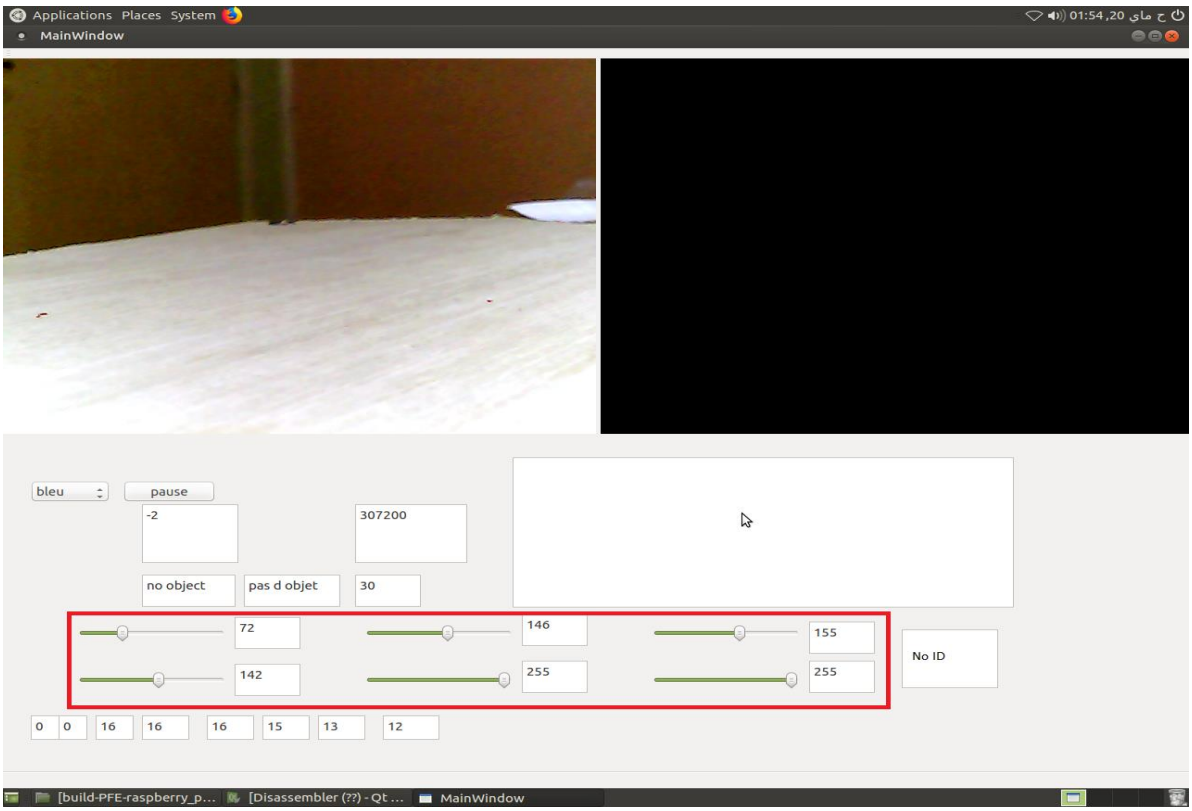

Figure 3-13 : l'interface graphique (rectangle rouge : introduction des paramètres de seuillage).

Cette interface (figure 3-14 et figure 3-15) ne sert pas juste à introduire les valeurs de seuillage , mais elle sert aussi à :

- Introduire la couleur désirée à capter.
- Afficher s'il ya une détection ou pas.
- Déterminer si l'objet est au centre de l'image.
- Donner la distance objet caméra.
- Définir la nature de l'objet.
- Donner l'image originale et binaire.

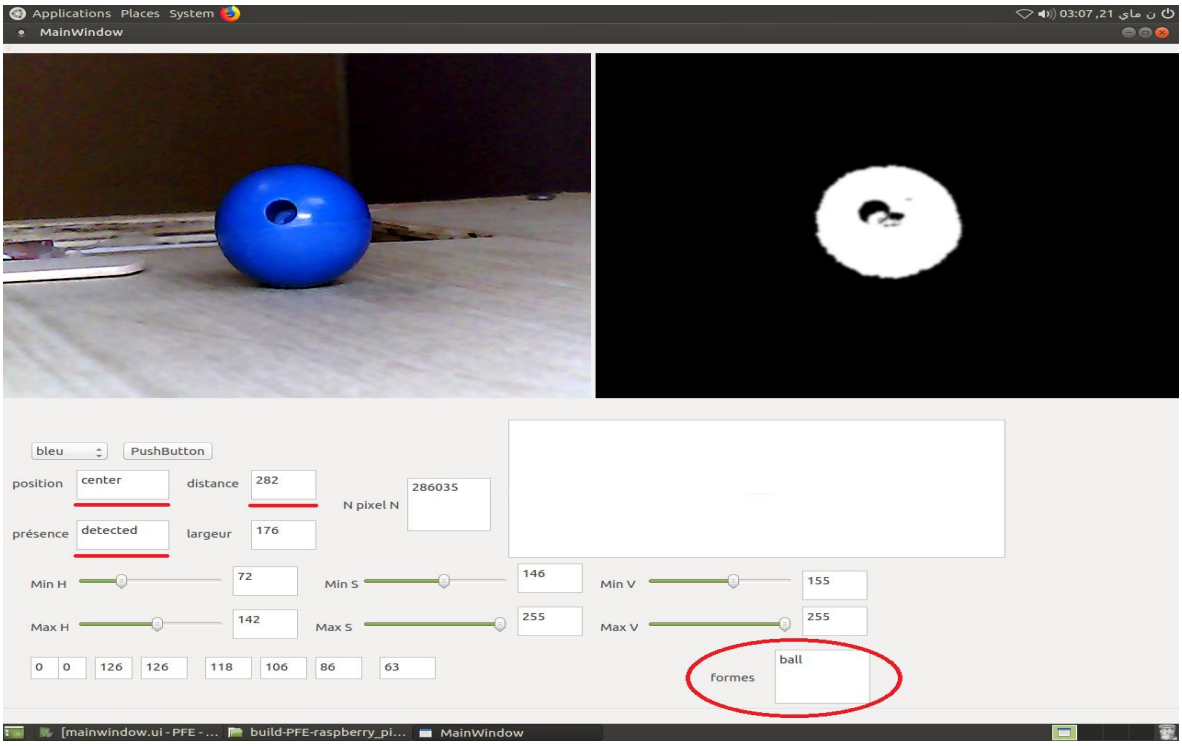

Figure 3-14 : Démonstration des fonctionnalités de l'interface graphique.

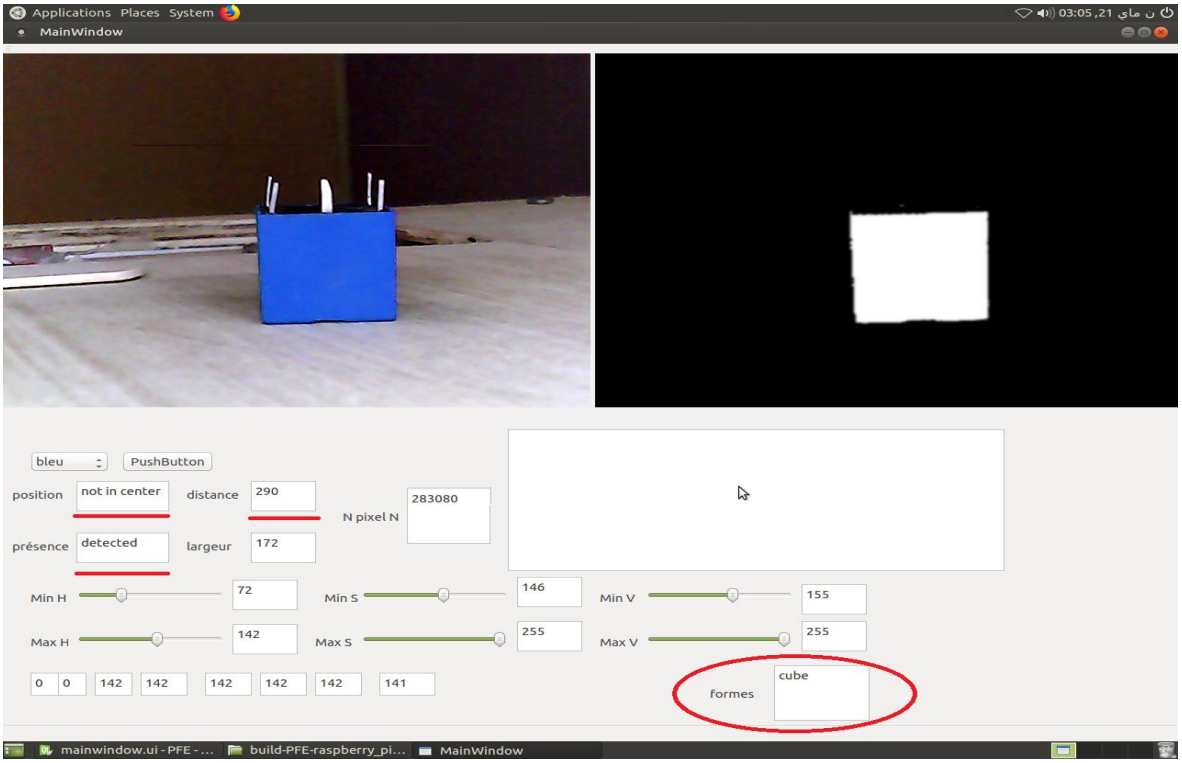

Figure 3-15 : Démonstration des fonctionnalités de l'interface graphique.

3.5 Exploitation de l'image binaire pour la commande

La communication entre le Raspberry et l'Arduino est une communication parallèle asynchrone . Deux algorithmes tournent en permanence pour assurer la performance du système, et cinq étapes sont essentielles dans le traitement.

3.5.1 Test de présence d'un objet

Le système cherche si l'objet est présent ou pas en utilisant cette formule:

$$\sum_{i=0}^{n}\sum_{j=0}^{m} pixels_{noire}(i,j) \leq N$$

avec $N=n*m$ dimension de l'image

La camera effectue un balayage sur tout le champ de travail, tant qu'un objet n'a pas été détecté. Cette fonction est traduite par cet algorithme.

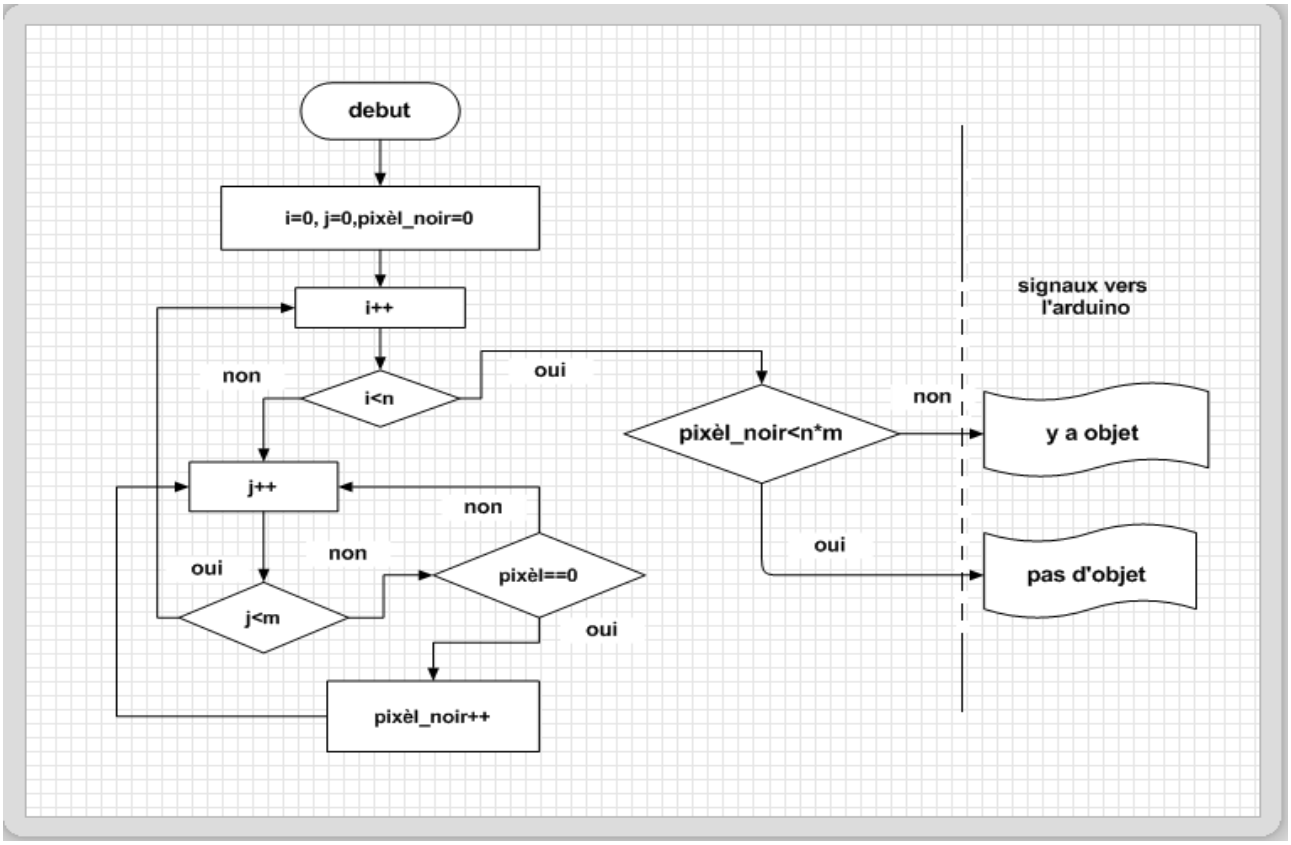

Figure 3-16 : Organigramme (fonction de détection).

Une fois un objet détecté, on passe à la phase suivante qui consiste a calculer la largeur de l'objet puis le centrer dans l'image pour que l'angle pris par le moteur de la camera soit exacte car le nombre de pas des moteurs pas a pas sont enregistrer puis convertie en angle voir la section (3.6.1 initialisation).

3.5.2 Calcul de la largeur

Le modèle mathématique de cette méthode est traduit par cette formule.

$$\sum_{0}^{m} pixel_{blanc} \begin{cases} si\ pixel_{blanc_{precedent}} < pixel_{blanc_{calculée}} => ecraser \\ si\ pixel_{blanc_{precedent}} > pixel_{blanc_{calculée}} => pas\ d'op \\ si\ pixel_{blanc_{precedent}} = pixel_{blanc_{calculée}} => pas\ d'op \end{cases} \} avec\ \ pixel \in img$$

Avec m : dimension verticale de l'image .

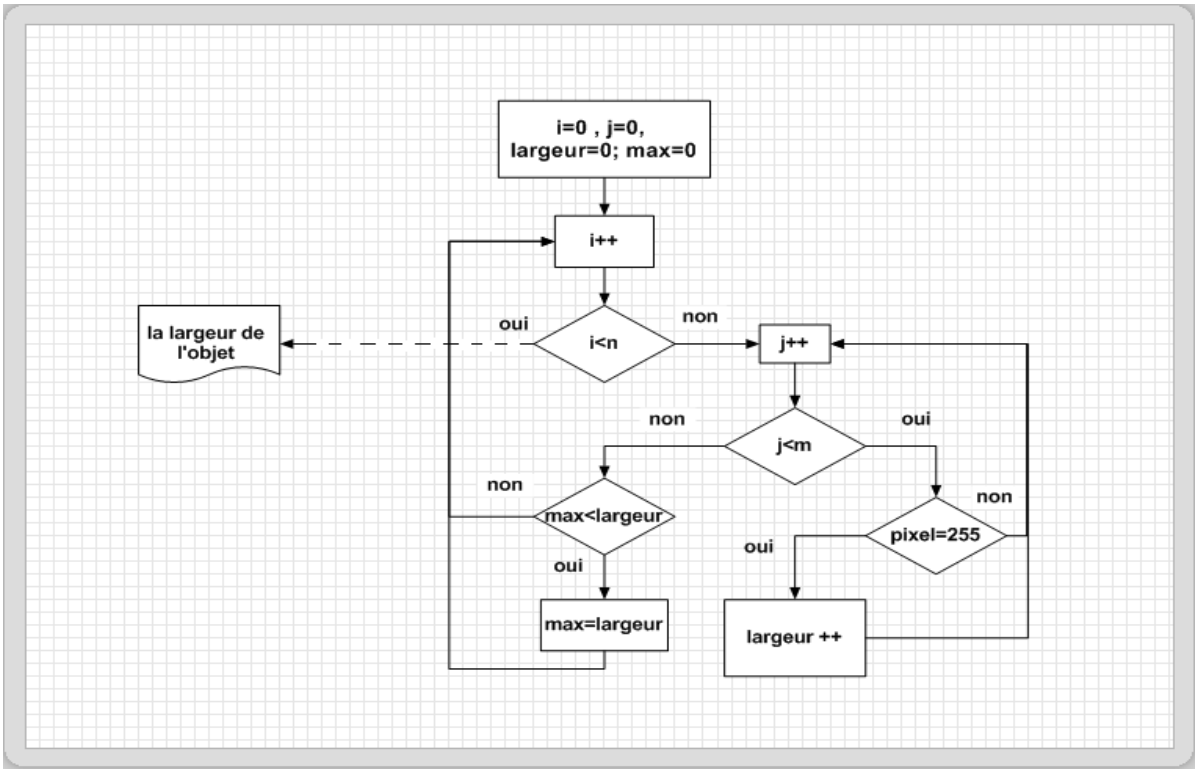

Figure 3-17 : Organigramme de calcul de la largeur en pixel.

La figure 3-17 est l'organigramme de calcul de la largeur maximale de l'objet, l'étape qui reste c'est de centrer l'objet dans l'image.

3.5.3 Centrer l'objet dans l'image

Pour cela, on fait une délimitation de l'image binaire selon la largeur de l'objet (voir figure 3-18) , et on teste les pixels blancs, si elle existe sur le coté gauche on fait tourner le moteur de la camera vers la droite sinon l'inverse et si l'objet est au milieu on n'envoie aucune commande.

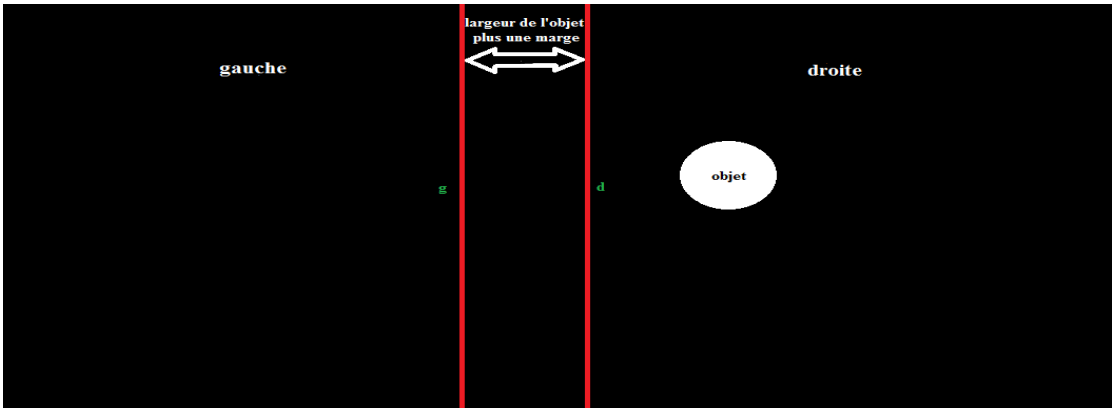

Figure 3-18 : image binaire.

Cette méthode est traduite par l'organigramme suivant .

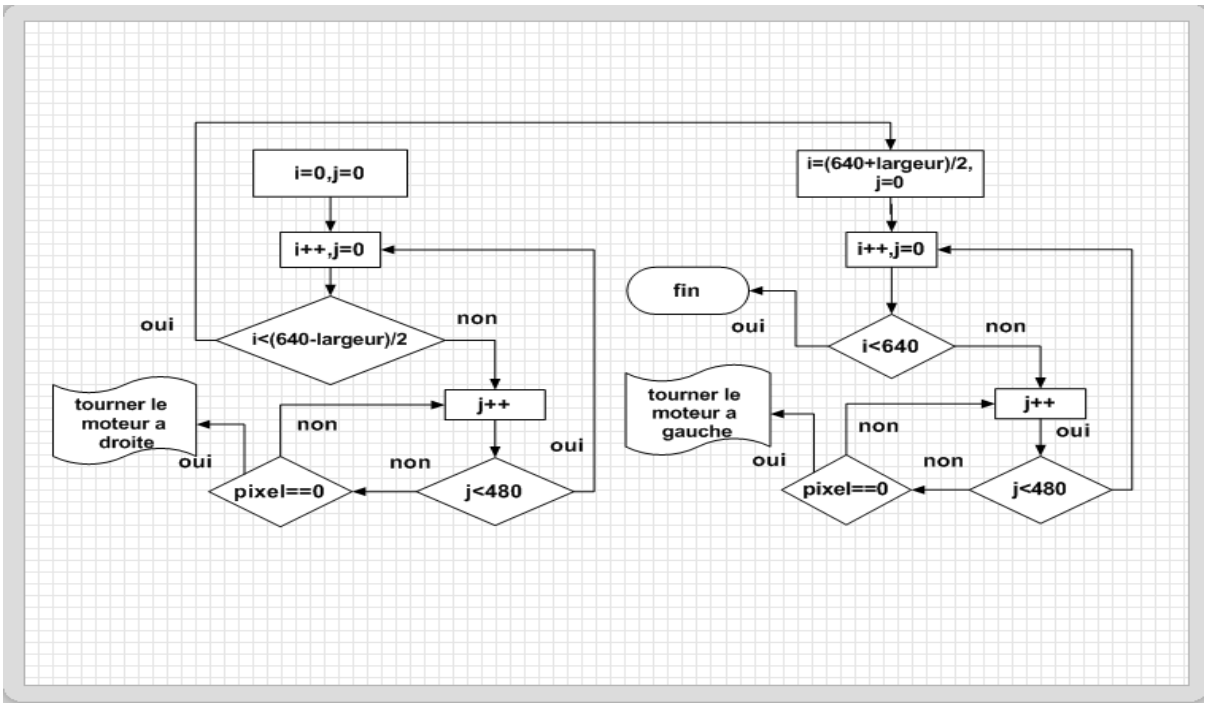

Figure 3-19 : organigramme d'asservissement du moteur de la webcam.

Les deux drapeaux sont des signaux envoyés vers l'Arduino, maintenant il reste à calculer la distance entre l'objet et la webcam. Mais avant on a besoin de la calculer la distance focale

3.5.4 Calcul de la distance focale

Pour calculer la distance objet-camera il existe deux méthodes , soit on utilise deux caméras (stéréoscopie) (voir l'annexe B), Soit avec une seule caméra mais cette méthode

nécessite la connaissance des dimensions de l'objet et la distance focale de la camera. (figure 3-20)

La distance focale est un paramètre spécifique à la camera , elle est soit donnée sur le datasheet du produit soit calculée par la méthode suivante :

Le calcul de la distance focale consiste à prendre plusieurs images d'un objet avec dimensions et distance (objet-camera) connus et à chaque prise de photo on fait varier la distance (objet-camera) et on calcule les dimensions d'objet en pixels puis on applique la formule suivante

distance_focale = (objet camera * largeur_pixel)/. largeur_objet_mm.

Il faux reprendre cette opération plusieurs fois (au moins dix fois) et calculer la moyenne.

$$(\sum_{0}^{n} distance_{focale}) \, /n$$

Avec n : nombre d'itération.

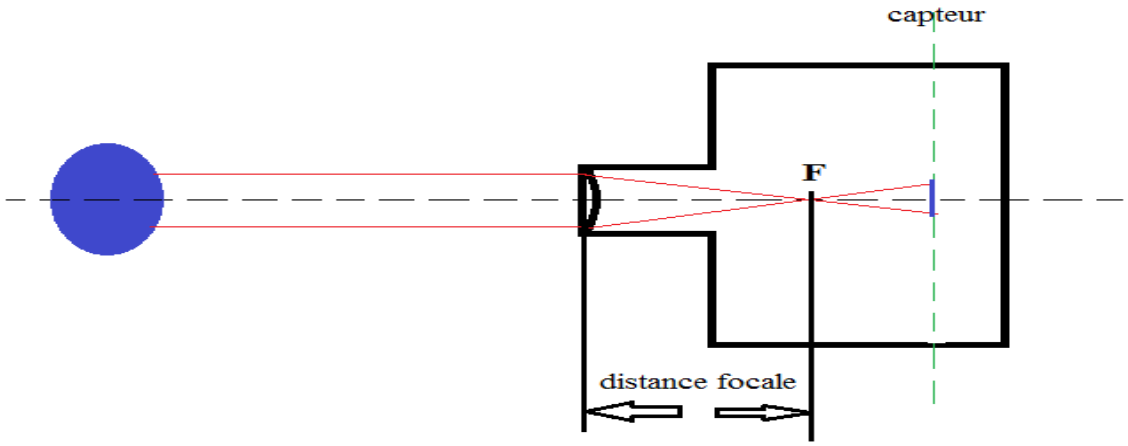

Figure 3-20 : Diagramme explicatif de la distance focale.

La distance :objet camera = (distance_focale * largeur_objet_mm)/largeur_pixel.

Dans le programme la distance est mise à jour à chaque prise d'image , c'est la raison pour laquelle le système ne demande pas de synchronisation de communication .

3.5.5 Conversion et transmission de la distance

Une fois la distance calculée ,il faut la convertir en binaire pour l'envoyer sur un bus de données parallèle de 8 bits.

Soit le LSB =U(0) et MSB=U(7).

$$U(i) = Distance\%2 \ \ pour\ i\ 0:7\ \{pour\ i++ => distance = \frac{distance}{2}\}$$

pour cela voici l'organigramme qui réalise cette fonction:

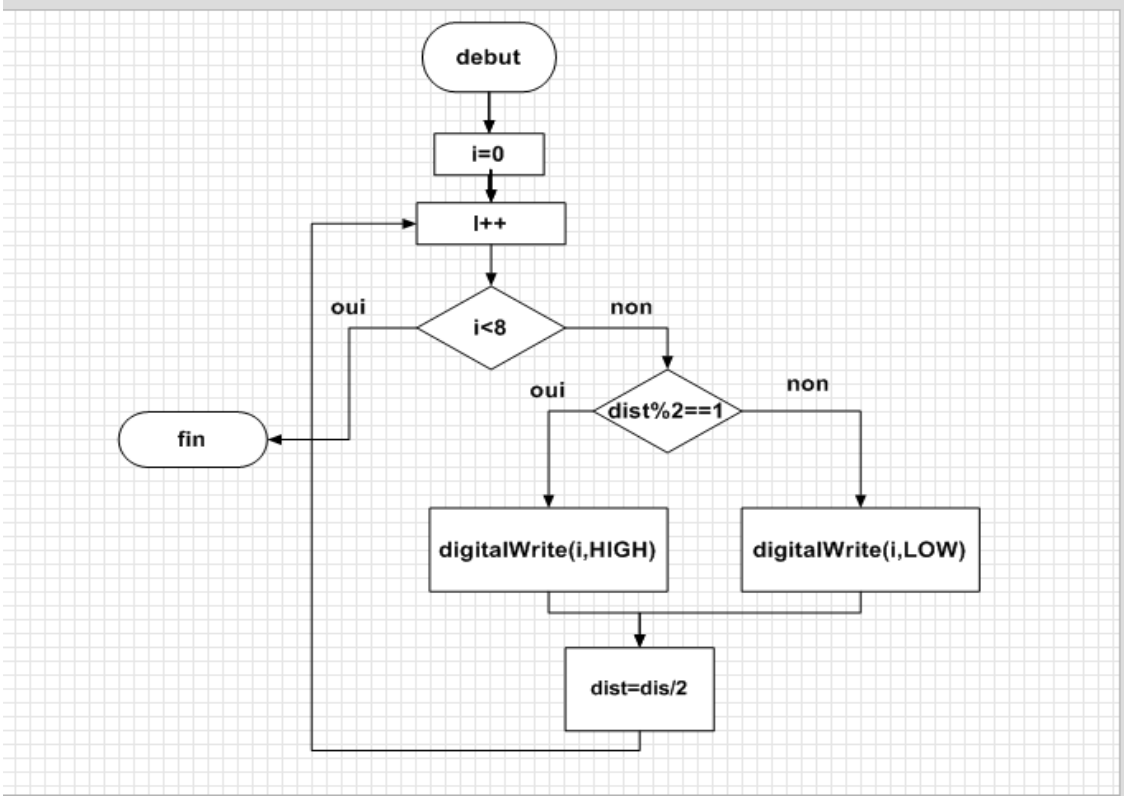

Figure 3-21 : Organigramme de conversion décimale-binaire.

3.5.6 Reconnaissance des formes

La reconnaissance de motifs peut être effectuée au moyen de divers algorithmes d'apprentissage automatique tels [25] :

- un réseau de neurones.
- une analyse statistique.
- l'utilisation de modèles de Markov cachés.
- une recherche d'isomorphisme de graphes ou sous-graphes.

Les formes recherchées peuvent être des formes géométriques, descriptibles par une formule mathématique, telles que :

- cercle ou ellipse ;

- courbes de Bézier, splines ;
- droite.

Mais par raison d'optimisation on a choisi de faire une architecture moins complexe pour résoudre le problème.

Si on remarque la variation de la largeur d'objet lors du calcul, on déduit que si l'objet est un cube alors la largeur ne varie pas trop , en gros elle fait une fonction stable , mais si l'objet est une boule alors automatiquement la largeur est une parabole (croissante puis décroissante) figure 3-22 .

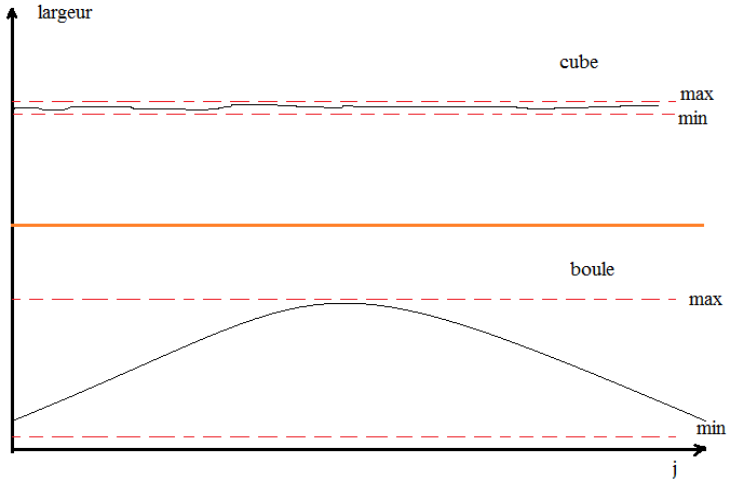

Figure 3-22 : variation de la largeur d'objet selon la forme.

Le but c'est de comparer la variance de la largeur avec l'une des fonctions illustrées dans la figure 3-22 . mais le problème c'est que la dimension en pixel de l'objet est variable selon sa positon , alors on a le choix entre de manière d'implémenter cette fonction :

Soit on dresse une liste dynamique (pour enregistrer la variance de la largeur) pour ensuite la comparer , sinon on discrétise la variance et on prend des échantillons pour les enregistrer dans un tableau standard .

La deuxième méthode a été choisi par raison d'optimisation de calcule et simplicité d'implémentation. La discrétisation a été prise par une précision de huit pixels , voir l'organigramme

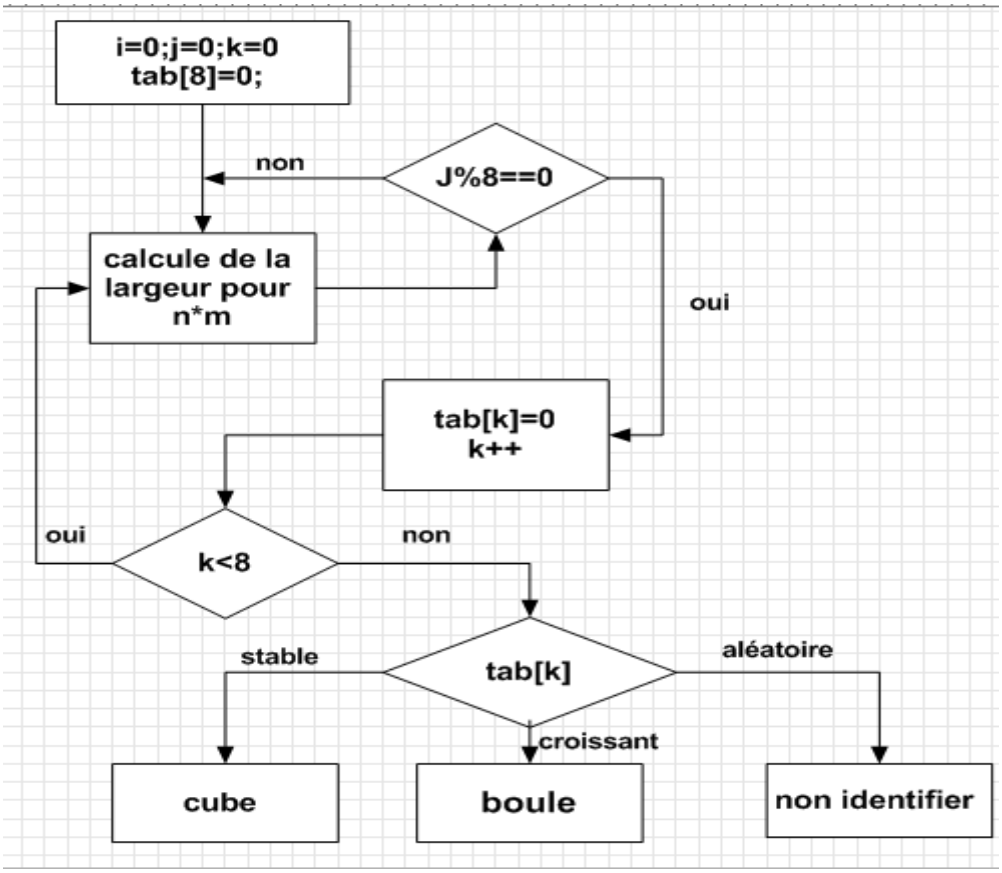

Figure 3-23 : organigramme de reconnaissance des formes.

3.6 Algorithme sur Arduino

Cette étape consiste à lire les données transmis par le Raspberry et de faire un conditionnement du signal pour l'adapter au besoins, voici l'organigramme qui décrit l'algorithme qui se déroule dans l'Arduino .

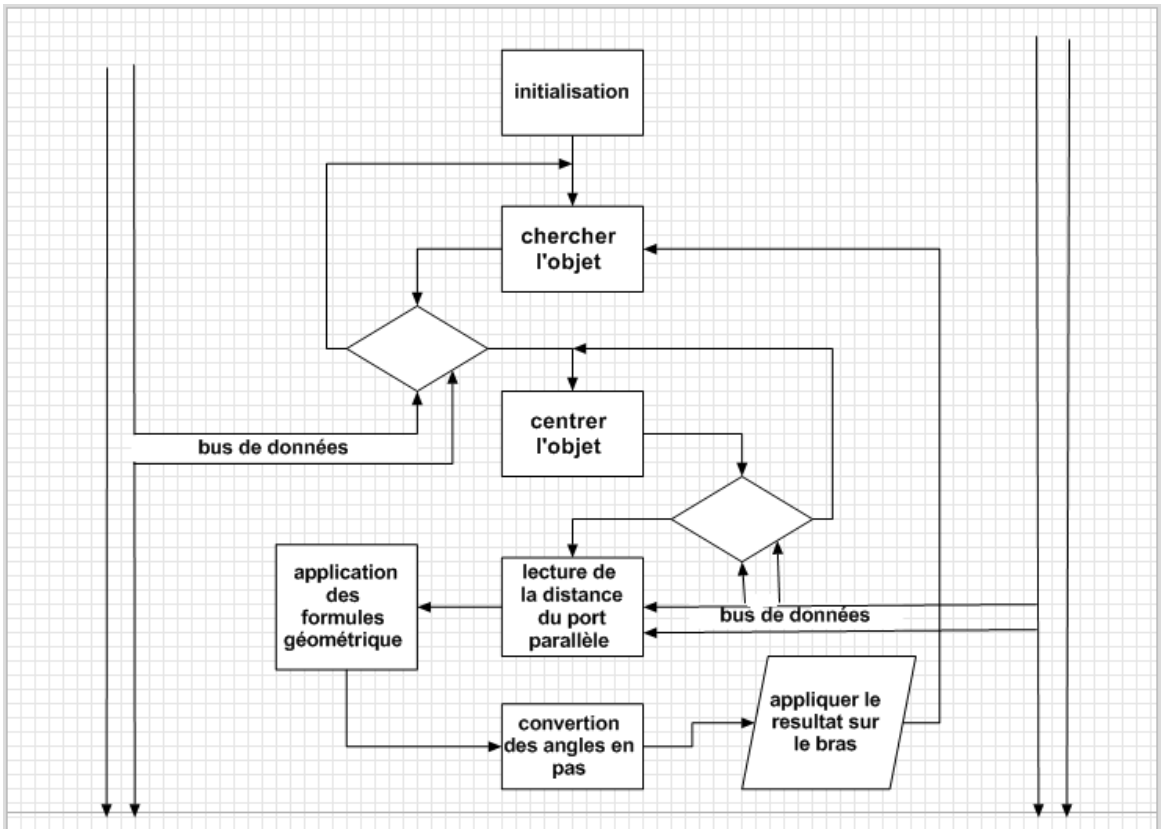

Figure 3-24 : organigramme qui se déroule dans l'Arduino.

Chaque bloc est une fonction en langage Arduino , on va détailler chacun de ces blocs.

3.6.1 Initialisation

Vu que les moteurs pas à pas ne sont pas asservis , il est impératif de les asservir en ajoutant des capteurs de fin de course. La fonction initialisation permet d'envoyer des commandes aux moteurs et de faire le comptage des pas pour connaître leurs positions le long du processus (calibrage), mais avant d'attaquer cette étape il faux connaitre le rapport du couple des moteurs, pour convertir les angles en nombre de pas .

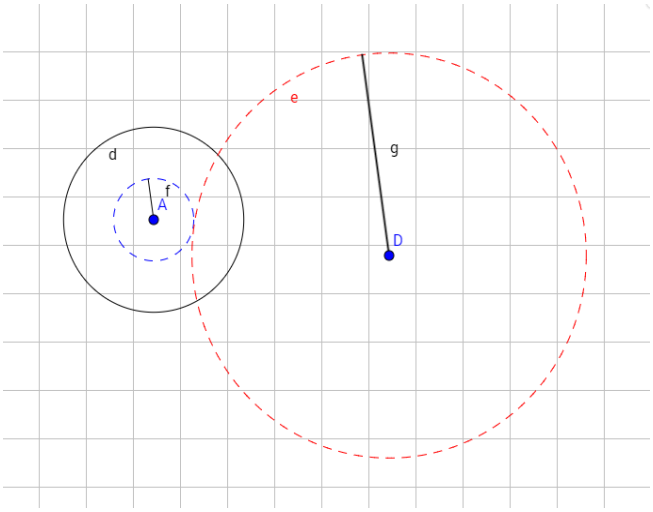

Figure 3-25 : Modèle géométrique du réducteur.

Sois 'A' l'axe du moteur et 'D' l'axe de rotation du bras ou de la camera , et le moteur fait 200 pas pour atteindre 360° .

Pour que le cercle (e) fait un tour entier il faut que le moteur tourne $g * 200/f$

L'organigramme est comme suit :

Avec initSteppC et initSteppB : le nombre de pas pour que la camera fais 360° et pas =1 .

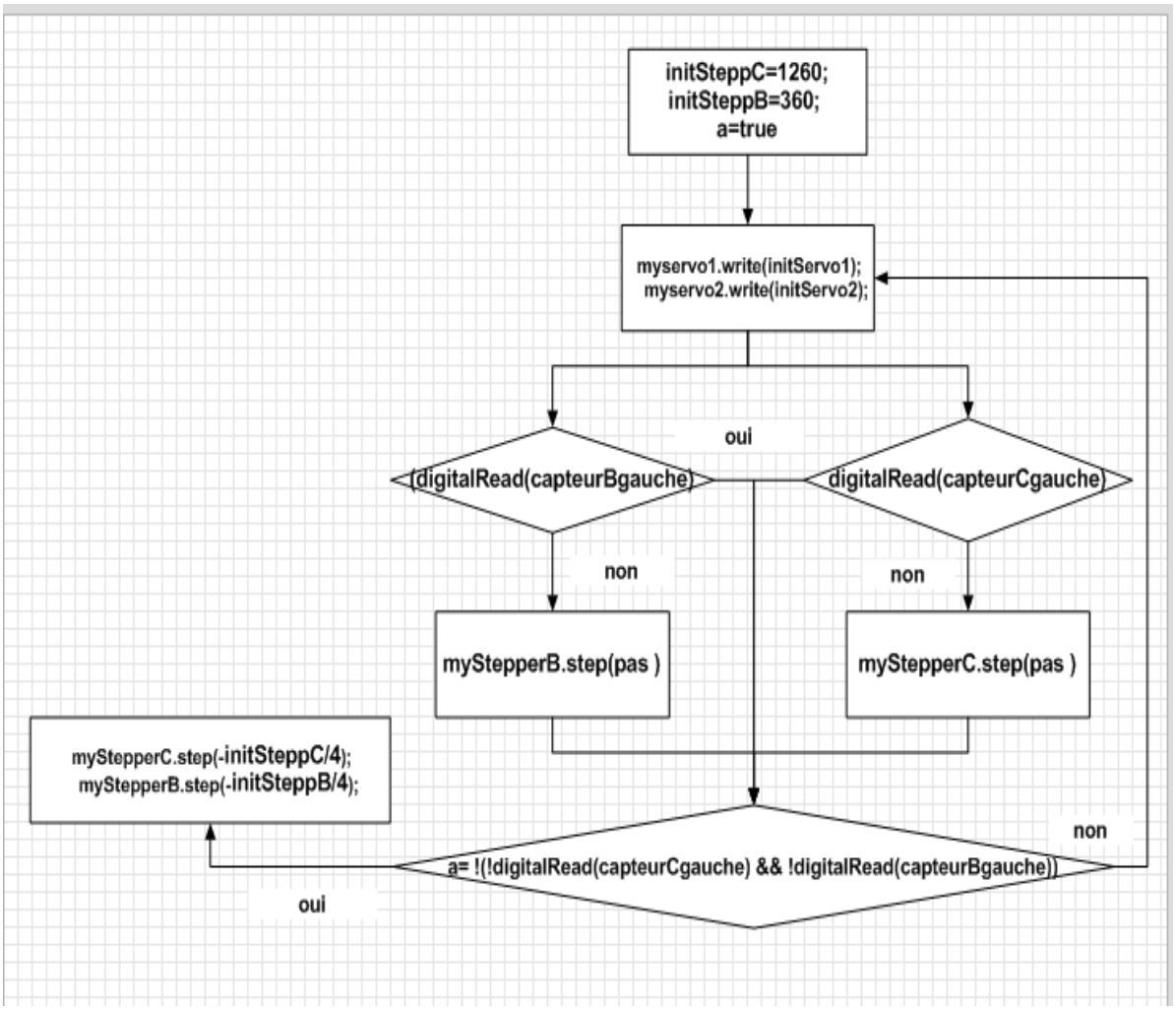

Figure 3-26 : Organigramme de l'initialisation.

Cette fonction consiste à positionner le bras et la camera dans leur état initial.

3.6.2 Présence d'objet

Puisque le champ de vision de la camera ne couvre pas toute la scène ,il est nécessaire de faire un balayage et attendre l'analyse réalisé par la carte Raspberry pour dire s'il y a présence ou pas d'un objet, voici l'algorithme de la fonction de recherche .

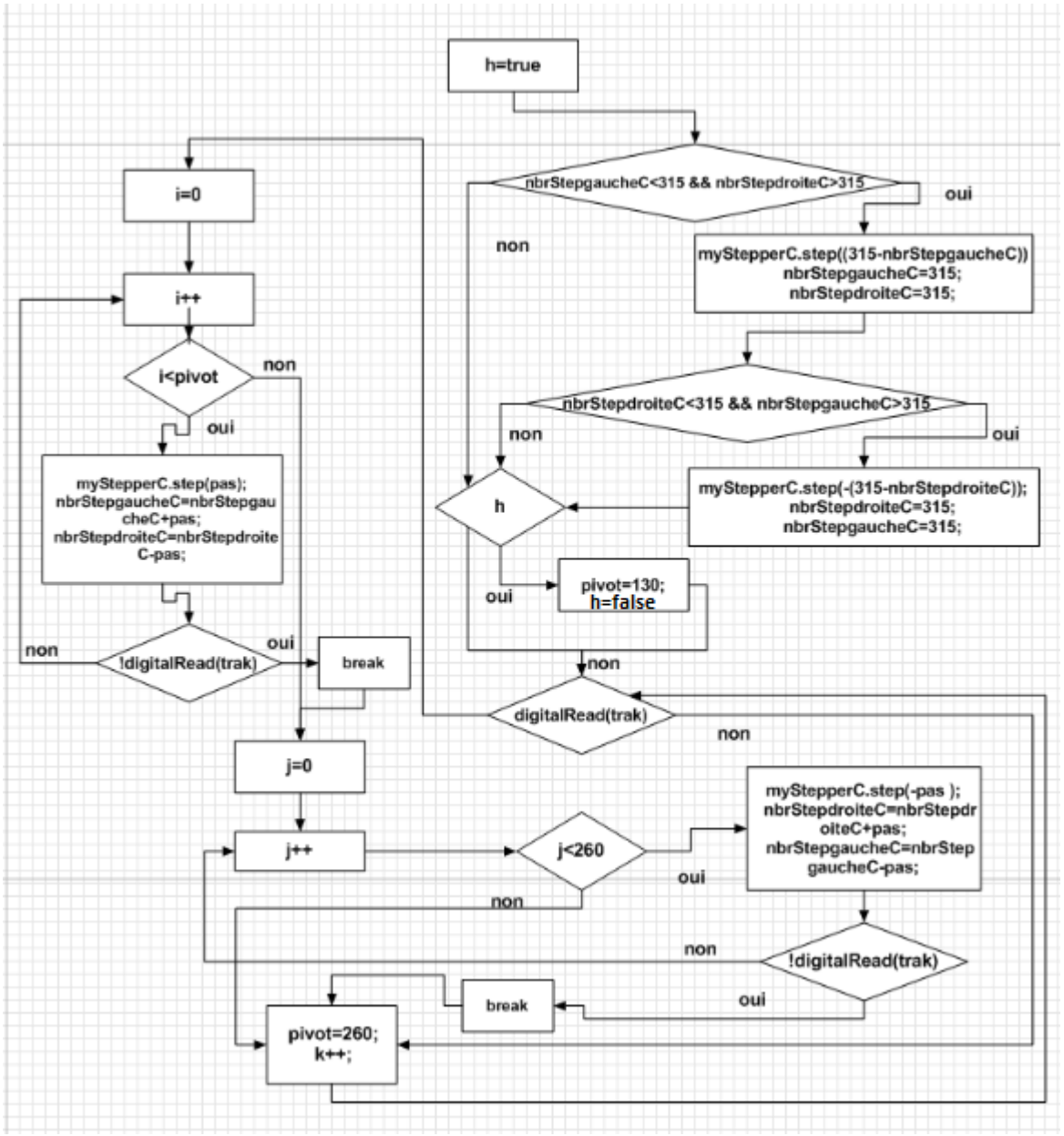

Figure 3-27 : organigramme de recherche.

Une fois l'objet détecté le programme sort de la boucle pour exécuter la fonction prochaine (centrer l'objet dans l'image).

3.6.3 Centrer l'objet dans l'image

Le rôle de cette fonction est d'asservir le moteur de la camera face à l'objet pour que l'angle calculé soit exact .

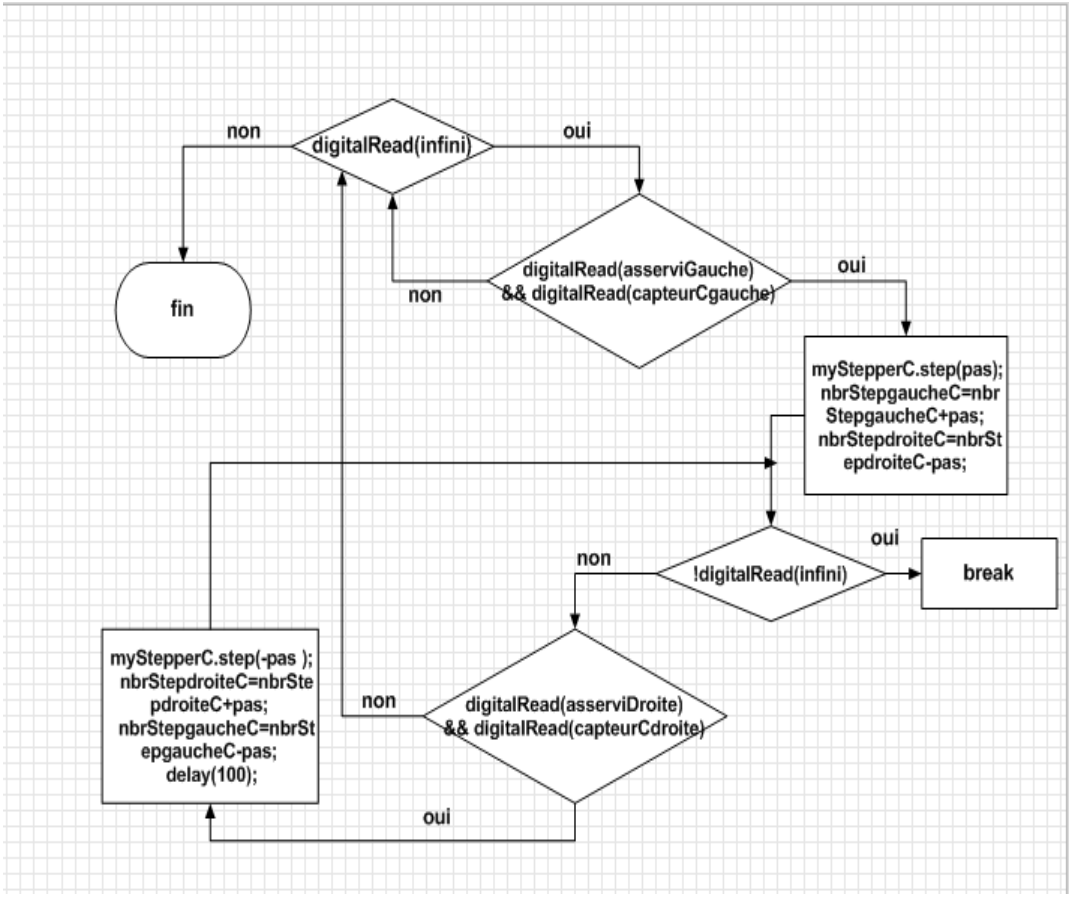

Figure 3-28 : Organigramme de mise en face de la camera.

Maintenant il faut lire la distance du port parallèle et la convertir en décimale.

3.6.4 Lecture et conversion

Comme déjà illustré dans la section (conversion et transmission de la distance) la distance est présente sous forme binaire sur le port , la fonction suivante permet de lire et de convertir cette dernière en décimale voici l'organigramme de cette fonction.

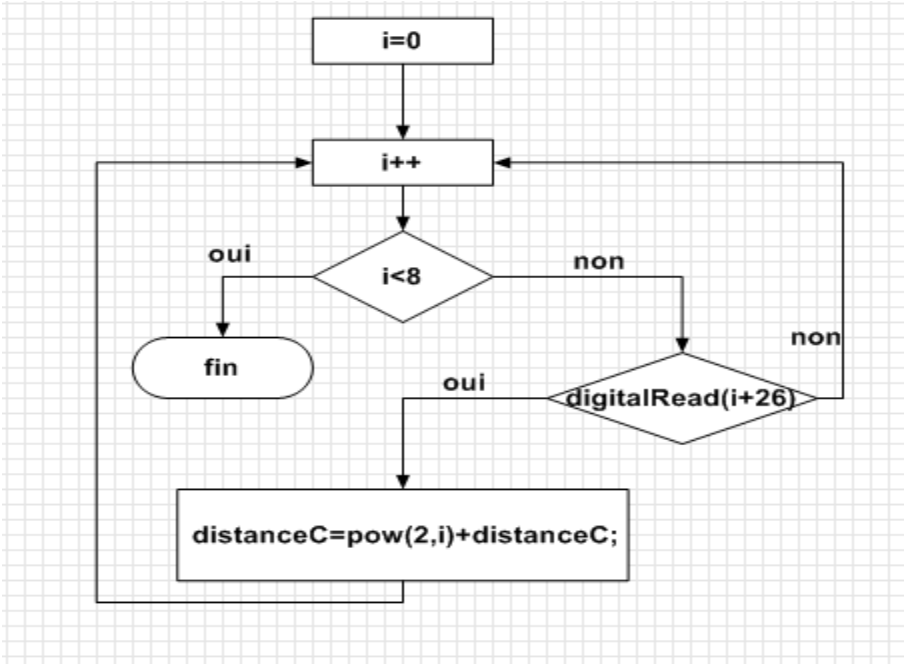

Figure 3-29 : organigramme de lecture et conversion de la distance.

Après l'enregistrement de la distance (camera-objet) , on applique la fonction géométrique pour le calcul de l'angle du bras et la distance (bras-camera).

3.6.5 Application des formules géométriques

Cette fonction exploite l'étude abordée dans la section (calcule de distance et géométrie) . Voici l'organigramme dédié à cette fonction.

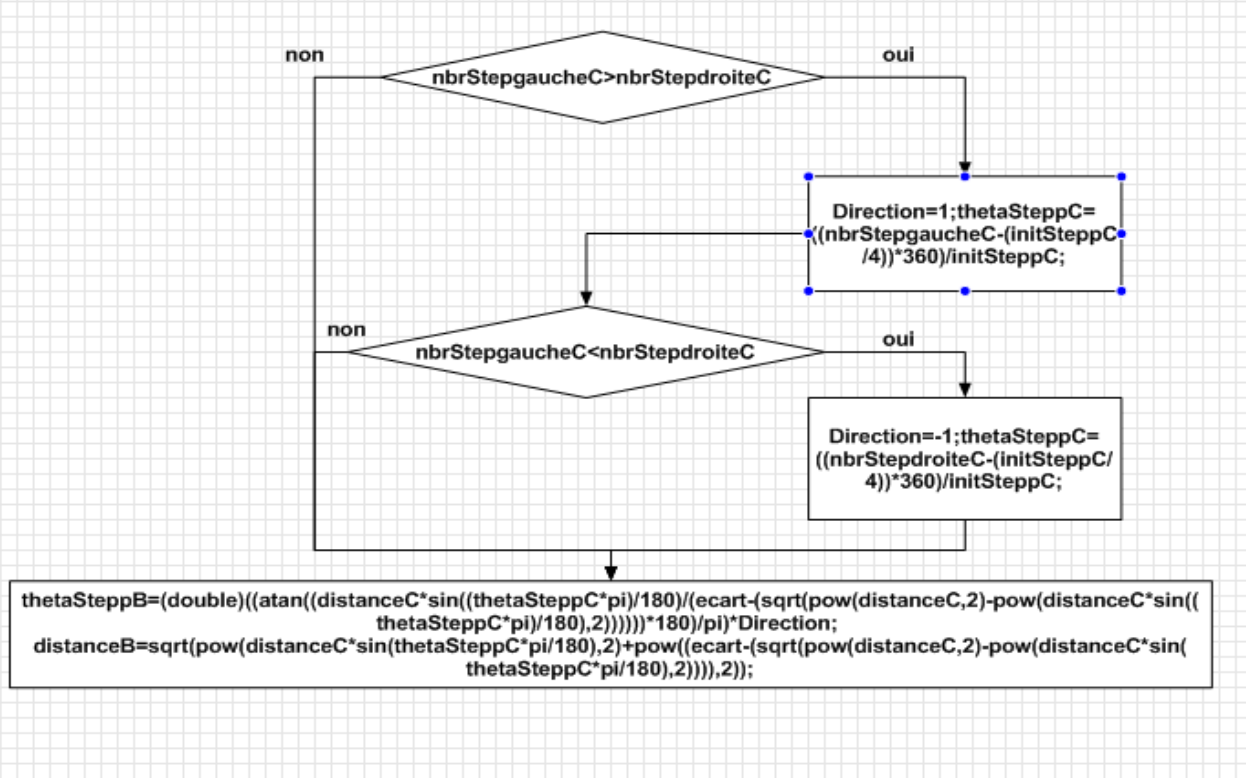

Figure 3-30 : Organigramme de calcul des données pour le bras.

3.6.6 Conversion de la distance en angles

Dans la partie (modèle géométrique du bras) , nous avons vu comment convertir la distance en angle , voici la partie du code qui fait ce travail

thetaServo1=(180*acos((distanceB/2)/x1))/pi.

thetaServo2=180-(2*thetaServo1);

avec pi : le nombre π .

3.7 Photo de réalisation

3.7.1 Conception de la pincette

Cette étape est plus complexe vue qu'elle comprend un mécanisme qui transforme mécaniquement la rotation en un mouvement de translation . Voici le schéma équivalent de la réalisation de la pincette.

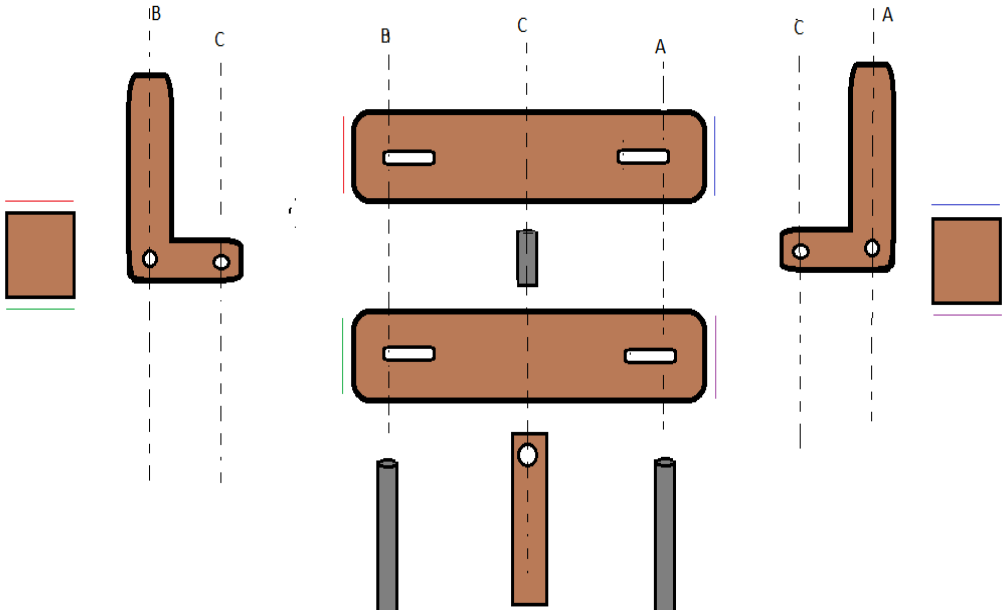

Figure 3-31 : Mécanisme de la pincette.

3.7.2 Réalisation du bras

Le bras doit être solide et possède un mouvement fluide, c'est la raison pour laquelle nous utilisons des roulement à billes dans l'axe de rotation .

Figure 3-32 : axe de rotation du bras.

La force est impérative , car le bras doit supporter sont poids et le poids de l'objet.

- Des servo-moteurs S3003 avec une tension d'opération de 6 volts , capables de fournir un couple de 4,1Kg.cm.

-Un moteur pas a pas équipé d'un motoréducteur et avec une tension de 12v fournit le couple nécessaire pour tourner le bras entier sur son axe .

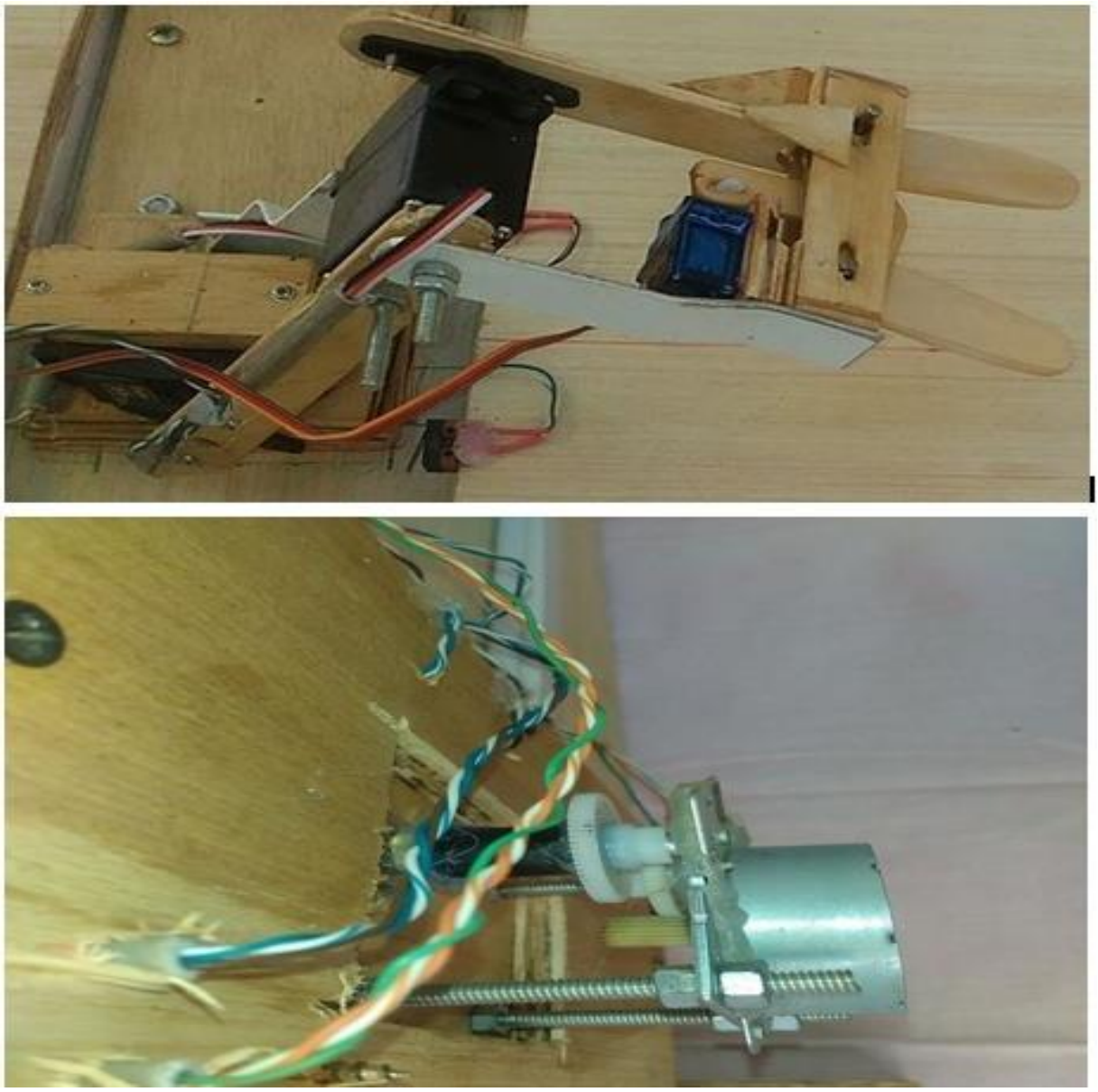

Figure 3-33 : en bas dessous du bras (moteur pas a pas) en haut le bras avec les trois servo-moteurs.

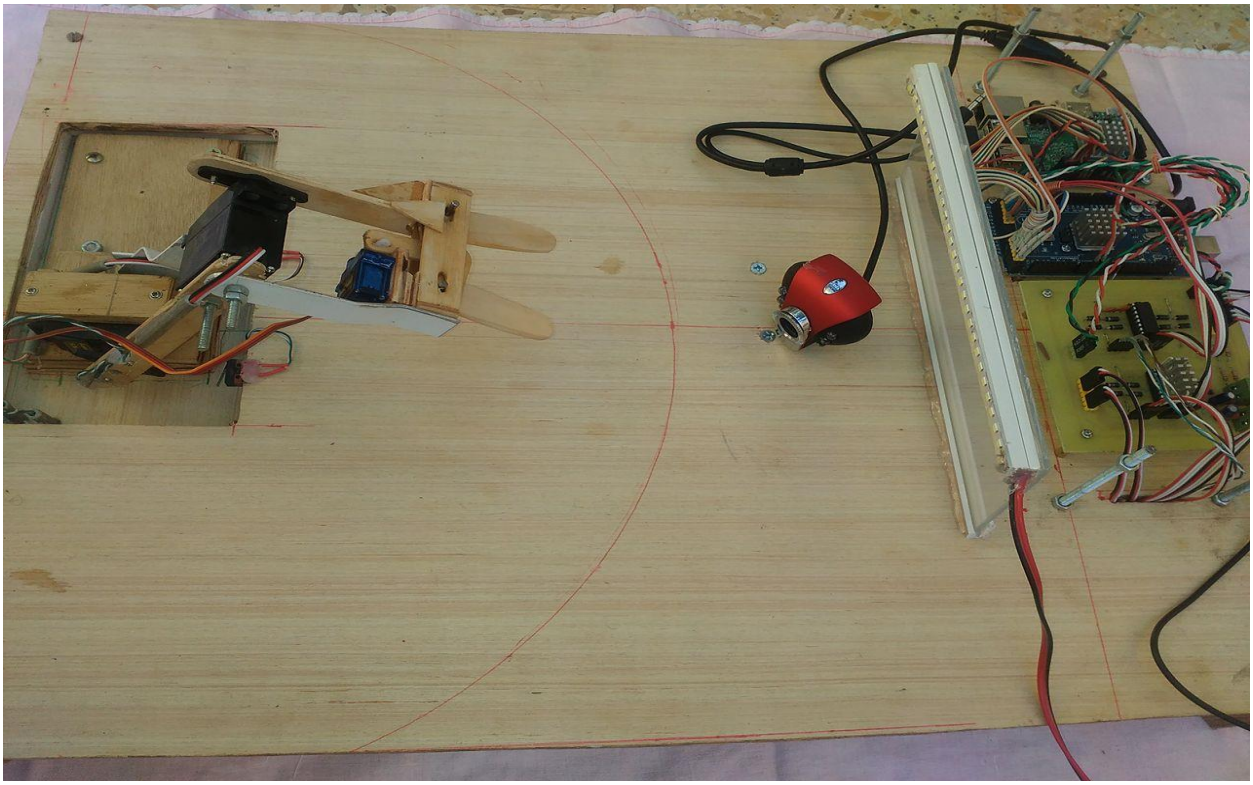

Figure 3-34 : vue de profil prototype.

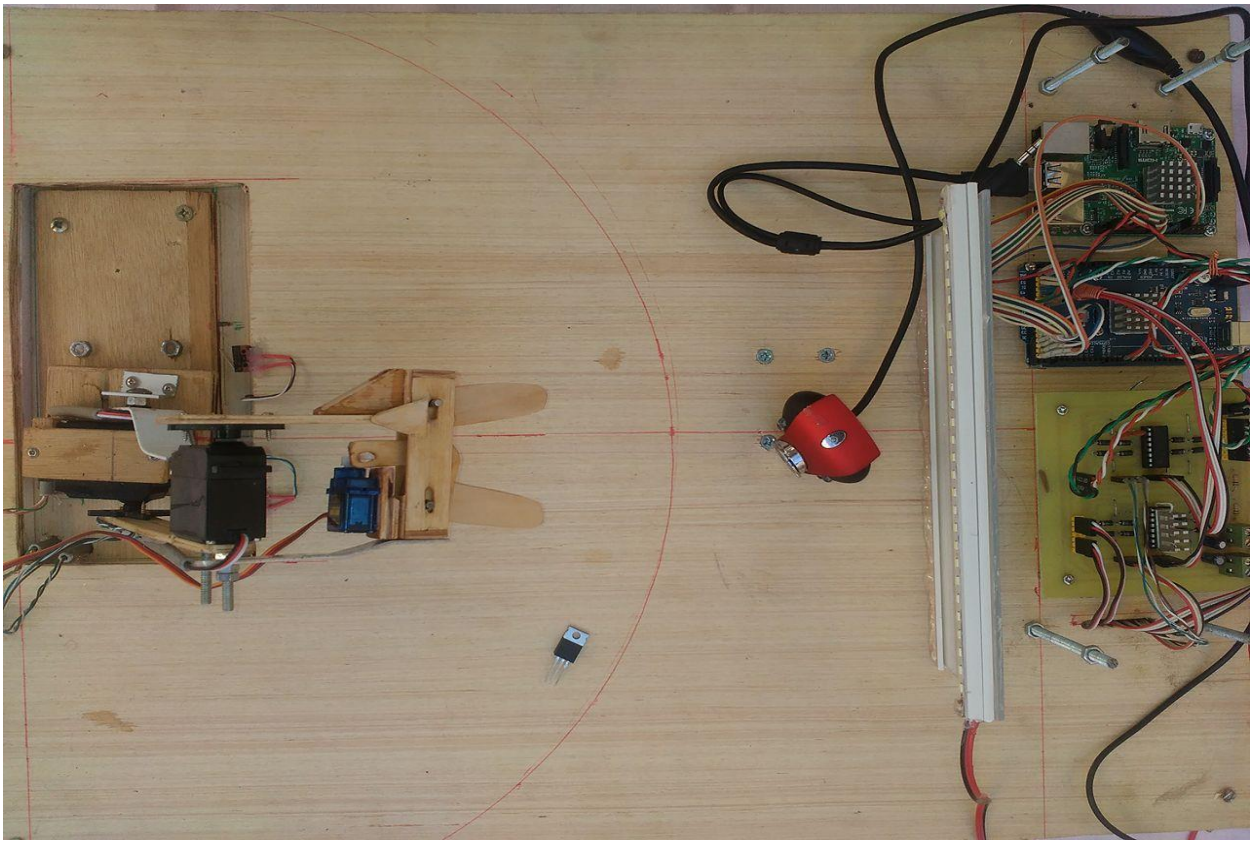

Figure 3-35 : vue de haut du prototype.

Figure 3-36 : vue de bas du prototype.

3.8 Taux de réussite

Un certain nombre de contraintes pratiques font que le système ne soit pas fiable à 100%. En effet l'imperfection mécanique et les problèmes d'imprécisions des algorithmes de traitement d'image, font que la tache devienne délicate.

Nous avons réalisé des tests pour estimer quantitativement le taux de réussite.

Le test se base sur la répétitivité et le changement des endroits d'objet. Pour différents emplacements d'objet E (x,y) sur le plan réel choisi aléatoirement , nous obtiendrons E'(x',y') (position calculer par le système) qui sont proche avec un taux de réussite de 90% .

Les erreurs tendent vers zéro à chaque fois que l'objet est plus proche de la camera car le focus est important et la distorsion de l'image est moins importante vue que c'est une caméra bon marché qu'on utilise. (figure 3-37) .

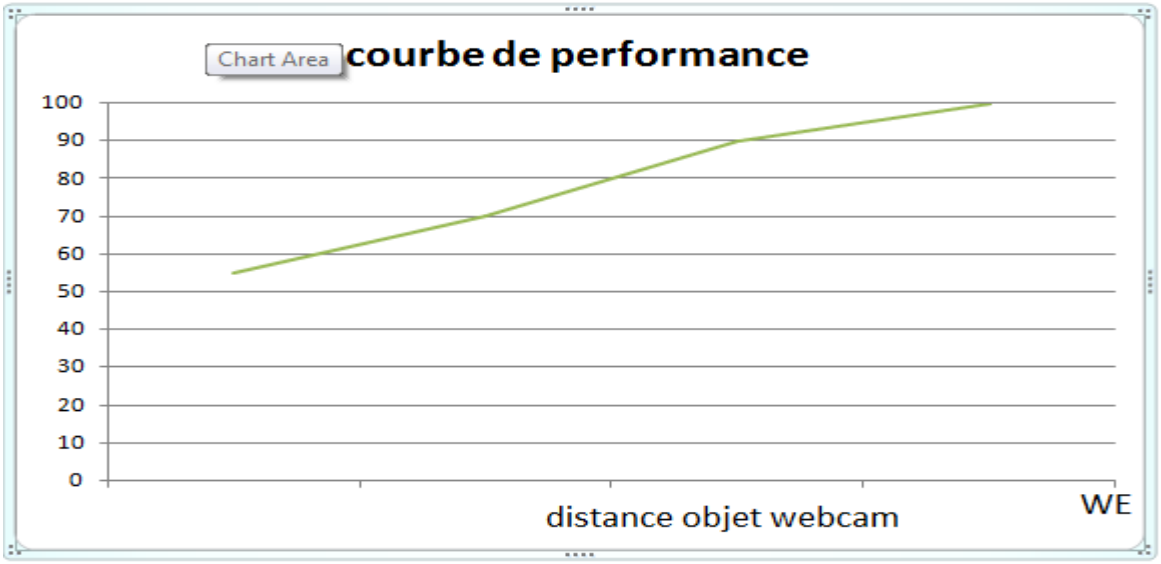

Figure 3-37 : performance du système (Taux de réussite).

3.9 Conclusion

Nous avons présenté dans ce chapitre la conception du bras ainsi que les algorithmes de traitement d'images utilisés, ces algorithmes nous ont permis d'assurer un bon asservissement des moteurs et ont permis une précision de notre bras.

Conclusion générale

La robotique envahie et embellie notre vie avec une attraction de taille à la fois pour les équipementiers, les opérateurs, les entreprises et le grand public. Si les enjeux économiques et sécuritaires justifient largement cette convoitise, il ne faut cependant pas négliger les contraintes techniques à surmonter.

L'objectif de ce travail consiste à concevoir un système de commande par vision artificielle. Pour cela nous avons réalisé un robot avec camera et bras manipulateur en utilisant un Raspberry Pi2, une carte Arduino et une caméra.

Durant le présent projet, nous avons décomposé la conception et la réalisation en trois étapes majeures. La première avait pour but d'étudier la géométrie du bras et de concevoir la carte de puissance pour la commande des moteurs. La seconde consiste à utiliser le traitement d'images via des algorithmes de seuillage pour la détection des objets dans le champ de vision du robot. La dernière étape avait pour but de développer les algorithmes sur Arduino afin d'assurer la précision des mouvements du bras en temps réel couplé avec la camera.

L'élaboration de ce travail nous a permis, d'une part, d'approfondir nos connaissances acquises durant les années de notre formation à l'université, et d'autre part, de préparer notre intégration à la vie professionnelle et de se situer sur le marché de la robotique.

Annexe A : l'algorithme d'Otsu

La méthode d'OTSU est utilisée pour effectuer un seuillage automatique à partir de la forme de l'histogramme de l'image. Cette méthode nécessite donc le calcul préalable de l'histogramme de l'image. L'algorithme suppose alors que l'image à binariser ne contient que deux classes, (Les objets et l'arrière-plan). L' algorithme itératif calcule alors le seuil optimal T qui sépare ces deux classes afin que la variance intra-classe soit minimale et que la variance inter-classe soit maximale [23]

- **Variance intra-classe :**

$$\sigma_w^2 = \omega_1(T) \times \sigma_1^2(T) + \omega_2(T) \times \sigma_2^2(T)$$

Oméga 1 représente la probabilité d'être dans la classe 1

Oméga 2 représente la probabilité d'être dans la classe 2

Sigma 1 représente la variance de la classe 1

Sigma 2 représente la variance de la classe 2

- **Variance inter-classe :**

$$\sigma_y^2 = \sigma^2 - \sigma_w^2$$

Sigma représente la variance de l'image

Sigma w représente la variance intra-classe

- **Calcul de la probabilité de la classe 1 et 2 :**

Pour calculer la probabilisé d'être dans la classe 1 ou 2 en fonction du seuil T, il suffit de sommer les probabilités de chaque niveau de gris.

$$\omega_1(T) = \sum_{k=1}^{T} P(k)$$

$$\omega_2(T) = \sum_{k=T+1}^{256} P(k)$$

- **Calcul de la probabilité de chaque niveau de gris**

Calcule d'histogramme

L'histogramme est un graphique représentant la répartition des valeurs de niveau de gris dans une image. Pour calculer l'histogramme, il faut donc parcourir l'image dans sa totalité et compter le nombre de pixels qu'il y a pour chaque niveau de gris.

$$Hist(k) = \sum_{i=1}^{N} \sum_{j=1}^{M} (Image(i,j) == k)$$

Calcul de la probabilité de chaque niveau de gris :

La probabilité de chaque niveau de gris est calculée en divisant le nombre de pixels présent pour chaque niveau de gris par le nombre total de pixels dans l'image.

$$P(k) = \frac{Hist(k)}{Nombre\ total\ de\ pixels\ dans\ l'image}$$

Annexe B : Calcule de distance par la stéréoscopie

Stereo imaging, also referred to as binocular imaging, is a method of image capture wherein the same field of view is recorded by 2 camera units that are offset slightly from one another as shown in figure 2

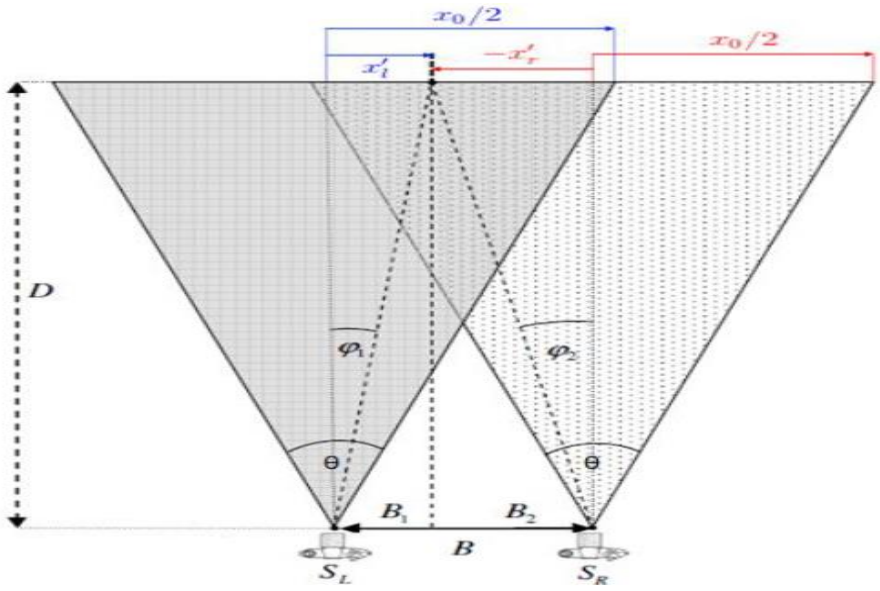

Figure 2 : géométrie de la stéréoscopie[web 8]

The offset distance between the cameras in this configuration, also referred to as the stereo baseline distance (B), causes an individual object to appear at different coordinates within a pair of images taken by both cameras simultaneously. If these cameras lie on the same plane and are aligned in parallel as shown, then this position difference, also referred to as pixel disparity (d), exists only in the lateral dimension of such image sets. Provided this condition is met, the variables Z and d are related by the trigonometric relationship [26]

$$D = \frac{Bx_0}{2d}\cot(\theta/2)$$

Bibliographie

[1]: BALI Chaher eddine, A. H. (2011/2012). *RÉALISATION D'UN ROBOT MOBILE AVEC.* Université Mohamed Khider Biskra, Faculté des Sciences et de la Technologie, Biskra.

[2]: Mattéo. (2014, 08 31). *Refrobot*. Consulté le 02 27, 2018, sur la référence de la robotique: https://web.archive.org/web/20141021105724/http://refrobot.org:80/introduction-a-la-vision-par-ordinateur/

[3]: Oldrik Ceugnart, Rodolphe Wronski2011/2012*Projet Bibliographique Robotique et vision*université de lille1lille,france

[4] : *RASPBERRY PI FOUNDATION*. (s.d.). Consulté le 02 27, 2018, sur raspberry.org: www.raspberry.org

[5] : *Raspberry Pi France*. (s.d.). Consulté le 02 27, 2018, sur 01 Net: http://www.raspberrypi-france.fr/raspberry-pi-windows-10-iot/

[6]: *raspberry pi france*. (s.d.). Consulté le 02 27, 2018, sur dernierement sur raspberry pi france: http://www.raspberrypi-france.fr/comment-installer-android-pi/

[7]: Wong, George. *Build your own prototype Raspberry Pi minicomputer*

http://www.ubergizmo.com/2011/10/build-raspberry-pi-minicomputer/

[8] Moorhead, Joanna. *Raspberry Pi device will 'reboot computing in schools'. The guardian*

[9] eskymon, o. (s.d.). *présentation de l'arduino*. Consulté le 02 28, 2018, sur zestedesavoire: https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-

[10]: *question fréquente*. (s.d.). Consulté le 02 27, 2018, sur arduino.cc: https://www.arduino.cc/en/Main/DebuterFAQ

[11] *Bootloader*. (s.d.). Consulté le 02 27, 2018, sur arduino.cc: https://www.arduino.cc/en/Hacking/Bootloader

[12] Oxer, Jonathan. *Installing the USB driver file for Windows.* Consulté le 02 17,2018 ,sur freetronic .*https://www.freetronics.com.au/pages/installing-the-usb-driver-file-for-windows#.WpWTb7ziYlw*

[13] Das, A. (s.d.). *11 Reasons Why Linux Is Better Than Windows*. Consulté le 02 29, 2018, sur It's FOSS: https://itsfoss.com/linux-better-than-windows/

[14] *Noyau Linux*. (s.d.). Consulté le 02 28, 2018, sur wikipidia: https://fr.wikipedia.org/wiki/Noyau_Linux

[15] *le noyau coeur du système d'exploitation*. (s.d.). Consulté le 02 28, 2018, sur Ubuntu-fr: https://doc.ubuntu-fr.org/kernel

[16] Blaess, C. (2014, 03 06). *Compilation native de modules kernel sur Raspberry Pi*. Consulté le 02 28, 2018, sur Ingénierie et formations sur les systèmes libres: https://www.blaess.fr/christophe/2014/03/06/compilation-native-de-modules-kernel-sur-raspberry-pi/

[17]*Choisir son système d'exploitation*. (s.d.). Consulté le 02 28, 2018, sur Developpez.com: http://alexandre-laurent.developpez.com/articles/hardware/raspberry-pi/choisir-systeme/

[18] : drogon. (2013, 05 14). *Gordons Projects*. Consulté le 04 18, 2018, sur WiringPi: https://projects.drogon.net/raspberry-pi/wiringpi/

[19] *opencv, 02 27 , 2018 . https://opencv.org/*

[20] *cmake*. (s.d.). Consulté le 03 10, 2018, sur cmake: https://cmake.org/

[21] *stack overflow*. (s.d.). Consulté le 04 07, 2018, sur https://stackoverflow.com/questions/9094941/compiling-opencv-in-c?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

[22] drogon. (s.d.). *GPIO Interface library for the Raspberry Pi*. Consulté le 16 04, 2018, sur wiringpi: http://wiringpi.com/

[23] christopher, l. (s.d.). *binarisation de l'image*. Récupéré sur methode d'otsu: https://sites.google.com/site/lizantchristopher/services/binarisation-1

[24] McCauly, M. (s.d.). *bcm2835 by Mike McCauley*. Consulté le 04 16, 2018, sur IBEX: http://www.raspberry-projects.com/pi/programming-in-c/io-pins/bcm2835-by-mike-mccauley

[25] christopher, l. (s.d.). *binarisation de l'image*. Récupéré sur methode d'otsu: https://sites.google.com/site/lizantchristopher/services/binarisation-1

[26]: Andrew Fullenkamp, C. N. (2016). *Use of Stereoscopic Imaging for Distance Determination.* Indiana University-.

[Web 1] : https://www.robotshop.com/eu/fr/plateforme-developpement-mobile-jaguar-v6-dr-robot-bras.html

[Web 2] : https://www.youtube.com/watch?v=ve8gPtXPax0

[web 3]https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html

[Web 4] : https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

[Web 5] : http://www.redohm.fr/2014/12/arduino/

[Web 6] : https://wisotop.de/hsv-und-hsl-farbmodell.php

[Web 7] : https://stackoverflow.com/questions/42679248/how-can-i-get-good-binary-image-using-otsu-method-for-this-image

[Web 8] : http://www.ni.com/white-paper/14103/en/