الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

FACULTE DE TECHNOLOGIE

Département du Génie Électrique



Projet de fin d'études

Présenté pour l'obtention du
Diplôme De Master en Génie
Electriques

Spécialité : Automatique Des Systèmes

THEME

Asservissement visuel d'un robot mobile avec évitement d'obstacles

Proposé et dirigé
par,

Dr. R. KARA

Dr. D. HASSANI

Préparé et présenté
par,

M. ZAIBAK Serhane

M. BOUAISSA Mouadh

Année Universitaire 2016-2017

Dédicaces

Je dédie ce travail :

A mes chers parents pour leur amour, leurs Encouragements, leur patience et leurs précieux conseils

À mes chers frères et ma chère sœur

A toute ma famille

À tous mes amis

Mouadh

Dédicace

Je dédie ce modeste travail à mes chers parents qui m'ont beaucoup encouragé et soutenu pendant tout leur vie pour voir ce jour-là et pour leurs précieux conseils ainsi que leurs recommandations.

A mes très chers frères et mes chères sœurs.

A toute ma famille, grande et petite sans exception.

A tous mes amis

Que ce mémoire soit l'expression de mon profond respect et mon grand amour.

Serhane

Remerciement

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la Patience d'accomplir ce modeste travail.

En second lieu, nous tenons à remercier notre encadreur Dr. KARA Reda et Dr. D. HASSANI pour leurs précieux conseils et leurs aides durant toute la période du travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

ملخص:

يهدف هذا المشروع إلى تحقيق المراقبة أو التقييد البصري للروبوت المتحرك مع تخطي العقبات، من أجل ذلك قمنا بتطوير البنية الهيكلية للأجهزة بربطها مع النظام البصري بيكسي بالإضافة إلى لوحة آردوينو و راسبيري باي، من جهة أخرى استطعنا تحقيق التحكم في الروبوت المتحرك عن طريق البيانات البصرية المستخرجة و أجهزة الاستشعار بالموجات الفوق صوتية.

الكلمات المفتاحية: تقييد بصري؛ تخطي العقبات؛ الروبوتات المتحركة؛ راسبيري باي؛ آردوينو؛ بيكسي

Résumé :

Ce projet consiste à réaliser un asservissement visuel d'un robot mobile avec évitement d'obstacle. Pour ce faire, n o u s avons développé une architecture matérielle embarquée en combinant un systeme de vision Pixy, une carte Raspberry à une carte Arduino Par ailleurs, nous avons mis en œuvre un contrôleur capable d'asservir les mouvements d'un robot mobile à partir de données visuelles et les capteurs à ultrasons.

Mots clés : asservissement visuel ; évitement d'obstacle ; robotiques mobiles ; Pixy ; Raspberry Pi ; Arduino ;

Abstract.

This project consists of performing a visual servoing of a mobile robot with obstacle avoidance. In order to do this, we have developed an embedded hardware architecture by combining a Pixy vision system, a Raspberry card with an Arduino card. In addition, we have implemented a controller capable of slaving the movements of a mobile robot from Visual sensors and ultrasonic sensors.

Keywords : Visual Servoing ; Obstacle avoidance ; Mobile robotics ; Raspberry Pi ; Arduino ; Pixy

Sommaire

Listes des figures

Listes des tableaux

Notation et acronymes

Introduction générale 1

Chapitre1 : Généralités sur les robots mobiles

1. Introduction 4
2. Aperçu historique 4
3. Les bases mobiles 5
   3.1 Holonomie 5
   3.2 Les plates-formes différentielles 6
   3.3 Les plates-formes omnidirectionnelles 7
   3.4 Les plates-formes non holonomes 8
   3.5 Les plates-formes à pattes 9
4. Les robots à roues 10
   4.1. Robot unicycle 10
   4.2. Robot tricycle 11
   4.3. Robot voiture 11
   4.4. Robot omnidirectionnel 12
5. La robotique mobile tout-terrain 13
6. Capteurs utilisés dans les systèmes de robotique mobile 17
   6.1. Les Capteurs extéroceptifs 18
      a) Les capteurs télémétriques 18
      b) Les GPS 18
      c) Les caméras 19
7. Les stratégies de navigation 20
   7.1. Approche d'un objet 20

7.2. Guidage....20
7.3. Action associée à un lieu....21
7.4. Navigation topologique....22
7.5. Navigation métrique....23
8. Les architectures de contrôle....25
8.1. Contrôleurs Hiérarchiques....26
8.2. Contrôleurs réactifs....27
8.3. Contrôleurs hybrides....28
9. Conclusion....29

Chapitre2 : Etude des systèmes embarqués utilisées

1. Introduction....31
2. La carte Arduino....31
2.1. Définition....31
2.2. Les Applications....32
2.3. Les atouts majeurs d'un système à carte Arduino....32
2.4. Arduino Uno....33
2.5. Structure de la carte Arduino Uno....34
2.6. Spécifications techniques....36
2.7. Le Logiciel de gestion de la carte Arduino....36
2.8. Comparaison Arduino Uno vs Arduino Due....39
3. La carte Arduino Due....40
3.1. Présentation de la carte Arduino Due....40
3.2. Spécifications techniques....41
3.3. Avantages de ARM Core....41
3.4. Alimentation....42
3.5. Mémoire....43
3.6. Entrée et sortie....43
3.7. Communication....46
3.8. Programmation....47
3.9. Protection contre les surintensités USB....48
3.10. Caractéristiques physiques et compatibilité des boucliers....48
3.11. Synthèse....49
4. La carte Raspberry Pi....49
4.1. La carte Raspberry Pi 2 model B....50
4.2. Les composants standards de Raspberry....51

4.3. Les GPIO 54
4.4. Mise en marche d'un Raspberry Pi 2 55
4.5. Le réseau 56
4.6. Préparation de la carte SD 56
4.7. Système d'exploitation 57
4.8. Programmation 57
4.9. Synthèse 58
5. La caméra Pixy 59
5.1. Aperçu 59
5.2. Bref historique 60
5.3. Simplicité de mise en œuvre du système de vision 60
5.4. Support du contrôleur 61
5.5. La détection d'objets 61
5.6. Définition simple de l'objet à suivre 62
5.7. L'application PixyMon 62
5.8. Spécifications techniques 63
6. Conclusion 64

Chapitre3 : Asservissement Visuel

1. Introduction 66
2. Historique de l'asservissement visuel : 66
3. Les systèmes de vision 68
3.1. La Vision Monoculaire 68
3.2. La Stéréovision 68
3.3. La Vision Omnidirectionnelle 70
4. Architecture des systèmes asservis par la vision : vers une taxonomie 71
5. Classification des systèmes d'asservissements visuels 74
5.1. Selon la position de capteur visuel 74
5.2. Selon la nature des grandeurs asservies 74
5.3. Selon le type de commande 76
6. Conclusion 78

Chapitre4 : Description de la plateforme robot mobile

1. Introduction 80
2. Robot mobile Arduino 82
2.1. Description 82

2.2. Modélisation 83
3. Le schéma d'asservissement visuel du robot mobile Arduino 84
4. L'architecture matérielle de gestion et de commande de la plate-forme robot mobile 85
4.1. L'extension (Shield) de commande des moteurs 85
a) Principales caractéristiques : 86
b) Les moteurs et le registre à décalage 86
4.2. Les capteurs ultrasons 89
a) Caractéristiques des ondes ultrasonores 90
5. HC-SR04 - Module de détection aux ultrasons 91
5.1. Caractéristiques 91
5.2. Fonctionnement de ce capteur 93
5.3. Distance de l'objet 93
6. Le système de vision Pixy 95
6.1. La détection d'objet objet avec Pixy 95
6.2. Objets à signatures multiples 97
6.3. Balance des blancs 98
7. L'utilisation de l'application PixyMon 98
7.1. Signature tuning 100
8. La carte Raspberry Pi: le cerveau 103
9. Les organigrammes des différents contrôleurs 104
9.1 L'organigramme du contrôleur visuel 104
9.2. L'organigramme du Contrôleur en charge de l'évitement d'obstacles 105
9.3. L'organigramme du contrôleur hiérarchique d'asservissement visuel avec évitement d'obstacle 106
10. Description du programme d'asservissement visuel 106
10.1. Résultat 111
11. Conclusion 115
Conclusions et Perspectives 116

Listes des figures

Figure 1.1 : La tortue de Grey Walter.

Figure 1.2 : Exemple de plate-forme différentielle. Pioneer 2 DX de la société Mobile Robots Urban Robot de la société iRobot.

Figure 1.3 : Exemple de plate-forme omnidirectionnelle à roues suédoises.

Figure 1.4 : Exemple de plate-forme non holonome de type Ackerman.

Figure 1.5: Exemples de robots à pattes. Hexapode de AAI Canada, Aibo de Sony, Nao de Aldebarran Robotics.
Figure 1.6 : Robot de type unicycle.
Figure 1.7 : Robot de type tricycle.
Figure 1.8 : Robot de type voiture.
Figure 1.9 : Robot de type omnidirectionnel Comparaison des différents types.
Figure 1.10 : Robots d'exploration planétaires.
Figure 1.11 : Chronologie de la robotique mobile.
Figure 1.12 : Quelques exemples de robots mobiles tout-terrain.
Figure 1.13 : Quelques exemples d'applications militaires pour les robots mobiles.
Figure 1.14 : Exemples de robots mobiles rapides.
Figure 1.15 : Action associée à un lieu.
Figure 1.16 : Navigation topologique.
Figure 1.17 : Navigation métrique.
Figure 1.18 : Illustration des architectures classiques des contrôleurs pour les robots mobiles Hiérarchique (A), Réactive (B) et Hybride (C).
Figure 2.1 : Structure de la carte Arduino Uno.
Figure 2.2 : logiciel Arduino.
Figure 2.3 : Présentation des boutons.
Figure 2.4 : la carte Arduino Due.
Figure 2.5 : Les composants standards d'un Raspberry.
Figure 2.6 : Pixy CMUcam5.
Figure 2.7 : structure de la Cam Pixy CMUcam5.
Figure 3.1 : capteur stéréovision.
Figure 3.2: Principe de la stéréovision.
Figure 3.3: Champ de vue des systèmes de vision (représenté en gris). (a) Vision panoramique - (b) Vision omnidirectionnelle.
Figure 3.4 : Asservissement visuel avec approche en position de type « dynamic look and move ».
Figure 3.5 : Asservissement visuel avec approche fondée sur l'image de type « dynamic look and move ».
Figure 3.6 : Asservissement visuel avec approche en position.
Figure 3.7 : Asservissement visuel avec approche fondée sur l'image.
Figure 3.8 : Principe de l'asservissement visuel 3D.

Figure 3.9 : Principe de l'asservissement visuel 2D.
Figure 3.10 : Asservissement visuel 2D1/2.
Figure 3.11 : Asservissement visuel cinématique.
Figure 3.12 : Asservissement visuel dynamique.
Figure 4.1 : Schéma bloc du projet.
Figure 4.2 : Robot mobile de type tricycle.
Figure 4.3 : Un robot mobile de type tricycle et son CIR.
Figure 4.4 : structure d'asservissement visuel utilisé.
Figure 4.5 : Adafruit Motor Shield.
Figure 4.6 : fonctionnement de L293D.
Figure 4.7: montage L293D avec Shield.
Figure 4.8 : IC 74HCT595N.
Figure 4.9 : fonctionnement de IC 74HCT595N.
Figure 4.10 : HC-SR04 ultrasons.
Figure 4.11 : Angel de détection.
Figure 4.12 : chronogramme de HC-SR04 ultrasons.
Figure 4.13 : le temps de propagation.
Figure 4.14 : les objets difficiles à détecter.
Figure 4.15 : des objets avec des teintes bonnes et distinctes.
Figure 4.16 : des objets avec teintes non distinctes.
Figure 4.17 : la majeure partie de l'objet (bon) couvre par une grille.
Figure 4.18 : maintenez la signature de l'objet que vous souhaitez enseigner.
Figure 4. 19 : sélectionner la région que vous souhaitez que Pixy utilise pour apprendre l'objet.
Figure 4.20 : Pixy "apprendra" l'objet.
Figure 4.21 : Pixy Paramètres.
Figure 4.22 : ajuster l'inclusivité de la signature 1.
Figure 4.23: la détection intermittente ou clairsemé.
Figure 4.24 : valeur de curseur qui procure une bonne détection.
Figure 4.25 : la carte Raspberry Pi utilisé.
Figure 4.26 : Voir ce que voit Pixy.
Figure 4.27 : test de détermination de largeur L, hauteur H et de X cas 1.
Figure 4.28 : test de détermination de largeur L, hauteur H et de X, cas 2.
Figure 4.29 : test de détermination de largeur L, hauteur H et de X, cas 3.

Figure 4.30 : test de détermination de largeur L, hauteur H et de X, cas 4.
Figure 4.31 : les angles des détections dans le robot arduino mobile.
Figure 4.32 : simulation de Motors shield.
Figure 4.33 : graphe de la tension d'alimentation du moteur on fonction de PWM.
Figure 4.34 : graphe de courant d'excitation du moteur on fonction de PWM.
Figure 4.35 : graphe de la vitesse du véhicule on fonction de PWM.
Figure 4.36 : l'évolution du suivi de cible avec le contrôleur visuel.

Listes des tableaux

Tableau 1.1 : Les avantages et des inconvénients des différents types de robots à roues.
Tableau 1.2 : Caractéristiques de quelques robots mobiles rapides.
Tableau 2.1 : caractéristiques de la carte arduino uno.
Tableau 2.2 : comparaison des caractéristiques de la carte uno vs due.
Tableau 2.3 : caractéristiques de la carte arduino due.
Tableau4.1 : brochage d'IC 74HCT595N.
Tableau 4.2 : Spécifications et limites.
Tableau 4.3 : Synoptique du système d'alimentation en pont des moteurs du robot.
Tableau 4.4 : Résultat de la simulation.

Notation

$\boldsymbol{s}$: Grandeur ou Primitive visuelle asservie par la boucle de vision.
$\boldsymbol{P}$: Primitive visuelle 3D (dans notre cas Pose de la caméra).
$\boldsymbol{p}$: Primitive visuelle 2D (dans notre cas un point sur l'image).
$\boldsymbol{vc}$: Vitesse linéaire instantanée de la caméra.
$\boldsymbol{wc}$: Vitesse angulaire instantanée de la caméra.
$\boldsymbol{Vc}$: Torseur cinématique de la caméra.
$\boldsymbol{\tau}$: Vecteur des forces généralisées appliqués aux articulations du robot.
$\boldsymbol{e}$: Fonction de tâche.
$\boldsymbol{Ls}$: Matrice d'interaction relative à la primitive visuelle.
$\boldsymbol{L\theta u}$: Matrice d'interaction relative à la rotation.
$\boldsymbol{\lambda}$: Matrice de gain de contrôle de la boucle de commande visuelle.
$\boldsymbol{t}$: Vecteur de translation.

$\boldsymbol{\theta}$: Angle de rotation.

$\boldsymbol{u}$: Vecteur de rotation.

$\boldsymbol{Rc}$: Repère actuel de la caméra.

$\boldsymbol{Ro}$: Repère de l'objet.

$\boldsymbol{T}$: Matrice de transformation homogène.

$\boldsymbol{Z}$: Profondeur des points attachés à l'objet dans le repère de la caméra.

C : vitesse de son.

T : température (° C).

H : humidité (g/m^3).

$\boldsymbol{w}$**1:** Vitesse angulaire de roue 1.

$\boldsymbol{w}$**2:** Vitesse angulaire de roue 2.

r : rayent de roue.

v : Vitesse de véhicule.

$\boldsymbol{w}$: Vitesse angulaire de véhicule.

l : distance entre les roues.

d : distance.

S : la surface d'objet.

L : largeur.

H : hauteur.

X : valeur de référence**.**

Ɛ : l'erreur.

Kp : gain proportionnelle.

V : vitesse de référence.

Liste des acronymes

CCD: Charge Coupled Device.

GPS: Global Positioning System.

Mini-PC : mini personal computer.

LED: light-emitting diode.

USB: Universal Serial Bus.

IDE : Integrated Development Environment.

PWM : Pulse Width modulation (Modulation de la largeur des pics).

ICSP : In-Circuit Serial Programming ou (Programmation Série Dans le circuit).

SRAM : Static Random Access Memory (mémoire vive statique).

GND : Ground (la masse).

TTL : Transistor-Transistor Logic.

SPI : Interface Série Périphérique.

TW : Two Wire Interface ou (Interface "2 fils").

UART : Universal Asynchronous Receiver Transmitter ou (émetteur-récepteur asynchrone universel).

I2C: inter-integrated circuit.

FPGA: field-programmable gate array.

HDMI: High Definition Multimedia Interface.

WIFI: Wireless Internet for Frequent Interface.

CPU: Central processing unit.

RAM: Random Access Memory (mémoire vive).

LCD: liquid crystal display.

GPIO: General Purpose Input/ Output.

MDI: Medium dependent interface.

LX: Réception data.

TX: émission data.

HTML: Hyper Text Markup Language.

VIN: Voltage Input.

CSI: Camera Serial Interface.

DSI: Desktop Serial Interface.

FFC: Flexible Flat Cable.

LAN: Local Area Network.

JTAG: Joint Test Action Group.

ADC: Analog to Digital Converter.

DAC: Digital to Analog Converter.

DMA*:* Direct Access Memory.

ROM: Read-Only Memory.

SDA: Serial Data Line.

SCL: Serial Clock Line.

MCU: Multipoint Control Unit.

PCB: Printed Circuit Board.

HD: High Definition.

SD: Secure Digital.

SDHC: Secure Digital High-Capacity.

RGB: Red/Green/Blue.

USB OTG: Universal Serial Bus On-The-Go.

AREF : Analogique Référence.

TWI : Two Wire Interface.

AC: Alternating Current.

DC: Direct Current.

SOC: System On a Chip.

3D : signifie trois dimensions. Ces dimensions permettent de définir un objet possédant trois axes x, y et z, associés respectivement à la longueur, la hauteur et la profondeur.

2D : signifie deux dimensions. Ces dimensions permettent de définir un objet possédant deux axes x, y, associés respectivement à la longueur, la hauteur

Introduction générale

Depuis 1970 le monde scientifique et industriel porte des efforts sans cesse croissants sur l'étude et la réalisation d'engins mobiles. Ces derniers sont capables d'effectuer des déplacements sans aucune liaison mécanique avec un point fixe de l'espace dans lequel ils évoluent, d'où le nom de robots mobiles.

Aujourd'hui, l'industrie mondiale subit une véritable mutation qui se modifie profondément. La robotisation constitue une nouvelle avancée de l'automatisation grâce aux progrès de la micro-électronique permettant la mémorisation des informations de réglage et de contrôle pour un coût abordable et un volume réduit.

La volonté des chercheurs à reproduire les capacités humaines de perception et d'action dans les systèmes robotisés a conduit à l'intégration des données issues de capteurs extéroceptifs, et plus particulièrement de celles issues d'une caméra. L'objectif affiché est que les capteurs visuels fournissent une information suffisamment riche pour permettre aux robots de réaliser, de manière autonome, des tâches dans des environnements partiellement connus, ou complètement inconnus. D'un point de vue méthodologique, l'asservissement visuel consiste à intégrer directement dans la boucle de commande des robots, des informations extraites des images fournies par des caméras afin de réaliser l'action souhaitée.

Dans ce contexte, notre travail s'articule autour de deux objectifs majeurs. Le premier étant de réaliser un asservissement visuel d'un robot mobile en utilisant une carte électronique Arduino, une carte électronique Raspberry et un système récent de vision Pixy. Le deuxième but consiste à modifier la structure matérielle existante du robot pour qu'elle puisse accueillir de nouveaux capteurs proprioceptifs, en l'occurrence des capteurs ultrason pour permettre au robot mobile d'éviter d'éventuels obstacles qui pourraient apparaitre lors de sa poursuite de la cible. Bien entendu ces apports majeurs à la plate-forme robot mobile nécessiterons forcement l'élaboration de lois de commandes complexes.

Ce document est organisé en quatre chapitres qui portent sur les aspects suivants :

Le **premier chapitre** présente une introduction à la robotique mobile, des descriptions des différents éléments constituants les robots mobiles ainsi que les différentes structures mécaniques qui existent.

Le **deuxième chapitre** est consacré aux systèmes embarqués utilisés dans notre projet, tel que le système de vision Pixy, la carte raspberry et la carte Arduino.

Le **troisième chapitre** sera pour nous l'occasion d'exposer les techniques d'asservissement visuel. Nous décrirons les déférentes architectures qui existent en asservissement visuel et nous présenterons une classification de ces systèmes de commande.

Le **quatrième chapitre** concernera le détaille de notre réalisation. Nous présenterons l'agencement et le fonctionnement de l'architecture matérielle que nous avons mise en œuvre. Par ailleurs, nous validerons, à partir d'une application réelle de suivi de cible, notre algorithme de commande basé sur l'exploitation de données visuelles.

Enfin, **la conclusion** nous permettra de résumer l'ensemble des travaux réalisés. Nous terminerons ce mémoire par quelques perspectives envisageables dans un futur proche ou plus lointain.

Chapitre 1 :

Généralités sur les robots mobiles

1. Introduction

La robotique a atteint son plus grand succès jusqu'à ce jour dans le monde de la fabrication industrielle.

Boulonné à son épaule à une position spécifique dans la ligne d'assemblage, le bras de robot peut se déplacer avec une grande rapidité et précision pour effectuer des tâches répétitives telles que le soudage par points et la peinture. Dans l'industrie de l'électronique, les manipulateurs place des composants montés en surface avec une précision surhumaine, ce qui rend le téléphone portable et ordinateur portable possible.

Cependant, pour tous leurs succès, ces robots commerciaux souffrent d'un inconvénient majeur : le manque de mobilité. Un manipulateur fixe dispose d'une gamme limitée de mouvement qui dépend de l'endroit où elle est boulonnée. En revanche, un robot mobile serait en mesure de voyager à travers l'usine de fabrication, en appliquant avec souplesse ses talents partout où il est le plus efficace.

De manière générale, on regroupe sous l'appellation robots mobiles l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs. L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues. Les autres robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens.

Le but de ce chapitre est de situé le robot mobile objet de notre étude. A partir des descriptions des différents éléments constituants les robots mobiles ainsi que les différentes structures mécaniques qui existent.

2. Aperçu historique

Ce n'est qu'au tout début du XX^{e} siècle que les robots firent leur apparition, suite aux travaux d'ingénieurs qui voulaient tester des hypothèses émises par des biologistes et des psychologues. Le chien électrique conçu par Hammond et Miessner en 1915 était attiré par une lumière, selon le phototropisme animal mis en évidence par Loeb en 1918. Les machines de Russell (1913) et de Stephens (1929), les tortues cybernétiques de Grey Walter (Figure 1.1) (1950), le renard électronique de Ducrocq (1953) ou l'homéostat d'Ashby (1952) étaient, elles, dotées de capacités d'apprentissage directement issues des travaux des psychologues Thorndike (1911), Hull (1943) et du physiologiste Pavlov (1903) sur l'Homme et l'animal.

Ces réalisations sont des robots, car elles ne se comportent plus comme de simples automates dont les organes moteurs - leurs mécanismes - obéissent à un programme préétabli. À la différence des automates, ils ont des organes sensoriels - les capteurs - qui recueillent des informations de l'environnement qui vont, elles, influencer l'activité de leurs organes moteurs - les actionneurs. [2]



Figure 1.1 : La tortue de Grey Walter.

3. Les bases mobiles

Nous présentons rapidement les différents types de bases mobiles utilisées en robotique, en nous focalisant sur les plateformes mobiles terrestres pour le milieu intérieur.

3.1 Holonomie

En robotique, une plateforme est dite holonome lorsque que le nombre de degrés de libertés contrôlables est égal au nombre total de degrés de liberté.

Pour un robot se déplaçant sur un plan, il y a 3 degrés de liberté (deux translations et une rotation).

A partir d'une position donnée, une plateforme holonome devra donc pouvoir se déplacer en avant, sur le côté et tourner sur elle-même. Cette capacité permet de contrôler très simplement le robot car tous les déplacements imaginables sont réalisables, ce qui simplifie le problème de planification de trajectoire.

De nombreuses plateformes simples ne sont pas holonomes. C'est par exemple le cas des voitures, ce qui oblige à manœuvrer pour réaliser certaines trajectoires. Par exemple, il est nécessaire de faire un créneau pour réaliser un déplacement latéral. Ces contraintes devront donc être prises en compte lors de la planification de trajectoires. Nous allons cependant voir quelques mécanismes permettant d'obtenir des plateformes holonomes, ou s'en approchant. [7]

3.2 Les plates-formes différentielles

Une des configurations les plus utilisées pour les robots mobiles d'intérieur est la configuration différentielle qui comporte deux roues commandées indépendamment. Une ou plusieurs roues folles sont ajoutées à l'avant ou à l'arrière du robot pour assurer sa stabilité (Figure1.2). Cette plate-forme est très simple à commander, puisqu'il suffit de spécifier les vitesses des deux roues, et permet de plus au robot de tourner sur place. Cette possibilité permet de traiter dans certains cas le robot comme un robot holonome, ce qui va simplifier la planification de déplacement et la commande du robot. [7]

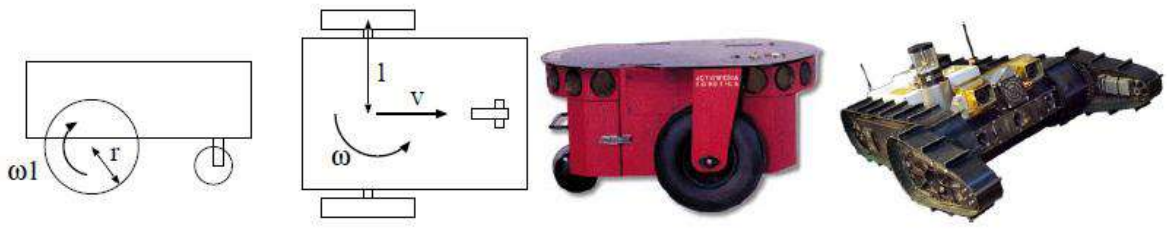


Figure 1.2 : Exemple de plate-forme différentielle. Pioneer 2 DX de la société Mobile Robots. Urban Robot de la société iRobot.

L'estimation du déplacement par odométrie est également très simple à partir de la mesure des vitesses de rotation des deux roues w1 et w2. Les vitesses de translation v et de rotation w sont en effet données par :

$$v = \frac{\omega_1 r + \omega_2 r}{2}$$
$$\omega = \frac{\omega_1 r - \omega_2 r}{2l}$$

Ce type de plate-forme peut également être utilisé avec des chenilles ce qui fournit une capacité de franchissement de petits obstacles intéressante (Figure 1.2). Ces plates-formes peuvent ainsi être utilisées en extérieur ou dans des décombres. L'utilisation de chenilles conduit cependant à une odométrie très bruitée à cause du contact mal défini entre les chenilles et le sol qui glissent beaucoup, notamment lors des rotations. L'estimation de la direction par l'odométrie sur ce type de plates-formes est donc en général rapidement inutilisable. [7]

3.3 Les plates-formes omnidirectionnelles

Les plates-formes omnidirectionnelles permettent de découpler de manière plus nette le contrôle de la rotation et de la translation d'un robot et sont donc quasiment holonomes.

Il existe différents types de plateformes omnidirectionnelles. Le premier utilise trois ou quatre roues qui tournent à la même vitesse pour fournir une translation et un mécanisme qui permet d'orienter simultanément ces roues dans la direction du déplacement souhaitée

Le corps du robot lui-même n'effectue pas de rotation mais uniquement des translations. Ce système permet un contrôle très simple et relativement rapide car les changements de direction ne concernent que les roues et peuvent donc se faire très vite. Par contre ces plates-formes sont relativement limitées en capacité de franchissement et requièrent un sol très plan.

Une deuxième catégorie de plateformes utilise des roues dites "suédoises", qui n'offrent pas de résistance au déplacement latéral (Figure 1.3). La plateforme comporte trois roues dont les axes sont fixes. Les déplacements dans toutes les directions et en rotation sont obtenus en faisant varier individuellement les vitesses des roues. La plateforme tourne sur place lorsque les trois

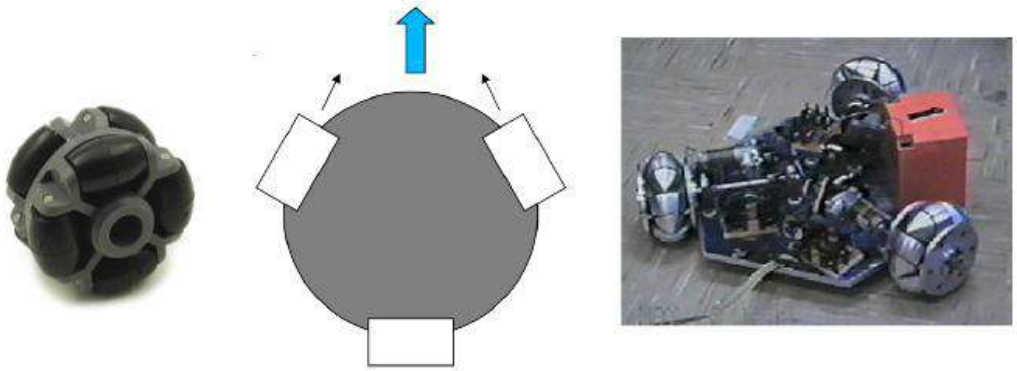


Figure 1.3 : Exemple de plate-forme omnidirectionnelle à roues suédoises.

Roues tournent dans le même sens, à la même vitesse. Lorsqu'une roue est fixe, et que les deux autres tournent en sens opposé, la plateforme avance en direction de la roue fixe. Différentes combinaisons de vitesses permettent d'obtenir des déplacements quelconques.

3.4 Les plates-formes non holonomes

Des plates-formes non holonomes, telles que les voitures, sont également utilisées en robotique mobile (Figure 1.4). C'est plus particulièrement le cas dans le domaine des véhicules intelligents. Ces plates-formes sont toutefois plus difficiles à commander car elles ne peuvent pas tourner sur place et doivent manœuvrer, ce qui peut être difficile dans des environnements encombrés.

La commande de ces plates-formes pour réaliser un déplacement particulier est un problème à part entière que nous n'aborderons pas dans ce cours. Par contre, il est possible de prendre en compte ces contraintes de manière relativement simple dans la planification.

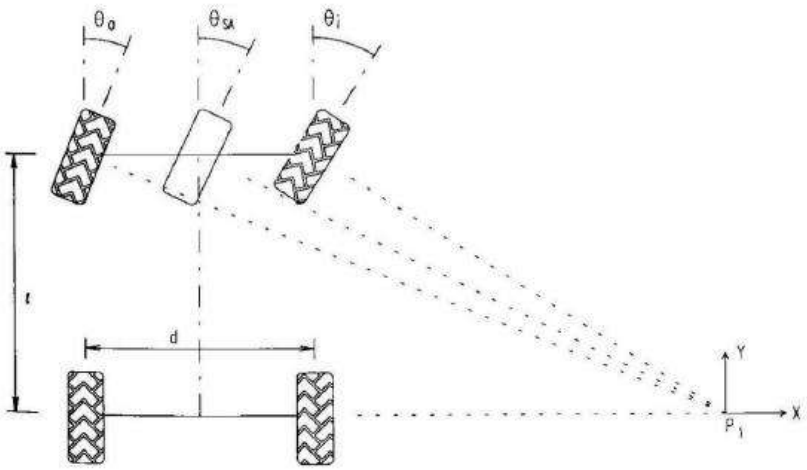


Figure 1.4 : Exemple de plate-forme non holonome de type Ackerman.

3.5 Les plates-formes à pattes

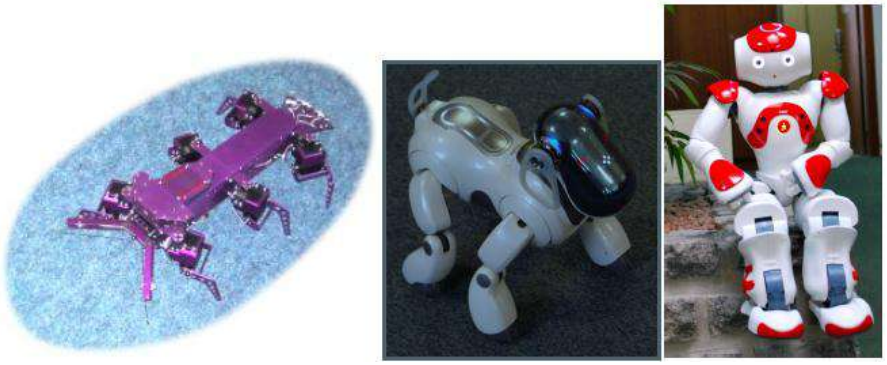


Figure 1.5: Exemples de robots à pattes. Hexapode de AAI Canada, Aibo de Sony, Nao de Aldebarran Robotics.

Des plates-formes à deux, quatre ou six pattes peuvent également être utilisées. Elles ont l'avantage théorique de pouvoir se déplacer sur des terrains assez complexes, même si en pratique la plupart de ces plates-formes ne fonctionnent que sur des sols plans. [7]

Les plates-formes à six pattes sont relativement pratiques car le robot peut être en équilibre permanent sur au moins 3 pattes, ce qui facilite le contrôle. Les plates-formes à deux ou quatre pattes sont plus complexes à commander et le simple contrôle de la stabilité et d'une allure de

marche correcte reste aujourd'hui difficile, ce qui les rend en général relativement lentes. L'odométrie de ce type de plates-formes est de plus généralement d'assez faible qualité.

Ces différents facteurs font que ces plates-formes sont rarement utilisées quand l'application visée a un besoin précis de positionnement et de navigation. De telles plates-formes commencent cependant à apparaître à relativement grande échelle. [7]

4. Les robots à roues

Il existe plusieurs classes de robots à roues déterminées, principalement, par la position et le nombre de roues utilisées.

Nous citerons ici les quatre classes principales de robots à roues.

4.1. Robot unicycle

Un robot de type unicycle est actionné par deux roues indépendantes, il possède éventuellement des roues folles pour assurer sa stabilité. Son centre de rotation est situé sur l'axe reliant les deux roues motrices.

C'est un robot non-holonome, en effet il est impossible de le déplacer dans une direction perpendiculaire aux roues de locomotion.

Sa commande peut être très simple, il est en effet assez facile de le déplacer d'un point A un autre par une suite de rotations simples et de lignes droites.

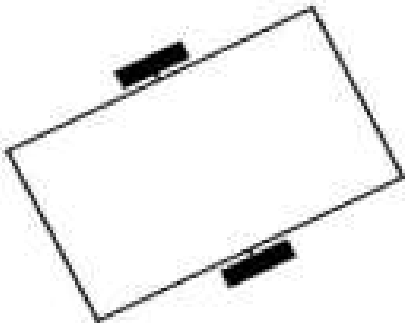


Figure 1.6 : Robot de type unicycle.

4.2. Robot tricycle

Un robot de type tricycle est constitué de deux roues fixes placées sur un même axe et d'une roue centrée orientable placée sur l'axe longitudinal. Le mouvement du robot est donné par la vitesse des deux roues fixes et par l'orientation de la roue orientable. Son centre de rotation est situé à l'intersection de l'axe contenant les roues fixes et de l'axe de la roue orientable.

C'est un robot non-holonome. En effet, il est impossible de le déplacer dans une direction perpendiculaire aux roues fixes. Sa commande est plus compliquée. Il est en général impossible d'effectuer des rotations simples à cause d'un rayon de braquage limité de la roue orientable.

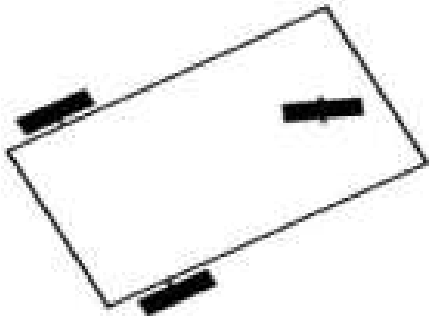


Figure 1.7 : Robot de type tricycle.

4.3. Robot voiture

Un robot de type voiture est semblable au tricycle, il est constitué de deux roues fixes placées sur un même axe et de deux roues centrées orientables placées elles aussi sur un même axe.

Le robot de type voiture est cependant plus stable puisqu'il possède un point d'appui supplémentaire.

Toutes les autres propriétés du robot voiture sont identiques au robot tricycle, le deuxième pouvant être ramené au premier en remplaçant les deux roues avant par une seule placée au centre de l'axe, et ceci de manière à laisser le centre de rotation inchangé.

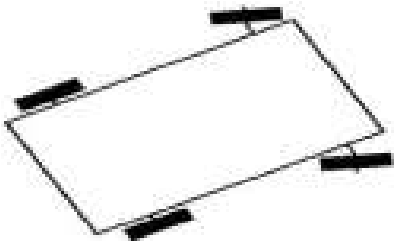


Figure 1.8 : Robot de type voiture.

4.4. Robot omnidirectionnel

Un robot omnidirectionnel est un robot qui peut se déplacer librement dans toutes les directions. Il est en général constitué de trois roues décentrées orientables placées en triangle équilatéral.

L'énorme avantage du robot omnidirectionnel est qu'il est holonome puisqu'il peut se déplacer dans toutes les directions. Mais ceci se fait au dépend d'une complexité mécanique bien plus grande.



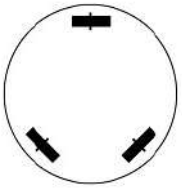


Figure 1.9 : Robot de type omnidirectionnel Comparaison des différents types.

Tableau 1.1 : Les avantages et des inconvénients des différents types de robots à roues.

Type de robot	Avantage	Inconvénient
Unicycle	• Stable • Rotation sur soi-même • Complexité mécanique faible	• Non-holonome
Tricycle	• Complexité mécanique modérée	• Non-holonome • Peu stable • Pas de rotation sur soi-même
Voiture	• Stable • Complexité mécanique modérée	• Non-holonome • Pas de rotation sur soi-même
Omnidirectionnel	• Holonome • Stable • Rotation sur soi-même	• Complexité mécanique importante

5. La robotique mobile tout-terrain

Un système de locomotion terrestre a pour finalité le déplacement d'un corps grâce à la génération de forces s'opposant aux phénomènes de résistance au mouvement. Dans le cas des véhicules à roues, le mécanisme de propulsion généré des couples qui mettent ces roues en rotation. La friction ou l'adhérence des roues sur le sol permet de convertir ces couples actionneurs en force motrice du véhicule. Le mouvement résultant dépend de l'environnement (résistance aérodynamique, interaction roue-sol, topologie du terrain et gravité) et du véhicule lui-même (géométrie, dynamique, frottements internes).

En robotique mobile, on cherche à rendre ce véhicule le plus autonome possible dans l'accomplissement de sa tâche. Cette tâche est soit le transport d'une charge utile (instruments de mesure, bras articulé, minerai, etc.), soit l'exploration, soit le travail sur site dans un environnement inconnu, impraticable ou dangereux pour l'être humain. [1]

On peut considérer qu'il s'agit du premier robot mobile dont la liaison perception action n'est plus une simple boucle réflexe mais résulte d'un traitement relativement complexe. Son objectif principal était la navigation dans un environnement très structuré tel que l'intérieur d'un bâtiment.

L'exploration et le transport en environnement non structuré, c'est-`a-dire non créé et non modifié par l'homme, apparaissent comme un enjeu majeur. Les robots mobiles à roues représentent une des solutions possibles. C'est le cas de l'exploration planétaire extraterrestre. Le premier véhicule planétaire automatique télé opéré est déployé en 1970

En 1997, la mission Pathfinder de la NASA[2] envoie Séjourner, le premier robot mobile



a. Robot lunaire Lunokhod b. Robot Spirit/Opportunity (image d'artiste)

Figure 1.10 : Robots d'exploration planétaires.

Cette chronologie succincte montre les développements des dernières décennies dans le domaine de la robotique mobile. Au cours des premières décennies, l'invention de l'ordinateur (1948) et le développement de l'automatique en tant que science permet la conception de véhicules autonomes d'intérieur.

La conquête du milieu non artificiel par les robots mobiles commence dans les années soixante. Mais l'utilisation de véhicules avec un degré d'autonomie dépassant la télé opération est plus récente grâce, en particulier, à la miniaturisation de la logique de commande (microprocesseur) et des capteurs.

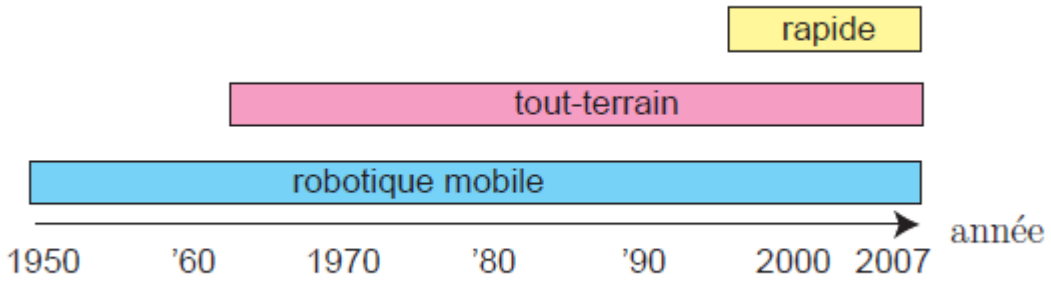


Figure 1.11 : Chronologie de la robotique mobile.

Il existe aujourd'hui de nombreux systèmes robotiques en environnement naturel exploités pour diverses applications.

La (figure1.12) montre quelques exemples pour le cas de l'exploration de planètes extraterrestres.

D'une manière générale, les recherches portent sur l'amélioration des performances des robots en matière de franchissement. L'attention est focalisée sur la navigation et la locomotion sur des terrains non structurés, présentant des difficultés topologiques (pentes, dépressions, obstacles, etc.).



a. Sojourner



b. Shrimp [Siegwart *et al.* 2002]



c. Nexus 6 [Yoshida et Hamano 2002b]



d. Robovolc [Caltabiano *et al.* 2004]

Figure 1.12 : Quelques exemples de robots mobiles tout-terrain.

Le problème des véhicules autonomes à grande vitesse a été peu abordé jusqu'ici. Ce concept de grande vitesse pour un robot mobile dépend du terrain traversé et du véhicule lui-même. On considérera qu'un véhicule autonome se déplace à grande vitesse sur un terrain donné si les effets dynamiques ont une influence importante sur sa trajectoire, au point d'engendrer des comportements spécifiques (retournements, glissement, dérapage, chutes balistiques, etc.)

Il existe plusieurs véhicules télé opérés rapides, principalement utilisés pour des applications militaires. (Figure 1.13)



Figure 1.13 : Quelques exemples d'applications militaires pour les robots mobiles.

Citons le Dragon Runner, développé par l'institut de robotique de l'Université de Carnegie Mellon. La plateforme est réversible et les roues sont directionnelles. Le RobucarTT est un véhicule à quatre roues motrices indépendantes. Toutes les roues sont directionnelles.



a. Rover Dragon Runner b. RobucarTT

Figure 1.14 : Exemples de robots mobiles rapides.

Le tableau 1.2 donne les caractéristiques de quelques systèmes robotiques rapides (d´passant 5 m/s). Il s'agit généralement de systèmes télé opérés avec une faible autonomie décisionnelle.

Tableau 1.2 : Caractéristiques de quelques robots mobiles rapides.

Robot	Longueur (Cm)	Largeur (Cm)	Poids (Kg)	Vitesse Maximale (m/s)
Drogon Runner	39,4	28,6	7,4	9
RobucarTT	184	130	350	5
UGCV RST	213	137	450	11
ARTmiS [Shimoda et al. 2005]	89	61	35,4	7

La commande de robots mobiles rapides implique des contraintes spécifiques que nous détaillons dans ce mémoire. Nous verrons en particulier qu'on ne peut plus supposer le roulement sans glissement et donc la non-holonomie du système. [1]

6. Capteurs utilisés dans les systèmes de robotique mobile

Pour réaliser un système de robotique mobile intelligent, il est nécessaire d'utiliser des capteurs qui fourniront la perception requise de l'environnement pour une prise de décision intelligente. A ce jour, il n'existe pas de capteur qui se démarque des autres par une efficacité vraiment supérieure. Ils ont tous des avantages et des inconvénients avec lesquels il faut pouvoir composer. Par contre, leur importance peut être passablement influencée par l'environnement dans lequel le système est amené à évoluer. Le défi est donc de réaliser des systèmes qui utilisent le mieux possible les capteurs ou une combinaison de capteurs en tenant compte de leurs conditions d'utilisation, et ceci requiert une bonne connaissance de leurs caractéristiques. Nous pouvons définir deux catégories de capteurs couramment utilisées en robotique mobile : ceux qui délivrent une information caractérisant l'environnement et ceux qui interviennent dans le fonctionnement du système robotique. Les premiers sont appelés capteurs extéroceptifs qui fournissent des informations sur le monde extérieur au système mobile, et les seconds sont les capteurs proprioceptifs qui fournissent des informations propres au comportement interne du système mobile, c'est-à-dire sur son état à un instant donné. Dans ce qui va suivre, nous allons nous contenter de décrire quelques capteurs extéroceptifs. [3] [4].

6.1. Les Capteurs extéroceptifs

Les capteurs extéroceptifs sont employés en robotique mobile pour collecter des informations sur l'environnement d'évolution du système mobile. Ils sont le complément indispensable aux capteurs proprioceptifs présentés précédemment. Des méthodes de fusion de données sont alors utilisées pour conditionner et traiter les informations sensorielles de natures différentes. Ils sont notamment utilisés dans les domaines d'application tels que l'évitement d'obstacle, la localisation, la navigation et la modélisation d'environnements. Les principaux capteurs utilisés en robotique mobile sont : les capteurs télémétriques (les ultrasons, les lasers et les infrarouges), le GPS et les caméras. [8]

a) Les capteurs télémétriques

On appelle télémétrie toute technique de mesure de distance par des procédés acoustiques, optiques ou radioélectriques. L'appareil permettant de mesurer les distances est appelé télémètre. Il existe différentes technologies pour réaliser un télémètre. Nous présentons dans ce qui suit les télémètres les plus utilisés dans les systèmes de robotique mobile, en donnant une idée de leur gamme de mesure et d'application. [8]

b) Les GPS

Un des systèmes les plus connus de ces dernières années est le Système de Positionnement Global (GPS) du département de la défense des Etats-Unis qui exploite une constellation de vingt-quatre satellites en orbite à haute altitude.

Chaque satellite détecte les récepteurs terrestres fixes ou mobiles qui se trouvent dans sa zone de visibilité et transmet à chacun, à période fixe, un signal contenant des informations telles que son identification, l'heure d'émission du signal et les paramètres définissant son orbite donc sa position. Théoriquement, un récepteur doit être visible d'au moins trois satellites, pour pouvoir calculer sa position dans un repère centré sur la terre, en calculant l'intersection de trois sphères (trois cercles pour un système mobile au sol) ; de fait, pour tenir compte des décalages temporels entre les mesures fournies par les satellites, quatre mesures de distance entre le satellite et le récepteur sont requises pour disposer d'une précision satisfaisante [8].

Ce système est cependant loin de résoudre tous les problèmes de localisation des systèmes mobiles. Il fonctionne en effet difficilement dans des environnements urbains, et n'est pas

utilisable à l'intérieur des bâtiments car les ondes employées sont fortement atténuées en traversant les murs. Sa précision est de plus souvent trop faible pour qu'un système mobile terrestre puisse utiliser ces informations seules. En pratique, il est souvent couplé à un système inertiel qui permet de pallier aux pertes du signal GPS et il ne remplace de toute façon pas les capteurs du système mobile qui lui permettent de percevoir son environnement immédiat [7].

Mis à part son utilisation pour la localisation, ce système est aussi utilisé pour l'aide à la navigation d'avions et de bateaux lors de l'approche d'aéroports ou de ports, ainsi que dans la surveillance de camionnettes de livraison et de véhicules d'urgence, en vue de choisir au mieux leurs itinéraires.

c) Les caméras

Depuis les années 70, les caméras CCD (Charge Coupled Device) ont été déterminantes dans l'évolution de la vision : la rapidité d'acquisition, la robustesse et la miniaturisation sont autant d'avantages qui ont facilité leur intégration.

Elles sont très performantes en termes de portée, précision et quantité d'informations exploitables.

Elles sont de plus les seules capables de restituer une image sensorielle de l'environnement la plus proche de celle perçue par l'être humain. C'est sans doute le capteur fournissant la plus grande variété d'informations à partir d'une seule acquisition. En effet, en appliquant divers algorithmes de traitement d'images, il est possible d'extraire des éléments caractéristiques de l'environnement comme des segments de droite, des plans, mais aussi des balises et des couleurs.

En revanche, l'inconvénient majeur de tels capteurs se situe d'abord au niveau de la gestion du flux important de données exploitables (traiter une image demeure une opération délicate et surtout coûteuse en temps de calcul), ensuite à leur sensibilité aux conditions d'éclairage.

Pour augmenter le champ de vision de ces caméras, on peut les associer soit avec d'autres caméras ou soit à des systèmes de réflexion de type miroir : il s'agit des techniques de vision omnidirectionnelle. Cette association permet d'obtenir une vue sur 360 degrés de l'environnement [5].

7. Les stratégies de navigation

Les stratégies de navigation permettant à un robot mobile de se déplacer pour rejoindre un but sont extrêmement diverses, de même que les classifications qui peuvent en être faites. Afin de situer les différentes méthodes de navigation que nous allons étudier dans un contexte général, nous reprenons ici une classification établie par Trullier [7]. Cette classification a été établie en prenant en compte à la fois les stratégies des robots et des animaux. Elle présente l'avantage de distinguer les stratégies sans modèles internes et les stratégies avec modèle interne.

Cette classification comporte cinq catégories, de la plus simple à la plus complexe :

7.1. Approche d'un objet

Cette capacité de base permet de se diriger vers un objet visible depuis la position courante du robot. Elle est en général réalisée par une remontée de gradient basée sur la perception de l'objet, comme dans l'exemple célèbre des véhicules de Valentino Braitenberg qui utilisent deux capteurs de lumière pour atteindre ou fuir une source lumineuse. Cette stratégie utilise des actions réflexes, dans lesquelles chaque perception est directement associée à une action. C'est une stratégie locale, c'est-à-dire fonctionnelle uniquement dans la zone de l'environnement pour laquelle le but est visible. [7]

7.2. Guidage

Cette capacité permet d'atteindre un but qui n'est pas un objet matériel directement visible, mais un point de l'espace caractérisé par la configuration spatiale d'un ensemble d'objets remarquables, ou amers, qui l'entourent ou qui en sont voisins. La stratégie de navigation, souvent une descente de gradient également, consiste alors à se diriger dans la direction qui permet de reproduire cette configuration. Cette capacité semble utilisée par certains insectes, comme les abeilles, et a été utilisée sur divers robots. Cette stratégie utilise également des actions réflexes et réalise une navigation locale qui requiert que les amers caractérisant le but soient visibles. [7]

7.3. Action associée à un lieu

Cette capacité est la première capacité réalisant une navigation globale, c'est-à-dire qui permette de rejoindre un but depuis des positions pour

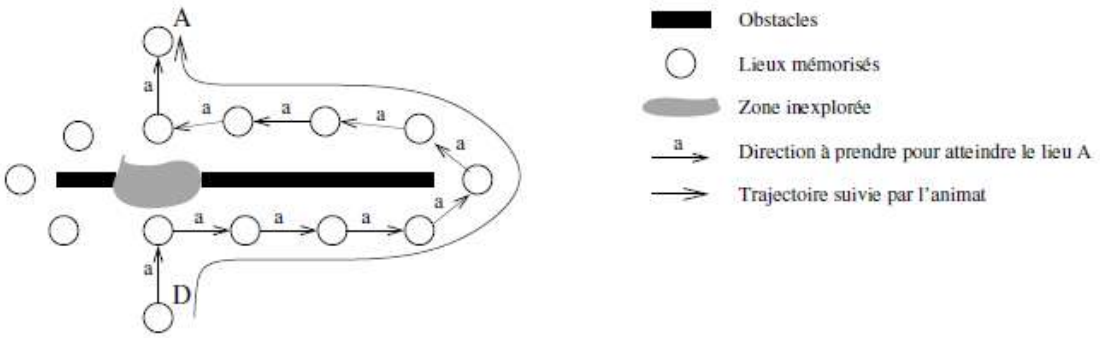


Figure 1.15 : Action associée à un lieu.

En chaque lieu, représenté par un cercle, l'action à accomplir pour rejoindre le but A est représentée par une flèche indiquant la direction à suivre à partir de ce lieu. Cette stratégie permet de rejoindre un but distant dans l'environnement mais repose sur des chemins figés. Dans cet exemple, le chemin joignant le lieu D au lieu A et passant par la droite de l'obstacle a été appris. Rejoindre le lieu A depuis le lieu D ne pourra alors être réalisé que par ce chemin. Le raccourci empruntant le chemin de gauche, par exemple, est inutilisable. Les quelles se but ou les amers qui caractérisent son emplacement sont invisibles. Elle requiert une représentation interne de l'environnement qui consiste à définir des lieux comme des zones de l'espace dans lesquelles les perceptions restent similaires, et à associer une action à effectuer à chacun de ces lieux (Figure 1.15).

L'enchaînement des actions associées à chacun des lieux reconnus définit une route qui permet de rejoindre le but. Ces modèles permettent donc une autonomie plus importante mais sont limités à un but fixé. Une route qui permet de rejoindre un but ne pourra en effet pas être utilisée pour rejoindre un but différent. Changer de but entraînera l'apprentissage d'une nouvelle route, indépendante des routes permettant de rejoindre les autres buts.

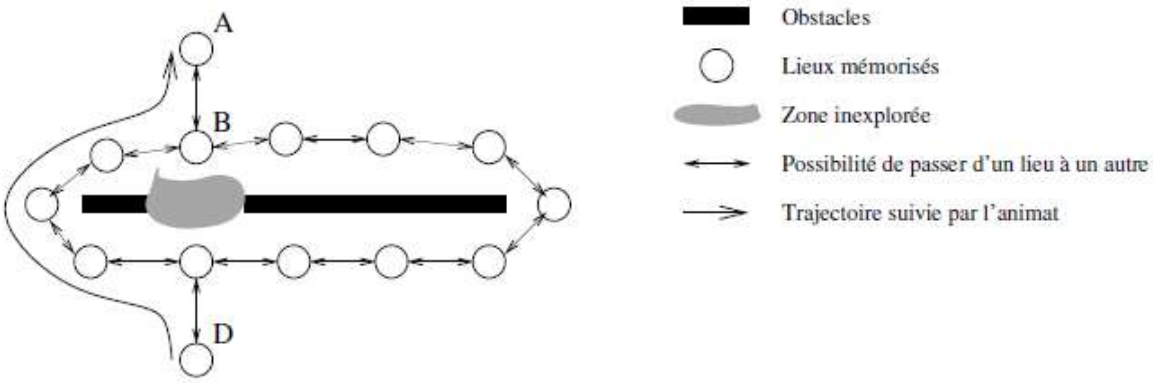


Figure 1.16 : Navigation topologique**.**

Cette stratégie permet de mémoriser un ensemble de lieux et les possibilités de passer de l'un à l'autre, indépendamment de tout but. Pour rejoindre un but, il faut alors une étape de planification qui permet de rechercher, parmi tous les chemins possibles, le chemin rejoignant le but. Dans notre exemple, le chemin le plus court entre D et A peut alors être calculé, mais uniquement parmi les lieux et les chemins déjà connus. Cette stratégie permet, par exemple, de contourner l'obstacle par la gauche mais ne permet pas de le traverser en ligne droite de D à A. [7]

7.4. Navigation topologique

Cette capacité est une extension de la précédente qui mémorise dans le modèle interne les relations spatiales entre les différents lieux. Ces relations indiquent la possibilité de se déplacer d'un lieu à un autre, mais ne sont plus associées à un but particulier. Ainsi le modèle interne est un graphe qui permet de calculer différents chemins entre deux lieux arbitraires. Ce modèle ne permet toutefois que la planification de déplacements parmi les lieux connus et suivant les chemins connus (Figure 1.16).

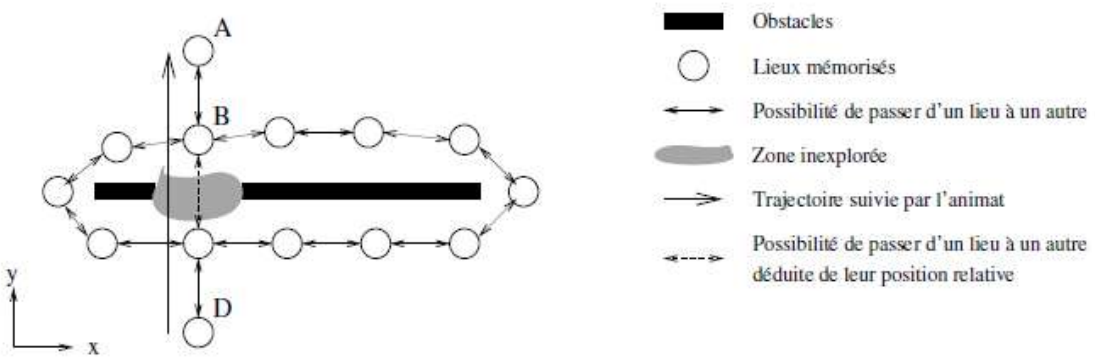


Figure 1.17 : Navigation métrique.

Cette stratégie permet de calculer le chemin le plus court entre deux lieux mémorisés, permettant même de planifier des raccourcis au sein de zones inexplorées de l'environnement. Pour cela, la carte mémorise la position métrique relative de chacun des lieux visités par le robot. Ainsi il est possible de prévoir un déplacement entre deux lieux, même si la possibilité de ce déplacement n'est pas enregistrée dans la carte.

Dans cet exemple, cette stratégie permet de d'aller du lieu A au lieu D en traversant la zone inexplorée. [7]

7.5. Navigation métrique

Cette capacité est une extension de la précédente car elle permet au robot de planifier des chemins au sein de zones inexplorées de son environnement.

Elle mémorise pour cela les positions métriques relatives des différents lieux, en plus de la possibilité de passer de l'un à l'autre. Ces positions relatives permettent, par simple composition de vecteurs, de calculer une trajectoire allant d'un lieu à un autre, même si la possibilité de ce déplacement n'a pas été mémorisée sous forme d'un lien (Figure 1.17). [7]

Les modèles des trois premières catégories utilisent des actions réflexes pour guider le robot et se différencient essentiellement par le type de perceptions utilisées pour déclencher ces actions.

Ils se regroupent sous le terme générique de navigation réactive dont nous parlerons dans la partie II. Ils peuvent être très simple, ne nécessitent pas de modèle global de l'environnement mais ont un domaine d'application souvent restreint. Dans le monde vivant, ces stratégies sont très répandues, notamment chez les insectes. Les comportements de ce type restent toutefois

essentiels dans les robots modernes car, du fait de leur simplicité, ils sont généralement exécutés très rapidement et ils permettent de réaliser des taches de bas-niveau, comme l'évitement des obstacles imprévus, essentielles à la sécurité d'un robot.

Les modèles des deux dernières catégories autorisent pour leur part une navigation globale et permettent de rejoindre un but arbitraire au sein de l'environnement. Ils s'appuient pour cela sur un modèle interne du monde, une carte, qui supporte une planification. Ce modèle interne mémorise donc la structure spatiale de l'environnement, indépendamment d'un but précis. Chacune des positions mémorisées dans ce modèle interne peut alors être utilisée comme but par le processus de planification dont le rôle est de calculer une route vers ce but. Ce sont ces deux stratégies qui sont regroupées sous le terme de navigation par carte.

Une telle représentation interne est naturelle pour les êtres humains, pour lesquels des processus cognitifs de haut niveau sont utilisés pour créer et utiliser une carte. Ces processus de haut niveau sont toutefois très difficiles à copier pour un robot réel qui ne dispose que de systèmes rudimentaires de perception et de traitement des informations en comparaison avec un homme. Par exemple, en environnement urbain, le processus de mise en correspondance de la carte avec l'environnement réel afin de déterminer sa position fait souvent appel, pour l'homme, à la lecture du nom des rues inscrit sur les bâtiments, ce qui est relativement difficile à automatiser, à cause de la diversité des configurations dans lesquelles peuvent se trouver ces noms.

On notera au passage que l'homme a quasiment toujours recours à des aménagements particuliers de l'environnement pour connaitre sa position, par exemple celui qui consiste à nommer les rues ou à lancer des satellites dans l'espace pour bénéficier du GPS. Le système de navigation idéal pour un robot mobile sera probablement celui qui sera capable de tirer parti de toutes ces informations, qui ne lui étaient pas destinées à l'origine.

L'utilisation de cartes par un robot mobile comme le font les hommes est probablement hors de notre portée pendant quelques années, cependant il existe également des preuves de l'existence de représentations internes similaires à de telles cartes chez les animaux, par exemple chez les rats. Ces représentations sont identifiables au niveau neurologique dans certaines parties de leur cerveau, notamment dans l'hippocampe. Cela montre que des cartes sont utilisées par des êtres vivants, sans le support de concept abstraits tels que les utilisent les humains. Ce type de carte qui fait appel à des structures neurologiques de base et probablement à des perceptions relativement simples, est un paradigme intéressant pour les robots mobiles.

En robotique mobile, comme pour l'homme ou certains animaux, l'utilisation de cartes est quasiment indispensable pour permettre d'effectuer des tâches de navigation dans des conditions environnementales complexes, qui ne sont pas spécialement adaptées pour le robot. La construction et l'utilisation de telles cartes posent cependant de nombreux problèmes, notamment pour garantir l'adéquation entre la carte et le monde réel. Pour cette raison, la plupart des robots trouvent aujourd'hui un compromis entre une approche réactive et une approche utilisant une carte afin de bénéficier de la rapidité et de la robustesse de la première et de la capacité de déplacement à long terme de la seconde. [7]

8. Les architectures de contrôle

Un robot est un système complexe qui doit satisfaire à des exigences variées et parfois contradictoires.

Un exemple typique pour un robot mobile est l'arbitrage qui doit être fait entre l'exécution la plus précise possible d'un plan préétabli pour atteindre un but et la prise en compte d'éléments imprévus, tels que les obstacles mobiles. Ces arbitrages, que ce soit au niveau du choix de stratégie, ou au niveau de l'utilisation des capteurs, des effecteurs ou des ressources de calcul, sont réglés par un ensemble logiciel appelé architecture de contrôle du robot. Cette architecture permet donc d'organiser les relations entre les trois grandes fonctions que sont la perception, la décision et l'action.

Selon cette définition une architecture doit donc être conçue pour un robot précis, mais en utilisant des modules génériques. De manière plus générale il existe également des règles de conception relativement générales qui permettent de réaliser ces implémentations. En fonction de ces règles, les architectures de contrôle peuvent être classées en trois grandes catégories que nous détaillerons par la suite : les contrôleurs hiérarchiques, les contrôleurs réactifs et les contrôleurs hybrides. [7]

(Figure 1.18). Comme le précise cette définition, toutes ces architectures ne diffèrent pas forcement par les méthodes élémentaires employées mais plutôt par leur agencement et leurs relations.

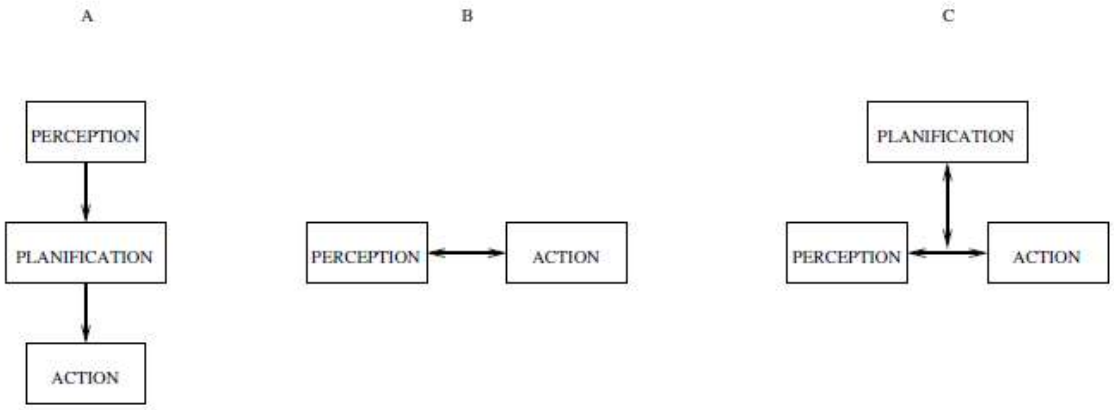


Figure 1.18 : Illustration des architectures classiques des contrôleurs pour les robots mobiles Hiérarchique (A), Réactive (B) et Hybride (C).

8.1. Contrôleurs Hiérarchiques

Historiquement, les premiers robots mobiles dérivés des recherches en intelligence artificielle utilisaient des contrôleurs hiérarchiques (Figure 1.18 A) dont le fonctionnement repose essentiellement sur la capacité de décision travaillant sur un modèle du monde supposé quasiment-parfait.

Ces architectures fonctionnent selon un cycle rigide de modélisation de l'environnement, planification des actions au sein de cette représentation, puis exécution du plan. La capacité de décision était issue des premiers travaux en intelligence artificielle et reposait essentiellement sur des traitements de données symboliques. Ces architectures ont rapidement montré leurs limites et leur incapacité à fonctionner dans un environnement qui ne soit pas statique et simplifié à l'extrême.

L'essentiel des problèmes de ces architectures provient de l'utilisation d'un modèle interne central qui est le seul pris en compte pour guider le robot. Elles se trouvent confrontées à tous les problèmes des premiers développements de l'intelligence artificielle symbolique.

Ces architectures supposent premièrement qu'un modèle informatique du monde puisse représenter toutes les informations pertinentes pour le déplacement du robot. Or un tel modèle ne peut être suffisant dans un environnement dynamique car au moment de la réalisation de l'action l'environnement peut avoir suffisamment changé pour que la décision ne soit plus valide. Ce problème était particulièrement crucial au début de la robotique mobile ou les

capacités de calcul limitées entraînaient des temps de planification de l'ordre de plusieurs dizaines de minutes. Mais l'augmentation des capacités de calcul ne suffit pas à résoudre ce problème qui ne permet pas de gérer un environnement de travail réaliste. C'est un problème intrinsèque lié d'une part à la trop grande longueur de la boucle qui relie la perception à l'action et d'autre part à l'invalidité de l'hypothèse de monde clos faite en intelligence artificielle.

De plus, ces architectures permettent peu de contrôle sur l'exécution des actions. En effet, une fois l'action choisie, elle est exécutée en supposant le modèle du monde correct et il n'y a pas de retour direct de la perception sur l'exécution de l'action. Les écarts modèles/environnement ne peuvent être pris en compte que via un nouveau cycle perception/modélisation/planification, ce qui, par définition, est très peu réactif et conduit rapidement à de graves problèmes. [7]

8.2. Contrôleurs réactifs

Rodney Brooks a proposé une solution radicale à tous ces problèmes sous la forme d'une architecture réactive (Figure 1.18 B). Dans cette architecture, un ensemble de comportements réactifs, fonctionnant en parallèle, contrôle le robot sans utiliser de modèle du monde. Cette architecture supprime évidemment les problèmes dûs aux différences entre la réalité, d'une part, et le modèle de l'environnement du robot, d'autre part, mais limite clairement les tâches que peut effectuer le robot. En effet, sans représentation interne de l'état de l'environnement, il est très difficile de planifier une suite d'actions en fonction d'un but à atteindre.

Les robots utilisant cette architecture sont donc en général efficaces pour la tâche précise pour laquelle ils ont été conçus, dans l'environnement pour lequel ils ont été prévus, mais sont souvent difficiles à adapter à une tâche différente.

Les réussites de ces architectures sont liées au couplage direct entre la perception et l'action qui permet une prise en compte très rapide des phénomènes dynamiques de l'environnement.

En donc une bonne robustesse dans des environnements complexes.

Comme nous l'avons mentionné, ces architectures sont en général basées sur plusieurs comportements : évitement d'obstacles, déplacement aléatoire, déplacement vers un but, fuite d'un point... Pour guider le robot, il faut donc choisir à chaque instant lequel de ces

comportements activer. Ce problème est connu dans la littérature scientifique sous le nom de sélection de l'action.

La solution proposée par Brooks, l'architecture de subsomption est devenue un classique et utilise une hiérarchie des comportements qui se déclenchent donc selon un ordre de priorité en fonction des perceptions du robot. [7]

8.3. Contrôleurs hybrides

La plupart des contrôleurs actuellement utilisés choisissent une solution intermédiaire entre ces deux approches sous la forme d'une architecture hybride (Figure 1.18 C). Ce haut niveau, telles que la localisation, la cartographie et la planification. Pour cela, il s'appuie sur un second niveau réactif qui est chargé d'exécuter les commandes avec le plus de précision possible et de gérer les éléments non modélisés de l'environnement tels que les obstacles inconnus ou dynamiques. L'action conjointe de ces deux niveaux permet de réagir rapidement face aux variations imprévues de l'environnement, tout en permettant la réalisation d'actions planifiées à plus long terme.

Le bas niveau de ces architectures peut être réalisé sous forme de comportements, tels que ceux utilisés dans les architectures réactives. Ces comportements sont des boucles sensorimotrices qui relient les actions aux perceptions avec une phase de décision très courte, qui assure la réactivité. Dans le même temps, les informations sensorielles sont utilisées par le haut niveau dans une boucle sensorimotrice à une échelle de temps beaucoup plus longue. C'est la mise en parallèles de ces deux échelles de temps qui fait la force de ces architectures.

Les exemples d'architectures hybrides foisonnent (4D/RCS, 3T, Harpic...) car de très nombreux laboratoires et organismes travaillant sur la robotique ont développé leur architecture de ce type. [7]

9. Conclusion

La présentation faite dans cette première partie nous a permis de décrire les éléments constituants d'un robot mobile. A partir de ce descriptif, nous avons constaté que pour réaliser la commande de ce type de robot, nous devions répondre à trois impératifs à savoir la perception, la décision et l'action.

La combinaison d'informations issues de capteur proprioceptifs et de capteurs extéroceptifs permet d'envisager la réalisation de tâches robotiques complexes. Dans le cas de la robotique mobile, le traitement de toutes ces informations nécessite une puissance de calcul importante.

Dans le prochain chapitre nous vous présenterons donc, les cartes électroniques que nous avons choisie d'utiliser pour gérer et commander le robot mobile Arduino.

Chapitre 2 :
Etude des systèmes embarqués utilisées

1. Introduction

Dans ce chapitre nous allons introduire et présenter les deux cartes électroniques embarquées ainsi que du système de vision que nous utiliserons dans l'élaboration de notre système d'acquisition de données, leur traitement et le calcul des lois de commandes nécessaires à l'asservissement de notre robot mobile. L'architecture matérielle que nous avons développée s'articule essentiellement autour de la carte Arduino de la carte Raspberry Pi et du système de vision Pixy. La caméra embarquée Pixy est considérée comme l'œil humain dans le sens ou grâce à son électronique elle nous fournira directement les caractéristiques visuelles nécessaires à la poursuite d'un objet cible. Le Raspberry Pi sera quant à lui assimilé au cerveau de notre robot tandis que le Arduino aura pour rôle la gestion de l'ensemble Capteurs / Actionneurs du robot.

Dans un premier temps, nous décrirons les cartes Arduino de manière générale, et la carte Arduino Due que nous avons utilisé.

Dans un second temps, nous décrirons l'architecture et le fonctionnement du mini-PC Raspberry Pi.

Enfin, nous détaillerons les spécifications et le fonctionnement du système de vision Pixy.

Finalement, nous vous décrirons la méthode que nous avons adopté pour faire communiqué l'ensemble des cartes et de la caméra dans le but de réaliser un ensemble cohérent capable de gérer toute les opérations nécessaires à la commande visuelle du robot mobile.

2. La carte Arduino

2.1. Définition

Arduino est une plate-forme électronique open-source basée sur un matériel et un logiciel faciles à utiliser. Arduino est né au Ivrea Interaction Design Institute comme un outil facile pour le prototypage rapide, destiné aux étudiants sans formation en électronique et en programmation. [12]

Au fil des années, Arduino a été le cerveau de milliers de projets, des objets de tous les jours aux instruments scientifiques complexes. Une communauté mondiale de décideurs - étudiants, amateurs, artistes, programmeurs et professionnels - s'est réunie autour de cette plate-forme open source, leurs contributions ont contribué à une connaissance incroyable de connaissances accessibles qui peuvent être d'une grande aide pour les novices et les experts.

2.2. Les Applications

Les cartes Arduino peuvent lire des entrées - lumière sur un capteur, un doigt sur un bouton ou un message Twitter - et le transformer en sortie - activer un moteur, allumer une LED, publier quelque chose en ligne.

L'étendue de l'utilisation de l'Arduino est gigantesque. Pour vous donner quelques exemples, vous pouvez :

- Contrôler des systèmes intelligents (smart home)
- Fabriques des robots et des machines (bras robots, cnc, imprimante 3D)
- Communiquer avec des plateformes différents (web, serveur)
- Systèmes de détection (gas, mouvement, visage)

2.3. Les atouts majeurs d'un système à carte Arduino

Grâce à son expérience utilisateur simple et accessible, Arduino a été utilisé dans des milliers de projets et d'applications différents. Le logiciel Arduino est facile à utiliser pour les débutants, mais assez flexible pour les utilisateurs avancés. Il fonctionne sur Mac, Windows et Linux. Les enseignants et les étudiants l'utilisent pour construire des instruments scientifiques à faible coût, pour prouver les principes de chimie et de physique, ou pour commencer avec la programmation et la robotique.

Les concepteurs et les architectes créent des prototypes interactifs, des musiciens et des artistes l'utilisent pour les installations et expérimentent de nouveaux instruments de musique. Les fabricants, bien sûr, l'utilisent pour construire plusieurs des projets exposés chez Maker Faire, par exemple. Arduino est un outil clé pour apprendre de nouvelles choses. Toute personne - les enfants, les amateurs, les artistes, les programmeurs - peut commencer à bricoler juste après les instructions étape par étape d'un kit, ou en partageant des idées en ligne avec d'autres membres de la communauté Arduino. [12]

- **Les cartes Arduino peu coûteuses** - sont relativement peu coûteuses par rapport aux autres plates-formes de microcontrôleurs. La version la moins chère du module Arduino

peut être assemblée à la main, et même les modules Arduino préassemblés coûtent moins de 50 $.

- **Cross-Platform** - Le logiciel Arduino (IDE) s'exécute sur les systèmes d'exploitation Windows, Macintosh OSX et Linux. La plupart des systèmes de microcontrôleurs sont limités à Windows.

- **Environnement de programmation simple et clair** - Le logiciel Arduino (IDE) est facile à utiliser pour les débutants, mais suffisamment flexible pour que les utilisateurs avancés puissent en profiter. Pour les enseignants, il est commodément basé sur l'environnement de programmation de traitement, afin que les élèves qui apprennent à programmer dans cet environnement connaîtront la façon dont l'IDE Arduino fonctionne.

- **Logiciel open source et extensible** - Le logiciel Arduino est publié sous la forme d'outils open source, disponibles pour l'extension par des programmeurs expérimentés. La langue peut être étendue à travers les bibliothèques C ++, et les personnes souhaitant comprendre les détails techniques peuvent faire passer le saut d'Arduino au langage de programmation AVR C sur lequel elles se basent. De même, vous pouvez ajouter le code AVR-C directement dans vos programmes Arduino si vous le souhaitez.

- **Matériel open source et extensible** - Les plans des cartes Arduino sont publiés sous une licence Créative Commons, de sorte que les concepteurs de circuits expérimentés peuvent créer leur propre version du module, l'étendre et l'améliorer. Même les utilisateurs relativement inexpérimentés peuvent créer la version boardboard du module pour comprendre comment cela fonctionne et économiser de l'argent. [12]

2.4. Arduino Uno

Arduino Uno est un panneau de microcontrôleur basé sur l'ATmega328P. Il dispose de 14 broches d'entrée / sortie numériques (dont 6 peuvent être utilisées comme sorties PWM), 6 entrées analogiques, un cristal à quartz 16 MHz, une connexion USB, une prise de courant, un

en-tête ICSP et un bouton de réinitialisation. Il contient tout le nécessaire pour supporter le microcontrôleur ; Il suffit de le connecter à un ordinateur avec un câble USB ou de l'alimenter avec un adaptateur AC-DC ou une batterie pour commencer. Vous pouvez bricoler avec votre UNO sans trop déranger pour faire quelque chose de mal, le pire scénario que vous pouvez remplacer Puce pour quelques dollars et recommencer. [11]

2.5. Structure de la carte Arduino Uno

Dans ce qui va suivre, nous détaillerons la structure de ce système et leurs intérêts.

- **Le microcontrôleur**

Le microcontrôleur correspond le cerveau de notre carte, c'est lui qui va traiter provenant des capteurs dans l'entrée et qui va donner la réponse à la sortie à travers le programme que vous avez créé et qui va le stocker dans sa mémoire.

- **L'alimentation**

Il faut tout d'abord connaître les spécifications de la carte Arduino que l'on possède. Elles utilisent tout d'abord toutes du courant continu.

Si l'on regarde les différentes broches possibles, on va généralement trouver une broche étiquetée 5V, une broche étiquetée 3,3V et une broche étiquetée VIN ou RAW.

- VIN ou RAW correspondent à l'entrée du régulateur de tension présent sur la carte. Pour alimenter l'Arduino via cette broche, il faut une alimentation dont la tension est comprise entre 7 et 12V ;
- 5V n'a pas le même rôle selon que le microcontrôleur qui équipe l'Arduino est alimenté en 5V ou en 3,3V. Dans le premier cas, c'est directement la tension d'alimentation du microcontrôleur. Dans le second cas, le microcontrôleur est alimenté via un régulateur 3,3V dont l'entrée est connectée sur le 5V ;
- De la même manière, le 3,3V va être la tension d'alimentation du microcontrôleur pour ceux alimentés en 3,3V et une tension de service pour connecter des dispositifs externes nécessitant du 3,3V si le microcontrôleur nécessite du 5V.

En peut alimenter la carte Arduino par :

- Alimentation par USB
- Alimentation par connecteur Jack
- Alimentation par 2 broches de l'Arduino : VIN et GND [9]

- **La visualisation**

On a trois LED qu'on trouve sur la carte donner à deux choses :

- **<u>La première (L) :</u>** elle est connectée à une broche du microcontrôleur et va servir pour tester le matériel.

  Nota : Quand on branche la carte au PC, elle clignote quelques secondes.

- **<u>Les deux LED :</u>** servent à visualiser l'activité sur la voie série (une pour l'émission (TX) et l'autre pour la réception (LX)).

  Le téléchargement du programme dans le microcontrôleur se faisant par cette voie, on peut les voir clignoter lors du chargement.

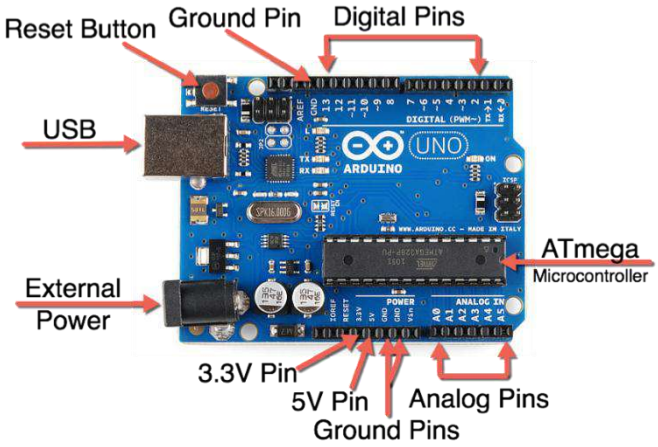


Figure 2.1 : Structure de la carte Arduino Uno.

2.6. Spécifications techniques

Tableau 2.1 : caractéristiques de la carte Arduino Uno.

Microcontrôleur	ATmega328P
Tension de fonctionnement	5V
Tension d'entrée (recommandée)	7-12V
Tension d'entrée (limite)	6-20V
Puces d'E / S numériques	14 (dont 6 fournissent une sortie PWM)
PWM Digital I / O Pins	6
Broches d'entrée analogique	6
Courant continu par broche E / S	20 mA
Courant pour broche de 3.3V	50 mA
Mémoire Flash	32Ko(ATmega328P) Dont 0,5 Ko utilisé par boot loader
SRAM	2 Ko (ATmega328P)
EEPROM	1 KB (ATmega328P)
Vitesse de l'horloge	16 MHz
LED_BUILTIN	13
Longueur	68,6 mm
Largeur	53,4 mm
Poids	25 g

2.7. Le Logiciel de gestion de la carte Arduino

Arduino avoir un environnement spéciale (IDE en anglais) est une application écrite en java inspiré du langage Processing, L'IDE permet d'écrire, de modifier un programme et de convertir en une série d'instructions compréhensibles pour la carte.

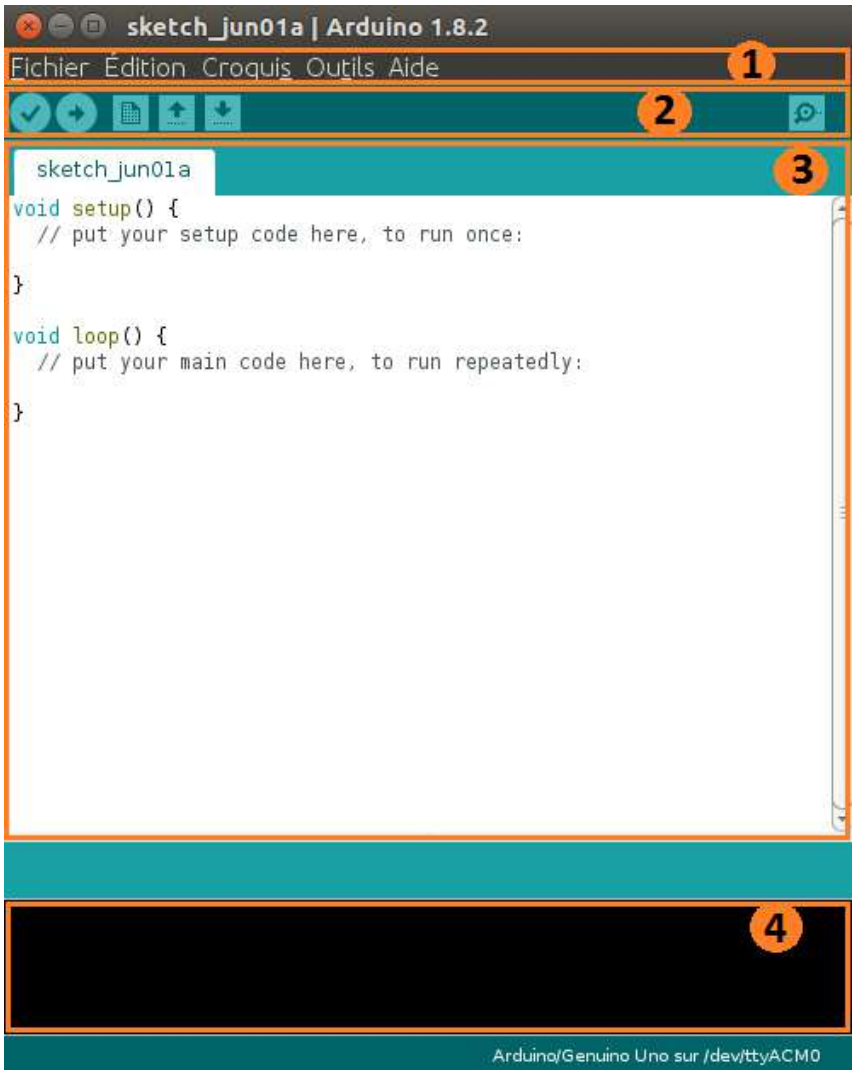


Figure 2.2 : logiciel Arduino.

- Le cadre numéro 1 : options de configuration du logiciel.
- Le cadre numéro 2 : il contient les boutons qui vont nous servir lorsque l'on va programmer nos cartes.
- Le cadre numéro 3 : ce bloc va contenir le programme que nous allons créer.
- Le cadre numéro 4 : celui-ci est important, car il va nous aider à corriger les

  Fautes dans notre programme. C'est le débogueur.

- **Les boutons**

Voyons à présent à quoi servent les boutons, encadrés en orange et numérotés par le chiffre 2.



Figure 2.3 : Présentation des boutons.

1. Vérifiée le programme.

2. Compile le code et exporte le programme dans le microcontrôleur.

3. Création d'un nouveau fichier.

4. Ouverture d'un fichier.

5. Sauvegarde le fichier.

6. Ouvre le moniteur série.

- **Moniteur série**

Le moniteur sériel est utilisé pour afficher l'information qui est envoyée par la carte Arduino vers l'application (habituellement par le câble USB). Il permet aussi d'envoyer de l'information à la carte Arduino. Pour ce faire, il suffit d'écrire du texte dans le champ situé en haut de la fenêtre et d'appuyer sur le bouton « Send ». Bien évidemment, avant de pouvoir faire ceci il vous faut coder un programme approprié, prêt à recevoir cette information. Il est également possible de régler le baud rate du moniteur sériel, qui précise à quelle vitesse le transfert des données s'effectuera. Il est également possible d'établir une communication sérielle entre la carte Arduino et d'autres périphériques ou logiciels. Nous verrons plus en détails cette fonction dans les projets « Texte brillant » et « Oscilloscope ». [12]

2.8. Comparaison Arduino Uno vs Arduino Due

Tableau 2.2 : comparaison des caractéristiques de la carte Uno vs due.

La carte Arduino	Uno	Due
Processeur	ATmega328P	ATSAM3X8E
Fonctionnement / Entrée Tension	5 V / 7-12 V	3.3 V / 7-12 V
Vitesse du processeur	16 MHz	84 MHz
Entrée / sortie analogique	6/0	12/2
Digital IO / PWM	14/6	54/12
EEPROM [kB]	1	-
SRAM [kB]	2	96
Flash [kB]	32	512
USB	Ordinaire	2 Micro
UART	1	4

Dans le tableau précèdent, nous avons voulu comparé les principales caractéristiques des deux cartes Arduino, à savoir le Uno et le Due. Il en ressort que l'utilisation de Arduino Due, nous permettra de réaliser des fonctions de commande plus évoluées et plus complexes en termes d'espace mémoire et de vitesse d'exécution. De plus l'augmentation du prix est négligeable au vu des performances remarquable de l'Arduino Due.

3. La carte Arduino Due

L'année dernière, nos collègues utilisons la carte Arduino Mega AT2560, cette année nous nous utilisons une carte Arduino plus performants, dans la partie suivante nous allons présenter la carte Arduino Due qui nous allons utiliser dans notre projet.

3.1. Présentation de la carte Arduino Due

L'Arduino Due est une carte de microcontrôleur basé sur la CPU Atmel SAM3X8E ARM Cortex-M3. C'est le première carte Arduino basé sur un microcontrôleur central ARM 32 bits. Il dispose de 54 broches d'entrée / sortie numériques (dont 12 peuvent être utilisées comme sorties PWM), 12 entrées analogiques, 4 UART (ports série matériels), une horloge à 84 MHz, une connexion USB OTG capable, 2 DAC (numérique à analogique), 2 TWI, une prise de courant, un en-tête SPI, un en-tête JTAG, un bouton de réinitialisation et un bouton d'effacement. [13]

Avertissement : Contrairement à la plupart des cartes Arduino, la carte Arduino Due fonctionne à 3.3V. La tension maximale que les broches E / S peuvent tolérer est 3.3V. L'application de tensions supérieures à 3,3 V sur une broche d'E / S pourrait endommager la carte.

La carte contient tout le nécessaire pour supporter le microcontrôleur ; Il suffit de le connecter à un ordinateur avec un câble micro-USB ou de l'alimenter avec un adaptateur AC-DC ou une batterie pour démarrer. Le Due est compatible avec tous les shields Arduino qui fonctionnent à 3.3V. [13]

3.2. Spécifications techniques

AVR Arduino Microcontrôleur

Tableau 2.3 : caractéristiques de la carte Arduino due.

Microcontrôleur	AT91SAM3X8E
Tension de fonctionnement	3.3V
Tension d'entrée (recommandée)	7-12V
Tension d'entrée (limites)	6-16V
Puces d'E / S numériques	54 (dont 12 fournissent une sortie PWM)
Goupilles d'entrée analogiques	12
Broches de sortie analogique	2 (DAC)
Courant de sortie continu total sur toutes les lignes d'E / S	130 mA
Courant continu pour broche de 3,3V	800 mA
Courant continu pour 5V Broche	800 mA
Mémoire Flash	512 KB disponible pour les applications utilisateur
SRAM	96 KB (deux banques : 64KB et 32KB)
Vitesse de l'horloge	84 MHz
Longueur	101.52 mm
Largeur	53,3 mm
Poids	36 g

3.3. Avantages de ARM Core

Un noyau 32 bits, qui permet des opérations sur des données de 4 octets de largeur dans une seule horloge de processeur. [13]

- Horloge CPU à 84Mhz.
- 96 KBytes de SRAM.
- 512 Ko de mémoire Flash pour le code.
- Un contrôleur DMA, qui peut décharger la CPU de tâches intensives en mémoire.

3.4. Alimentation

L'Arduino Due peut être alimenté via le connecteur USB ou avec une alimentation externe. La source d'alimentation est sélectionnée automatiquement.

L'alimentation externe (non-USB) peut provenir d'un adaptateur AC-DC (verrue murale) ou d'une batterie. L'adaptateur peut être connecté en branchant une prise centrale positive de 2,1 mm dans la prise d'alimentation de la carte. Les câbles d'une batterie peuvent être insérés dans les en-têtes de broche GND et Vin du connecteur POWER.

La carte peut fonctionner sur une alimentation externe de 6 à 20 volts. S'il est fourni avec moins de 7V, cependant, la broche 5V peut fournir moins de cinq volts et la carte peut être instable. Si vous utilisez plus de 12 V, le régulateur de tension peut surchauffer et endommager la carte. La plage recommandée est de 7 à 12 volts. [13]

Les broches d'alimentation sont les suivantes :

- **Vin**. La tension d'entrée à la carte Arduino lorsqu'elle utilise une source d'alimentation externe (par opposition à 5 volts à partir de la connexion USB ou d'une autre source d'alimentation régulée). Vous pouvez alimenter la tension à travers cette broche, ou si vous fournissez une tension via la prise d'alimentation, accédez-la à cette broche.

- **5V**. Cette broche produit un 5V régulé du régulateur sur la carte. La carte peut être alimenté par la prise de courant continu (7 - 12V), le connecteur USB (5V) ou la broche VIN de la carte (7-12V). La tension d'alimentation via les broches 5V ou 3.3V contourne le régulateur et peut endommager votre carte. Nous ne le conseillons pas.

- **3V3**. Une alimentation de 3,3 volts générée par le régulateur embarqué. Le tirage maximum de courant est de 800 mA. Ce régulateur fournit également l'alimentation du microcontrôleur SAM3X.

- **GND**. Broches au sol.

- **IOREF**. Cette broche sur la carte Arduino fournit la référence de tension avec laquelle le microcontrôleur fonctionne. Un blindage correctement configuré peut lire la tension de la broche IOREF et sélectionner la source d'alimentation appropriée ou activer les traducteurs de tension sur les sorties pour travailler avec les 5V ou 3.3V.

3.5. Mémoire

Le SAM3X possède 512 Ko (2 blocs de 256 Ko) de mémoire flash pour le stockage du code. Le chargeur de démarrage (boot loader) est pré chargé en usine à partir d'Atmel et est stocké dans une mémoire ROM dédiée. La SRAM disponible est de 96 Ko sur deux banques contiguës de 64 Ko et 32 Ko. Toute la mémoire disponible (Flash, RAM et ROM) peut être accessible directement comme un espace d'adressage plat.

Il est possible d'effacer la mémoire flash du SAM3X avec le bouton d'effacement intégré. Cela supprime l'esquisse actuellement chargée de la MCU. Pour effacer, appuyez et maintenez enfoncer le bouton Effacer pendant quelques secondes pendant que la carte est alimentée. [13]

3.6. Entrée et sortie

- **E / S numériques : broches de 0 à 53**

  Chacune des 54 broches numériques sur Due peut être utilisée comme entrée ou sortie, en utilisant les fonctions pinMode (), digitalWrite () et digitalRead (). Ils fonctionnent à 3,3 volts. Chaque broche peut fournir (source) un courant de 3 mA ou 15 mA, selon la broche, ou recevoir (évier) un courant de 6 mA ou 9 mA, selon la broche. Ils ont également une résistance d'extraction interne (déconnectée par défaut) de 100 KOhm. En outre, certaines broches ont des fonctions spécialisées :

  -Serial : 0 (RX) et 1 (TX)
  -Série 1 : 19 (RX) et 18 (TX)
  -Serial 2 : 17 (RX) et 16 (TX)
  -Serial 3 : 15 (RX) et 14 (TX)

Permet de recevoir (RX) et transmettre (TX) des données en série TTL (avec un niveau de 3,3 V). Les broches 0 et 1 sont connectées aux broches correspondantes de la puce série ATmega16U2 USB-to-TTL.

- **PWM : Pins 2 à 13**

  Fournir une sortie PWM 8 bits avec la fonction analogWrite (). La résolution du PWM peut être modifiée avec la fonction analogWriteResolution ().

- **SPI : En-tête SPI (en-tête ICSP sur d'autres cartes Arduino)**

  Ces broches supportent la communication SPI à l'aide de la bibliothèque SPI. Les broches SPI sont éclatées sur l'en-tête central à 6 broches, qui est physiquement compatible avec Uno, Leonardo et Mega2560. L'en-tête SPI peut être utilisé uniquement pour communiquer avec d'autres périphériques SPI, et non pour la programmation du SAM3X avec la technique de programmation en circuit-série. Le SPI du Due à également avancé des fonctionnalités qui peuvent être utilisées avec les méthodes SPI étendues pour Due. [13]

- **CAN : CANRX et CANTX**

  Ces broches prennent en charge le protocole de communication CAN mais ne sont pas encore prises en charge par les API Arduino.

- **DEL "L" : 13**

  Il existe une LED intégrée connectée à la broche numérique 13. Lorsque la broche est ÉLEVÉE, la LED est allumée, lorsque la broche est FAIBLE, elle est éteinte. Il est également possible de réduire la LED car la broche numérique 13 est également une sortie PWM.

- **TWI 1 : 20 (SDA) et 21 (SCL)**

- **TWI 2 : SDA1 et SCL1.**

Prise en charge de la communication TWI à l'aide de la bibliothèque Wire. SDA1 et SCL1 peuvent être contrôlés à l'aide de la classe Wire1 fournie par la bibliothèque Wire. Alors que SDA et SCL ont des résistances internes pull up, SDA1 et SCL1 n'ont pas. L'ajout de deux résistances pull up sur les lignes SDA1 et SCL1 est requis pour l'utilisation de Wire1.

- **Entrées analogiques : broches de A0 à A11**

The Due à 12 entrées analogiques, chacune pouvant fournir 12 bits de résolution (c'est-à-dire 4096 valeurs différentes). Par défaut, la résolution des lectures est définie sur 10 bits, pour la compatibilité avec d'autres cartes Arduino. Il est possible de modifier la résolution de l'ADC avec analogReadResolution (). Les broches d'entrée analogiques du Due mesurent du sol à une valeur maximale de 3.3V. L'application de plus de 3.3V sur les broches de Due endommagera la puce SAM3X. La fonction analogReference () est ignorée sur Due.

La broche AREF est connectée à la broche de référence analogique SAM3X à travers un pont de résistance. Pour utiliser la broche AREF, la résistance BR1 doit être désolée de la PCB.

- **DAC1 et DAC**

Ces broches fournissent des sorties analogiques vraies avec une résolution de 12 bits (4096 niveaux) avec la fonction analogWrite (). Ces broches peuvent être utilisées pour créer une sortie audio à l'aide de la bibliothèque audio.

Veuillez noter que la gamme de sortie du DAC est en fait de 0,55 V à 2,75 V uniquement.

Autres broches sur la carte :

- **AREF**

Tension de référence pour les entrées analogiques. Utilisé avec analogReference ().

- **Réinitialiser**

  Apportez cette ligne LOW pour réinitialiser le microcontrôleur. Habituellement utilisé pour ajouter un bouton de réinitialisation aux blindages qui bloquent celui sur la carte.

3.7. Communication

L'Arduino Due dispose d'un certain nombre d'installations pour communiquer avec un ordinateur, un autre Arduino ou d'autres microcontrôleurs, et différents appareils comme les téléphones, les tablettes, les caméras, etc. Le SAM3X fournit un UART matériel et trois interfaces matérielles USART pour la communication en série TTL (3.3V).

Le port de programmation est connecté à un ATmega16U2, qui fournit un port COM virtuel sur un ordinateur connecté (Pour reconnaître le périphérique, les machines Windows auront besoin d'un fichier .inf, mais les machines OSX et Linux reconnaîtront automatiquement la carte en tant que port COM). Le 16U2 est également connecté à l'UART matériel SAM3X. Serial on pins RX0 et TX0 fournissent une communication Serial-à-USB pour la programmation de la carte via le microcontrôleur ATmega16U2. Le logiciel Arduino comprend un moniteur série qui permet d'envoyer des données textuelles simples vers et depuis le la carte. Les voyants RX et TX de la carte clignotent lorsque des données sont transmises via la puce ATmega16U2 et la connexion USB à l'ordinateur (mais pas pour la communication série sur les broches 0 et 1).

Le port USB natif est connecté au SAM3X. Il permet la communication en série (CDC) via USB. Cela fournit une connexion série au Serial Monitor ou à d'autres applications sur votre ordinateur. Il permet également à Due d'imiter une souris USB ou un clavier sur un ordinateur connecté.

Le port USB natif peut également servir d'hôte USB pour les périphériques connectés tels que les souris, les claviers et les smartphones.

Le SAM3X prend également en charge la communication TWI et SPI. Le logiciel Arduino comprend une bibliothèque Wire pour simplifier l'utilisation du bus TWI. [13].

3.8. Programmation

Le Due peut être programmé avec le logiciel Arduino (IDE), Il suffit de sélectionner "Arduino Due" dans le menu Outils >Type de carte.

Le fait de télécharger des croquis sur le SAM3X est différent des microcontrôleurs AVR trouvés dans d'autres cartes Arduino car la mémoire flash doit être effacée avant d'être programmée. Le téléchargement sur la puce est géré par ROM sur le SAM3X, qui est exécuté uniquement lorsque la mémoire flash de la puce est vide. [13]

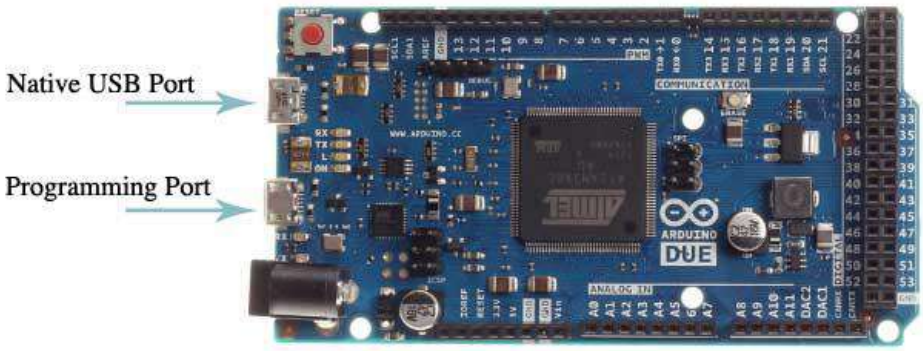


Figure 2.4 : la carte Arduino due.

L'un ou l'autre des ports USB peut être utilisé pour la programmation de la carte, bien qu'il soit recommandé d'utiliser le port de programmation en raison de la façon dont l'effacement de la puce est traité :

- **Port de programmation :** pour utiliser ce port, sélectionnez "Arduino Due (Port de programmation)" comme carte dans l'IDE d'Arduino. Connectez le port de programmation de Due (le plus proche de la prise de courant continu) à votre ordinateur. Le port de programmation utilise le 16U2 comme une puce USB-à-série connectée au premier UART du SAM3X (RX0 et TX0). Le 16U2 comporte deux broches connectées aux broches Reset et Errasse du SAM3X. L'ouverture et la fermeture du port de programmation connectée à 1200bps déclenchent une procédure de "effacement dur" de la puce SAM3X, en activant les broches d'effacement et de réinitialisation sur le SAM3X avant de communiquer avec

l'UART. C'est le port recommandé pour la programmation du débit. Il est plus fiable que le "soft effacement" qui se produit sur le port natif, et il devrait fonctionner même si le MCU principal s'est écrasé.

- **Port natif :** Pour utiliser ce port, sélectionnez "Arduino Due (Port USB natif)" que votre carte dans l'IDE Arduino. Le port USB natif est directement connecté au SAM3X. Connectez le port USB natif de Due (celui le plus proche du bouton de réinitialisation) sur votre ordinateur. L'ouverture et la fermeture du port Native à 1200bps déclenchent une procédure d'effacement doux : la mémoire flash est effacée et la carte est redémarrée avec le chargeur de démarrage. Si le MCU s'est écrasé pour une raison quelconque, il est probable que la procédure d'effacement doux ne fonctionnera pas car cette procédure se passe entièrement dans un logiciel sur le SAM3X. L'ouverture et la fermeture du port natif à une baud rate différente ne réinitialiseront pas le SAM3X.

3.9. Protection contre les surintensités USB

L'Arduino Due à un poly fuse réinitialisant qui protège les ports USB de votre ordinateur contre les court-circuit et la surintensité. Bien que la plupart des ordinateurs fournissent leur propre protection interne, le fusible fournit une couche supplémentaire de protection. Si plus de 500 mA est appliqué au port USB, le fusible brise automatiquement la connexion jusqu'à ce que la surcharge ou la surcharge soit supprimée. [13]

3.10. Caractéristiques physiques et compatibilité des boucliers

La longueur et la largeur maximales de la PCB Arduino Due sont respectivement de 4 et 2.1 pouces, les connecteurs USB et la prise de courant s'étendant au-delà de la première dimension. Trois trous de vis permettent d'attacher la carte à une surface ou à un boîtier. Notez que la distance entre les broches numériques 7 et 8 est de 160 mille (0,16 "), pas un multiple même de l'espacement de 100 mm des autres broches.

L'Arduino Due est conçu pour être compatible avec la plupart des boucliers conçus pour l'Uno, Diecimila ou Duemilanove. Les broches numériques 0 à 13 (et les broches AREF et GND

adjacentes), les entrées analogiques 0 à 5, l'en-tête de puissance et l'en-tête "ICSP" (SPI) sont tous dans des emplacements équivalents. En outre, l'UART principal (port série) se trouve sur les mêmes broches (0 et 1). [13]

3.11. Synthèse

Grâce à ce que nous avons présenté une description détaillée de la carte Arduino, identifier les principales caractéristiques qui nous ont fait choisir ce dernier. Cette partie se veut une synthèse pour que les lecteurs intéressés par cette carte puissent avoir une prise en main rapide et développer rapidement leurs applications.

Pour nous, nous avons choisie cette carte embarquée pour interfacer l'ensemble capteur / actionneur au Mini-PC Raspberry. La carte Arduino associée à d'autres extensions (Shield). Selon ce dernier on peut contrôler la vitesse la rotation et le sens, tout ça pour assurée le bon fonctionnement de notre robot. L'utilisation de l'Arduino rendra notre architecture matérielle évolutive dans les sens ou peut rajouter ou retirer aisément un ou plusieurs capteurs sans que cela n'implique un développement couteux en temps et en énergie. C'est l'atout majeur de ce type de système.

D'autre part, nous pouvons relier la carte Arduino de diverses méthodes (USB, I2C, ...) à un PC ou à la carte Raspberry Pi, C'est cette dernière carte que nous avons choisi d'intégrer à notre architecture matérielle. Dans la deuxième partie de ce chapitre, nous vous décrirons en détaille la constitution et l'utilisation (programmation) de ce mini-PC.

4. La carte Raspberry Pi

Nous présentons dans la suite, le Raspberry pi 2 Model B que l'on a utilisé dans notre projet. Ce dernier est considéré comme le cerveau de notre système parce qu'il va gérer les informations issues de la caméra, les traiter et calculer les lois de commandes nécessaires aux mouvements du robot mobile.

Le Raspberry Pi 2 Model B est un petit ordinateur (Mini-PC) qui fonctionne sous le système d'exploitation Linux sur carte SD destiné à des applications d'informatique embarquée.

Le cœur de l'ordinateur est un FPGA (Broadcom 2836) intégrant un processeur ARMv7 cadencé à 900MHz et de nombreux périphériques.

- **Pourquoi Pi ?**

  • C'est économique.

  • Très facile à utiliser

  • C'est un code libre basé sur Linux avec un accès aux logiciels et téléchargements gratuits pour vous aider avec votre projet.

  • Il existe une grande communauté Pi.

  • Excellente connectivité

  • Flexible pour l'utilisateur. Vous voulez jouer à des jeux ? Diffuser de la musique et/ou des vidéos ? Travailler ? Créer une station météorologique ? Créer une caméra de sécurité domestique ? Aucune limite.

- **Pourquoi Pi 2 B ?**

  • Six fois la puissance de traitement des précédents modèles Pi

  • Démarrage plus rapide

  • Une meilleure compatibilité avec les derniers programmes et logiciels éducatifs

  • Le processeur ARM Cortex-A7 32 bits prend en charge toutes les dernières distributions GNU/LINUX ARM, telles que Snappy Ubuntu, et Microsoft Windows 10.

  • Le système sur circuit (SoC) BCM2836 fonctionne à 900 MHz, contre 700 MHz pour le PI B+ BCM2835

4.1. La carte Raspberry Pi 2 model B

Ce dernier Model est le Raspberry Pi de deuxième génération. Il a remplacé l'original Raspberry Pi 1 Model B + en février 2015 qui a été utilisé par nos collègues de l'année dernière. [10] Par rapport au Raspberry Pi 1, il a :

- Un processeur quad-core ARM Cortex-A7 de 900 MHz
- 1 Go de RAM

Comme le (Pi 1) Modèle B +, il a aussi :

- 4 ports USB
- 40 broches GPIO
- Port HDMI complet
- Port Ethernet
- Prise audio combinée de 3,5 mm et vidéo composite
- Interface caméra (CSI)
- Interface d'affichage (DSI)
- Emplacement pour carte Micro SD
- VideoCore IV noyau graphique 3D

Parce qu'il dispose d'un processeur ARMv7, il peut exécuter la gamme complète de distributions ARM GNU / Linux, y compris Snappy Ubuntu Core, ainsi que Microsoft Windows 10.

Le Raspberry Pi 2 a un facteur de forme identique au Modèle B + antérieur (Pi 1) et a une compatibilité complète avec Raspberry Pi 1. [10]

4.2. Les composants standards de Raspberry

Afin de bien visualiser ce qu'il est possible d'adjoindre à la carte Raspberry Pi nue, la figure 2.5 suivante représente tout à la fois son organisation générale et les différents connecteurs disponible d'un Raspberry standard :

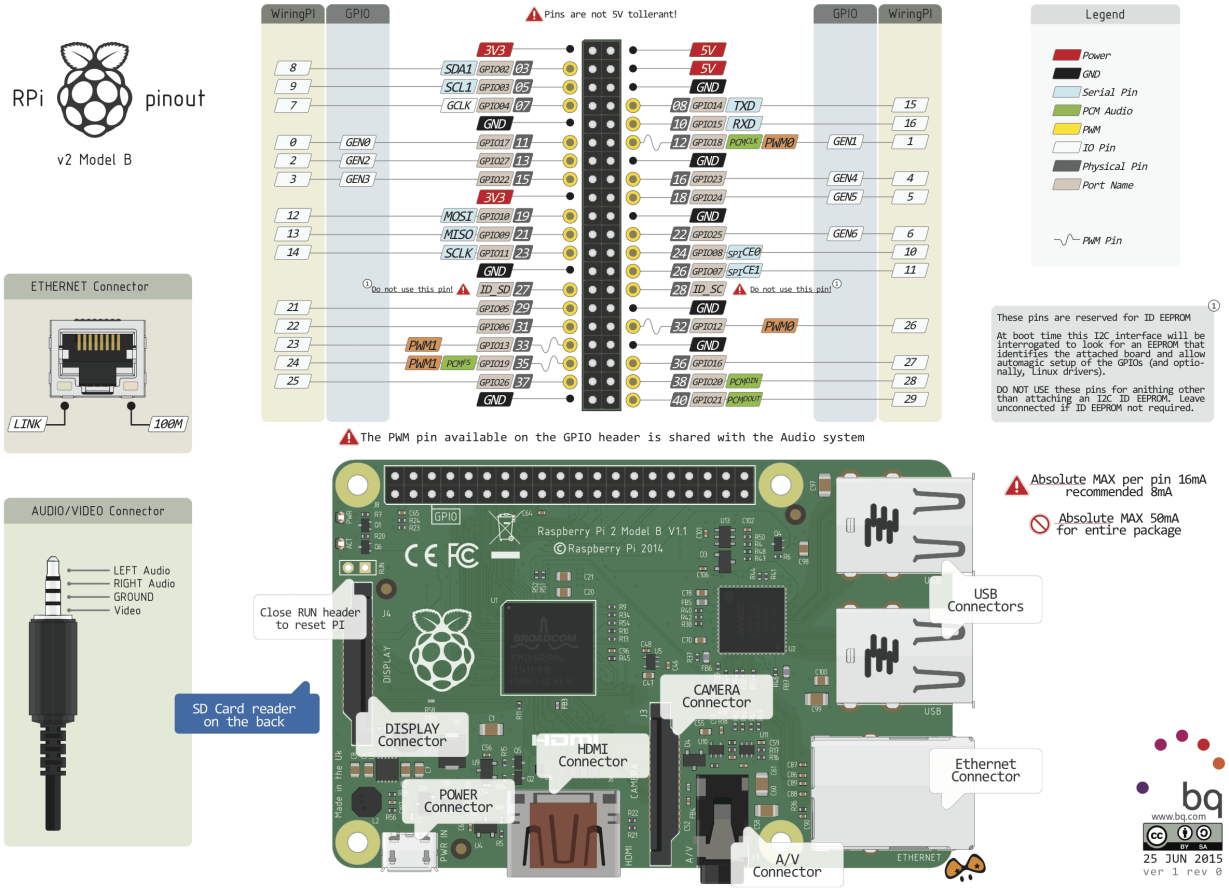


Figure 2.5 : Les composants standards d'un raspberry.

Le centre de la carte est occupé par le processeur ou, plus exactement, par un empilement du processeur et de sa mémoire qui sont littéralement montés l'un sur l'autre. [12]

La connectique qui occupe la majorité de la surface de la carte et non les composants actifs, Dans ce qui va suivre nous détaillerons les différents de la carte Raspberry pi 2 Model B :

- Connecteurs à câble plat flexible (FFC) à la Raspberry Pi 2B :

  - **<u>Connecteur d'affichage</u>,** près du connecteur micro-USB, est l'interface série d'affichage (DSI). Il permet une interface de bas niveau avec les écrans LCD et autres écrans avec Raspberry Pi. Il s'agit d'un connecteur plat flexible monté en surface à 15 broches, offrant deux voies de données, une voie d'horloge, 3.3V et GND.

- **<u>Connecteur de caméra</u>**, Situé entre le connecteur LAN et HDMI, est le connecteur d'interface MIPI Camera Serial Interface 2 (CSI-2) pour les modules de caméra. Il s'agit d'un connecteur flexible plat à 15 broches monté en surface, fournissant deux lignes de données, une voie d'horloge, une interface de commande bidirectionnelle compatible avec I2C, 3.3V et GND.

- La prise réseau accepte une prise normalisée RJ45 et tout réseau Ethernet Classique à 10 Mbits ou 100 Mbits par seconde.

- Quatre connecteur USB voisin et quant à lui conforme à la norme USB 2.0, rétro-compatible USB 2.0 comme il se doit.

- HDMI :(Hight Defenition Multimedia Interface) permet de relier le Raspberry Pi a un dispositif (écran LCD ou un vidéo projecteur), capable de surcroît de fournir de la vidéo au format full HD de 1080p, et véhicule également le son.

- Un slot les cartes SD : le Raspberry a besoin d'une mémoire externe supplémentaire pour fonctionner. Ce slot permet de connecter la mémoire externe.

- Prise jack audio/vidéo : Prise audio combinée de 3,5 mm et vidéo composite.

- GPIO : (General Prupose Input/Output) sont des ports d'Entres/sorties c'est-à-dire de toutes les interfaces externes proposées par le microprocesseur qui équipe le Raspberry.

4.3. Les GPIO

- **GPIO simples :** entrées /sorties numériques : interrupteurs ON/OFF numériques (manette de jeu, bouton-poussoir, allumer-éteindre lumière ou moteur, etc.)

- **PWM :** « Modulation de la largeur des pics » signaux carres largeur variable : commande moteur pas à pas, variateur lumière LED, variateur vitesse moteur continu, Il y a un seul port GPIO configurable en PWM hardware sur Raspberry Pi, ce dernier est nécessaire pour piloter les servomoteurs (temps de réponse courts pour encodeur en plus régularité précise).
  Pour contourner ce problème : la bibliothèque WiringPI permet d'ajouter autant de PWM que souhaite par soft (valable pour périphériques autres que servomoteurs).

- **UART :** échange informations série asynchrone (pas d'horloge). Dialogue avec PC Windows(Putty).

- **I2C :** échange informations série synchrone (horloge). Tous les récepteurs au même niveau. → Automobile, matériel hifi.

- **SPI :** échange informations série synchrone (horloge). Récepteur maitre/esclave, maitre tournant possible. → Pilotage écrans LCD.

Toutes ces choses nous allons les exploiter dans les parties suivantes pour relier notre Raspberry au monde extérieur et lui permettre ainsi de commander et de s'interfacer avec la carte Arduino.

Autrement, notre carte dispose de connecteur micro USB qui ne véhicule pas d'interface USB comme on aurait pu le penser, mais ce connecteur sert uniquement à alimenter le Raspberry Pi.

Par ailleurs, le Raspberry Pi ne comporte pas de convertisseur analogique/numérique ADC/DAC. Il est impossible de lire directement les signaux des capteurs analogiques. C'est pour cela que l'utilisation de la carte Arduino vient palier cette limitation.

4.4. Mise en marche d'un Raspberry Pi 2

Pour faire fonctionner notre Raspberry Pi, nous avons besoin des accessoire suivants :

- **Une carte mémoire SD**

Cette carte va contenir le système d'exploitation du Raspberry Pi, il ne pourra donc pas démarrer sans cette dernière. La fondation recommande une carte d'au moins 4 Go et de classe 4 minimum. [12]

- **Alimentation micro USB externe**

L'alimentation peut être la cause de nombreux problèmes sur lesquels nous aurons l'occasion de revenir à plusieurs reprises. Le Raspberry Pi ne peut pas être alimenté par le port USB d'un ordinateur ou d'une télévision, il faut par conséquent utiliser une alimentation USB dédiée.

Ce dernier s'alimente sous une tension unique de 5 volts ; tension sur laquelle il peut consommer jusqu'à 750 mA selon les taches qu'il exécute. Cette alimentation doit être normalement fournie via le connecteur micro USB placé dans un angle de la carte. [12]

- **Un clavier et une souris USB**

En cas général, pour faire fonctionner notre Raspberry Pi nous avons besoin d'un clavier et une souris, rien de bien particulier n'est à signaler en ce qui les concerne. N'importe quel clavier USB et n'importe quelle souris USB conviennent au Raspberry Pi et se connectent directement dans les deux ports USB prévus à cet effet sur le modèle que nous allons utiliser le modèle B.

Mais attention car certains claviers ou certaines souris peuvent être de gros consommateurs d'énergie, il sera donc nécessaire dans ce cas de les brancher sur un HUB USB alimenté par une alimentation externe. C'est souvent le cas par exemple des claviers et des souris sans fils. La souris n'est pas indispensable mais elle devient très utile si on passe du mode texte au mode graphique. [12]

- **Un câble HDMI**

Par défaut, le Raspberry Pi délivre des signaux à la norme HDMI 1.4 comportant vidéo et son. Il peut donc être raccordé directement à tout moniteur informatique ou à tout téléviseur à écran plat disposant d'une telle entrée.

Si l'écran choisi est équipé de haut-parleurs, ces derniers reproduiront le son fourni par le Raspberry Pi puisque celui-ci est véhiculé par la liaison HDMI. C'est bien sur les cas des téléviseurs à écran plat mais pas nécessairement celui des moniteurs informatiques.

Dans une telle situation, nous pourrons alors récupérer le son depuis la prise jack stéréo de 3,5 mm que nous avons vu.

En fonction du type d'écran que nous allons utiliser, nous devons nous procurer le câble correspondant. L'usage de la sortie HDMI est bien entendu recommandé, l'image sera de meilleure qualité et le son sera intégré. [12]

4.5. Le réseau

En effet, tous les logiciels susceptibles d'être utilisés avec le Raspberry Pi sont téléchargeables sur internet ou doivent être mis à jour par ce biais, ce qui rend bien vite cet accès indispensable.

Fort heureusement, si vous disposez d'un réseau filaire, sa connexion au Raspberry Pi est immédiate puisqu'il suffit de faire appel à n'importe quel câble RJ45 standard. L'interface réseau du Raspberry Pi est évidemment de type auto MDI et accepte donc indifféremment les câbles réseau droits ou croisés.

Le Raspberry Pi n'intègre pas en standard un module WIFI donc il faudra faire l'acquisition d'une clé USB-WIFI si nous voulons l'utiliser. [12]

4.6. Préparation de la carte SD

Le Raspberry Pi ne dispose en standard d'aucune mémoire de masse comme nous avons déjà indiquée. Ce rôle est donc dévolu à une carte mémoire de type SD qu'il faut enficher dans le connecteur prévu dans la Figure 2.5 Cette carte mémoire, de type SD ou SD HC, peut avoir toute taille comprise entre 2 Go et 32 Go. Comme elle supporte le système d'exploitation ainsi que les éventuels logiciels supplémentaires que nous allons utiliser avec le Raspberry Pi comme la librairie Pixy.

En effet, pour des raisons inexplicables, même par son concepteur, et alors que l'interface des cartes SD et SD HC est parfaitement normalisée, toutes ne fonctionnent pas nécessairement avec le Raspberry Pi.

Tout comme le disque dur d'un PC doit être préparé avec le système d'exploitation que l'on souhaite utiliser, la carte mémoire destinée au Raspberry Pi doit également être initialisée avec le système d'exploitation choisi.

Malgré que nous pouvons utiliser Windows 10 dans le Raspberry Pi 2B, mais on a gardé Linux comme monde du logiciel libre. Plusieurs versions (on dit habituellement « distribution ») ont été recompilées pour le Raspberry Pi et c'est donc avec ce système d'exploitation que vous allez devoir l'utiliser. [12]

4.7. Système d'exploitation

La plupart des systèmes qui fonctionnent sur Raspberry Pi sont des versions du système d'exploitation Linux. Parce que Linux est open source, les développeurs peuvent l'adopter pour des buts spécifiques. Dans le cas du Raspberry Pi, le matériel léger nécessite un système réduit et privé d'applications inutiles.

La distribution est recommandée Raspbian. C'est un système d'exploitation libre basé sur Debian optimisé pour le matériel Raspberry Pi. Cependant Raspbian fournit plus d'un simple système d'exploitation : il est livré avec plus de 35.000 paquets, des logiciels précompilés qui nous facilitent le développement soft [12].

Tous langages qui peuvent être compilés sur ARMv7 sont installés par défaut sur le Raspberry Pi : Python, C, C++, Java, Scratch et Ruby. La fondation Raspberry Pi recommande Python comme langage pour les étudiants.

4.8. Programmation

Il existe plusieurs langages de programmation qui peut être compilé sur le Raspberry Pi, telle que le langage C, C++, HTML5, le Java, le JavaScript...etc. Le langage utilisant dans notre projet et le Python.

Le Python est un langage de programmation inventé par Guido van Rossum, la première version de python est sortie en 1991.

C’est un langage de programmation interprété, c'est à dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter.

Python étant un langage très riche, surtout si l’on prend en compte les nombreuses Bibliothèques complémentaires disponibles, Ce dernier est à la fois simple et puissant, c’est un langage facile à apprendre et son code est plus lisible, il est donc plus facile à maintenir. Il est parfois jusqu'à 5 fois plus concis que le langage Java par exemple, ce qui augmente la productivité du développeur et réduit mécaniquement le nombre de bugs. [12]

4.9. Synthèse

La haute performance de l’Arduino due et de Raspberry Pi 2B Insuffisantes pour atteindre le bon performance à notre Robot, Le Raspberry va se charger de la partie acquisition, traitement d’images et calcul des lois de commandes qu’il va transmettre à la carte Arduino, tous ces processus font fonctionner lentement et cela affecte de manière significative à la vitesse de traitement.

Pour résoudre ce problème et réduire la pression sur le Raspberry Ce qui est causée par le traitement d’images. parmi les solutions l’utilisation du Pixy Cam cette caméra possède un processeur graphique qui sera fait le traitement d’image au lieu de faire ça dans le Raspberry et on a gardé comme cerveau pour calcul des lois des commandes .

Dans la partie suivante nous allons présenter la Cam Pixy qui nous allons utiliser dans notre projet.

5. La caméra Pixy

5.1. Aperçu

- Système de vision petit, rapide, facile à utiliser, peu coûteux et facilement accessible.
- Apprends à détecter des objets que vous l'enseignez.
- Indique ce qu'il détecte 50 fois par seconde.
- Se connecte à Arduino avec un câble inclus. Fonctionne également avec Raspberry Pi, BeagleBone et des contrôleurs similaires.
- Toutes les bibliothèques pour Arduino, Raspberry Pi, etc. sont fournies.
- C / C ++ et Python sont pris en charge.
- Communique via l'une des nombreuses interfaces : SPI, I2C, UART, USB ou sortie analogique / numérique.
- L'utilitaire de configuration s'exécute sous Windows, MacOs et Linux.
- Tout logiciel / firmware est open-source GNU-licence.
- Toute la documentation sur le matériel, y compris les schémas, la liste de matériaux, la mise en page des PCB, etc. [14]



Figure 2.6 : Pixy CMUcam5

5.2. Bref historique

Pixy (CMUcam5) est un partenariat entre Carnegie Mellon Robotics Institute et Charmed Labs. Pixy vient d'une longue ligne de CMUcams, mais Pixy a vraiment démarré comme une campagne de Kick Starter. Il a d'abord commencé à être livré en mars 2014, mais c'est déjà devenu le système de vision le plus populaire de l'histoire. [14]

5.3. Simplicité de mise en œuvre du système de vision

Si vous souhaitez que votre robot effectue une tâche, comme la collecte d'un objet, la poursuite d'une balle, la localisation d'une station de chargement, etc., et que vous souhaitez qu'un seul capteur aide à accomplir toutes ces tâches, la vision est votre capteur. Les capteurs Vision (image) sont utiles car ils sont si flexibles. Avec le bon algorithme, un capteur d'image peut détecter ou détecter pratiquement n'importe quoi. Mais il existe deux inconvénients avec les capteurs d'image : 1) ils sortent beaucoup de données, des dizaines de mégaoctets par seconde, et 2) le traitement de cette quantité de données peut submerger de nombreux processeurs. Et si le processeur peut suivre les données, une grande partie de sa puissance de Traitement ne sera pas disponible pour d'autres tâches.

Pixy résout ces problèmes en associant un puissant processeur dédié au capteur d'image. Pixy traite les images du capteur d'image et envoie uniquement les informations utiles (par exemple, le dinosaure violet détecté à x = 54, y = 103) à votre microcontrôleur. Et cela se fait à cadence (50 Hz). L'information est disponible via l'une des nombreuses interfaces : UART série, SPI, I2C, USB ou sortie numérique / analogique. Ainsi, votre Arduino ou un autre microcontrôleur peut facilement parler avec Pixy et dispose de nombreuses CPU disponibles pour d'autres tâches.

Il est possible de brancher plusieurs Pixys sur votre microcontrôleur - par exemple, un robot avec 4 Pixys et 360 degrés de détection. Ou utiliser Pixy sans microcontrôleur et utiliser les sorties numériques ou analogiques pour déclencher des événements, des commutateurs, des servos, etc. [14]

5.4. Support du contrôleur

Pixy peut facilement se connecter à de nombreux contrôleurs différents car il prend en charge plusieurs options d'interface (UART série, SPI, I2C, USB ou sortie numérique / analogique), mais Pixy a commencé sa vie à parler à Arduino. Au cours des derniers mois, nous avons ajouté un support pour Arduino Due, Raspberry Pi et BeagleBone Black. Des bibliothèques de logiciels sont fournies pour toutes ces plates-formes afin que vous puissiez vous lancer rapidement. En outre, nous avons ajouté une API Python si vous utilisez un contrôleur basé sur Linux (par exemple, Raspberry Pi, BeagleBone). [14]

5.5. La détection d'objets

Pixy utilise un algorithme de filtrage basé sur les couleurs pour détecter les objets. Les méthodes de filtrage basées sur les couleurs sont populaires car elles sont rapides, efficaces et relativement robustes. La plupart d'entre nous connaissent le RGB (rouge, vert et bleu) pour représenter les couleurs. Pixy calcule la couleur (teinte) et la saturation de chaque pixel RGB du capteur d'image et les utilise comme paramètres de filtrage primaires. La teinte d'un objet reste en grande partie inchangée avec les changements d'éclairage et d'exposition. Les changements d'éclairage et d'exposition peuvent avoir un effet frustrant sur les algorithmes de filtrage des couleurs, ce qui leur permet de se casser. L'algorithme de filtrage de Pixy est robuste en matière d'éclairage et de changements d'exposition.

Pixy peut trouver littéralement des centaines d'objets à la fois. Il utilise un algorithme de composants connectés pour déterminer où un objet commence et un autre se termine. Pixy compile les tailles et les emplacements de chaque objet et les signale à travers l'une de ses interfaces (par exemple SPI).

Dans la documentation technique de ce système de vision, nous retrouvons souvent le terme "50 images par seconde" ou "50 frames par seconde". En bref, cela signifie que Pixy est rapide. Pixy traite toute une image 640x400 toutes les 1 / 50ème de seconde (20 millisecondes). Cela signifie que vous obtenez une mise à jour complète de toutes les positions des objets détectés toutes les 20 ms. À ce rythme, il est possible de suivre le parcours de la balle tombante / rebondissant. (Une balle qui se déplace à 30 mph se déplace moins d'un pied en 20 ms.) [14]

5.6. Définition simple de l'objet à suivre

Pixy est unique du fait que l'on peut rapidement et simplement définir l'objet à suivre. Par exemple si cet objet est un dinosaure violet placez le dinosaure devant l'objectif de la caméra Pixy et appuyez sur le seul bouton. Boule orange ? Placez le ballon devant Pixy et appuyez sur le bouton. C'est simple, et c'est rapide.

Plus précisément, vous enseignez à Pixy en tenant l'objet devant son objectif tout en maintenant le bouton situé en haut. En faisant cela, la LED RGB sous l'objectif fournit des commentaires concernant l'objet qu'il regarde directement. Par exemple, la LED devient orange lorsqu'une boule orange est placée directement devant Pixy. Relâchez le bouton et Pixy génère un modèle statistique des couleurs contenues dans l'objet et les stocke en flash. Il utilisera ensuite ce modèle statistique pour trouver des objets avec des signatures de couleurs similaires dans leur cadre à partir de là.

Pixy peut apprendre sept signatures couleur, numérotées 1-7. La signature couleur 1 est la signature par défaut. Pour enseigner à Pixy, les autres signatures (2-7) nécessitent un simple bouton qui appuie sur la séquence. [14]

5.7. L'application PixyMon

PixyMon est une application qui fonctionne sous Windows, MacOs et Linux. Il vous permet de voir ce que Pixy voit, en tant que vidéo brute ou transformée. Il vous permet également de configurer votre Pixy, de définir le port de sortie et de gérer les signatures des couleurs. PixyMon communique avec Pixy sur un câble mini USB standard.

PixyMon est idéal pour le débogage de votre application. Vous pouvez brancher un câble USB à l'arrière de Pixy et exécuter PixyMon, puis voir ce que Pixy voit lorsqu'il est accroché à votre Arduino ou à un autre microcontrôleur - pas besoin de débrancher quoi que ce soit. PixyMon est open source, comme tout le reste. [14]

5.8. Spécifications techniques

La puissance du système de vision Pixy est l'intégration sur une seule carte à la fois du système optique de vision est des composants électroniques (processeur, RAM, …) nécessaires à l'acquisition et aux traitements des images. [14]

- Processeur: NXP LPC4330, 204 MHz, dual core
- Capteur d'image: Omnivision OV9715, 1/4 ", 1280x800
- Lens field-of-view: 75 degrés horizontal, 47 degrés vertical
- Type d'objectif: standard M12 (plusieurs types différents disponibles)
- Consommation électrique: 140 mA typique
- Entrée d'alimentation: entrée USB (5V) ou entrée non régulée (6V à 10V)
- RAM: 264K octets
- Flash: 1M octets
- Sorties de données disponibles: UART série, SPI, I2C, USB, numérique, analogique
- Dimensions: 2.1 "x 2.0" x 1.4
- Poids: 27 grammes

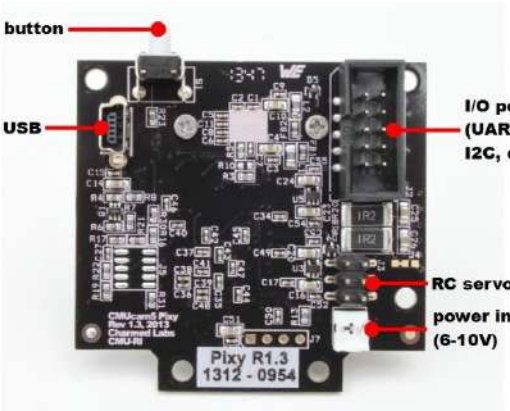


Figure 2.7 : structure de la Cam Pixy CMUcam5.

6. Conclusion

Ce chapitre a été pour nous l'occasion de présenter les deux cartes Raspberry PI et Arduino ainsi que le système de vision Pixy que nous avons utilisé pour concevoir l'architecture matérielle nécessaire à la commande du robot mobile.

L'asservissement de ce robot basé sur l'utilisation des données visuelles, nécessite d'une part une puissance de calcule importante et d'autre part des calculateurs peu volumineux et léger pouvant être embarqués sur la structure mécanique du robot mobile.

La solution que nous vous présenterons dans le dernier chapitre répondra en tous points au cahier des charges que nous nous somme imposé. Mais avant de rentrer dans ces détails de conception, nous allons, dans le prochain chapitre, vous présenter les aspects théoriques de l'asservissement visuel de robot mobile.

Chapitre 3 : Asservissement Visuel

1. Introduction

Dans ce chapitre, nous allons introduire les techniques d'asservissement visuel. Après un bref historique, nous décrirons en détail les différentes architectures existantes des systèmes asservis par vision. Ce descriptif sera suivi d'une présentation des différentes classifications des systèmes d'asservissement visuel selon des déférents critères tel que la position de capteur visuel, la nature des grandeurs asservies et le type de commande. Finalement, nous allons parler d'une modélisation spécifique à l'asservissement visuel à savoir la modélisation à point central unique.

Ce chapitre ainsi conçu nous permettra de situer et de justifier la technique de commande que nous avons utilisé pour asservir notre robot mobile.

2. Historique de l'asservissement visuel :

Historiquement, les premières expérimentations combinaient les informations visuelles aux mouvements d'un robot dans une approche en boucle ouverte présentant successivement les opérations « regarde » (looking), puis « bouge » (moving).

La précision du système ainsi réalisé est liée à la précision du capteur visuel, à celle du robot, ainsi qu'à celle de sa commande. Pour améliorer la précision de l'ensemble, il est alors nécessaire d'affiner autant que possible la précision des trois briques qui le composent. Une alternative à cette escalade dans la recherche qualitative des sous-éléments du système consiste à créer une boucle de commande par retour, traditionnelle en automatique, asservie à partir d'informations issues du capteur de vision (visual-feedback).

Le terme d'asservissement visuel, « visual servoing » en anglais, fut introduit pour la première fois par Hill et Park en 1979 [15] pour distinguer leur approche d'expérimentations antérieures dites « blocks world » dans lesquelles les systèmes alternaient les étapes « prises d'images » puis « bouge ».

L'asservissement visuel est le résultat de la fusion de différents domaines spécialisés tels le traitement d'images, la cinématique, la dynamique, la théorie du contrôle, ou encore l'informatique temps réel. C'est un domaine qui a beaucoup en commun avec la vision active et la « structure from motion » (reconstruction 3D à partir du mouvement caméra). Les premiers travaux rapportés sur l'asservissement visuel remontent à 1980. Les progrès dans l'art de commander un robot par la vision ont été laborieux, cependant, depuis quelques années, le

domaine a connu un regain d'intérêt, comme le prouve l'abondante littérature qui y est consacrée. [12]

Le positionnement du capteur visuel

- La configuration du capteur embarqué : Comme l'expression l'indique, le capteur visuel est monté sur l'organe terminal du robot. Cette configuration est dénommée "eye in hand" dans la littérature anglo-saxonne. L'objectif de l'asservissement est généralement de déplacer le capteur visuel pour atteindre ou suivre par exemple un objet en mouvement.

- La configuration du capteur déporté : Pour cette configuration "eye to hand", le capteur visuel n'est pas solidaire du robot asservi. L'objectif est de faire évoluer le robot par rapport à son environnement de travail, en percevant généralement à l'aide du capteur visuel l'image du robot et des objets avec lesquels il interagit.

Quelle que soit la configuration du capteur, il s'agit de sélectionner au mieux un ensemble de mesures, permettant de contrôler les degrés de liberté souhaités, et d'élaborer une loi de commande afin que ces mesures atteignent une valeur désirée ou suivent une trajectoire spécifiée. Le principe de la commande se ramène ensuite à réguler (c'est-à-dire amener et maintenir à zéro) le vecteur d'erreur entre mesure et consigne.

Avec un capteur de vision, fournissant à la base des informations 2D, la nature des mesures potentielles est extrêmement riche, puisque l'on peut considérer en asservissement visuel aussi bien des mesures 2D, telles que les coordonnées de points caractéristiques dans l'image par exemple, que des mesures 3D, fournies par un module de localisation exploitant les informations 2D extraites. De cette richesse provient la difficulté majeure de l'asservissement visuel, à savoir, parmi l'ensemble des informations potentielles, comment sélectionner, construire et combiner celles qui fourniront un comportement satisfaisant au système. [12]

3. Les systèmes de vision

En robotique les systèmes de vision sont basés sur l'utilisation d'une caméra CCD (Charge Coupled Device). L'utilisation d'une seule camera ne peut fournir qu'une seule information 2D qu'on appelle vision monoculaire. Pour obtenir des informations 3D, il existe différentes techniques qui sont généralement liées à l'adjonction d'un autre capteur ; par exemple : l'utilisation de deux caméras nous permet de reconstruire la structure 3D des objets avec deux images, on l'appelle : la stéréovision. On peut aussi ajouter un système de réflexion de type miroir donnant ainsi la vision omnidirectionnelle. Donc, on peut dire qu'on peut identifier trois techniques : la vision monoculaire, la stéréovision (binoculaire) et la vision omnidirectionnelle. [16]

3.1. La Vision Monoculaire

Dans le cadre de cette thèse, nous nous intéressons à la vision monoculaire qui utilise une seule caméra qui observe directement la scène. L'inconvénient de ce système est : son champ visuel est limité et ne permet pas de restituer la profondeur de la scène. Il ne fournit qu'une seule représentation bidimensionnelle de l'image et de l'espace tridimensionnel, ce qui n'est pas suffisant. La profondeur est déterminée à partir d'une ou plusieurs images. [16]

Pour contourner cette limitation (en évitant d'introduire une deuxième caméra), nous avons calculé la surface de la cible. Si la valeur de surface grandie, la cible est proche de la caméra et donc du robot et inversement si, la valeur de la surface diminue, la cible s'éloigne du robot. Par conséquence, cette approche nous permettra d'asservir et de suivre la cible quand elle se déplace en perpendiculaire du plan image.

3.2. La Stéréovision

La stéréovision consiste à utiliser deux caméras séparées l'une de l'autre d'une certaine distance connue et observant la même scène figure (3.1). Lorsqu'on utilise deux caméras, c'est le système binoculaire, il est suffisant pour reconstruire en 3D la scène observée ou retrouver l'information de la profondeur par triangulation qui peut être utilisée pour l'évitement d'obstacles ou pour la cartographie.

La principale difficulté de la stéréovision réside dans la mise en correspondance des pixels, c'est-à-dire la détermination dans les images des projections m 1 et m 2 d'un même point M.

Une fois les pixels sont mis en correspondance, les coordonnées du point M sont obtenues par triangulation des points m 1 et m 2 à partir des centres optiques C 1 et C 2 des caméras [16], comme l'illustre la figure (3.2)



Figure 3.1 : capteur stéréovision.

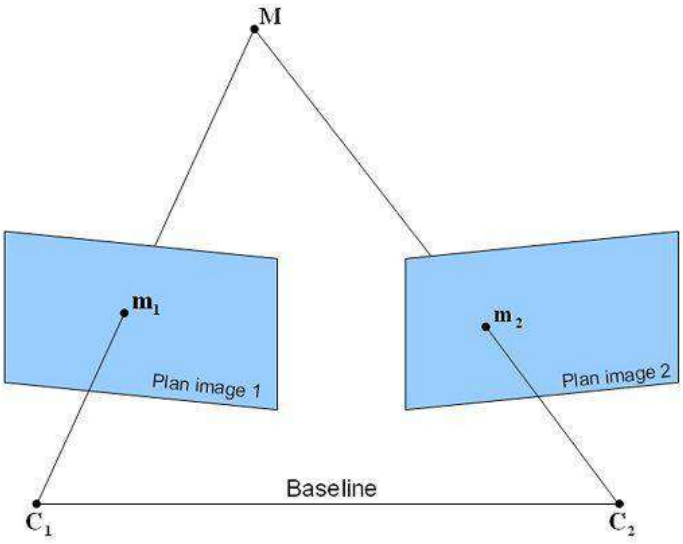
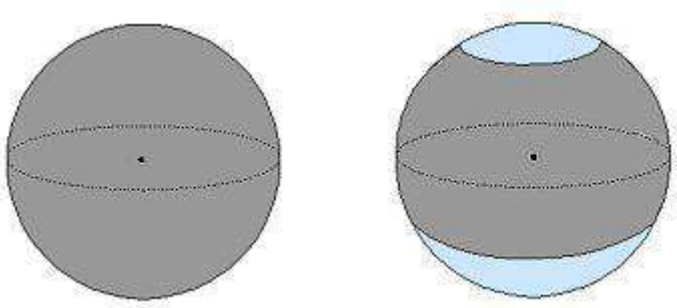


Figure 3.2: Principe de la stéréovision.

3.3. La Vision Omnidirectionnelle

Le terme de la vision omnidirectionnelle se réfère au capteur de vision avec un très large champ de vue, c'est-à-dire un capteur de champ de vue de 360°. Comme il existe un autre terme pour un champ de vue large c'est la vision panoramique. Ces deux termes sont généralement employés indifféremment alors qu'il existe une différence entre eux.

- La vision omnidirectionnelle est le procédé de vision avec un champ de vue de 360° horizontale avec un champ de vision vertical qui peut être limité ou non, donc son champ de vue doit être sphérique, et permet d'observer le monde à partir de son centre, comme l'illustre la figure (3.3 (a)).
- La vision panoramique est une réduction du premier procédé, son champ de vue est limité. Il est de 360° horizontale avec un champ de vue vertical limité. Donc elle ne couvre pas tout l'environnement dans un autre sens comme l'illustre la figure (3.3 (b)).



(a) (b)

Figure 3.3: Champ de vue des systèmes de vision (représenté en gris).
(a) Vision panoramique - (b) Vision omnidirectionnelle

4. Architecture des systèmes asservis par la vision : vers une taxonomie

En 1980, Sanderson et Weiss [12] ont introduit une taxonomie de l'asservissement visuel qui permet de catégoriser les différents systèmes rencontrés. Leur schéma est basé sur les réponses à deux questions :

- La structure de contrôle est-elle hiérarchique, c'est-à-dire avec un système de vision qui fournit directement le jeu de points au contrôleur articulaire du robot, ou alors est-ce le système de vision qui calcule les valeurs des angles des articulations du robot ?
- Le signal d'erreur est-il défini en coordonnées 3D (espace des tâches), ou alors est-il directement exprimé en termes de caractéristiques image 2D ? On parlera alors respectivement d'approche en position ou image.

La taxonomie qui en résulte comporte quatre catégories que nous allons décrire ci-après.
Si l'architecture du système de contrôle est hiérarchique et utilise le système de vision pour fournir le jeu des coordonnées articulaires du robot, en utilisant un retour articulaire pour stabiliser le robot, elle est référencée comme « dynamic look and move ».

Par opposition, les systèmes « visual servo » éliminent le contrôleur robotique au profit d'un contrôle visuel qui calcul les angles articulaires du robot directement à partir du système de vision afin de stabiliser le mécanisme.

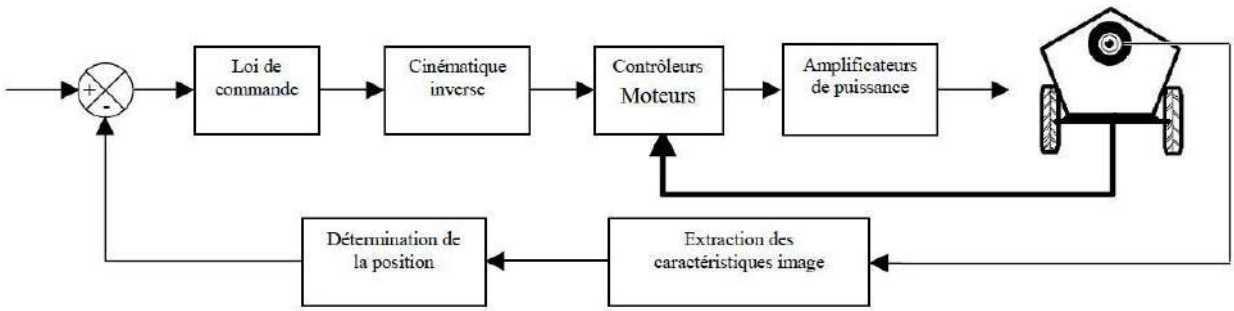


Figure 3.4 : Asservissement visuel avec approche en position de type « dynamic look and move ».

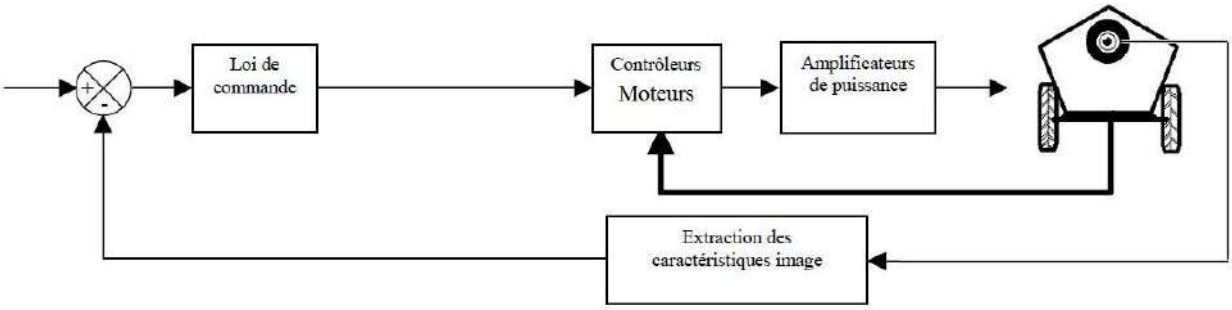


Figure 3.5 : Asservissement visuel avec approche fondée sur l'image de type « dynamic look and move ».

Pour différentes raisons, la plupart des systèmes actuels implémentent le « dynamic look and move » :

- Premièrement, la faiblesse du taux d'échantillonnage du système de vision rend le contrôle direct de l'effecteur du robot ayant une dynamique complexe et non linéaire très difficile. L'emploi d'un retour interne ayant une fréquence d'échantillonnage élevée implique de supposer que le contrôleur visuel possède une dynamique de ses axes idéalisés [12],
- Deuxièmement, la plupart des robots ont une interface qui gère les commandes en position et vitesse dans l'espace cartésien, ce qui simplifie la construction du système de contrôle visuel,
- Troisièmement, l'approche « look and move » isole les singularités cinématiques inhérentes au mécanisme articulaire du contrôleur visuel, ce qui permet de considérer le robot comme un appareil au mouvement purement cartésien.

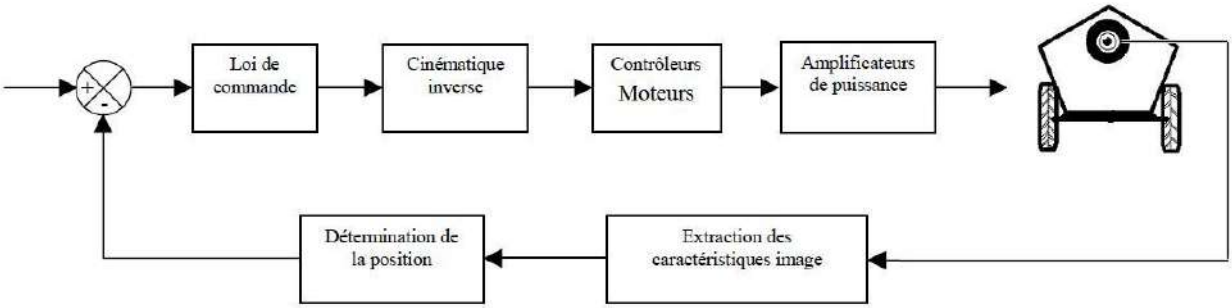


Figure 3.6 : Asservissement visuel avec approche en position.

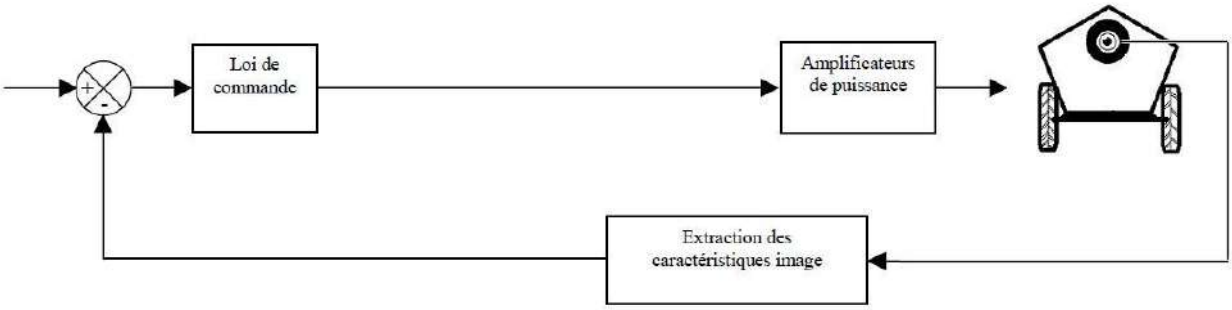


Figure 3.7 : Asservissement visuel avec approche fondée sur l'image.

La deuxième classification des systèmes est liée à la distinction du système de contrôle en position, ou à partir des images :

- Pour les systèmes asservis en position, les caractéristiques intéressantes (point effecteur, point cible, ...) sont extraites des images et utilisées en conjonction avec les modèles géométriques de la cible et de la caméra afin de déterminer la position de la cible relativement à la caméra.

Le retour de la commande est obtenu en réduisant une erreur de position dans l'espace, estimée des positions des éléments de la scène robotique.

- Pour les systèmes asservis dans l'image, les valeurs des commandes sont calculées directement à partir des caractéristiques images, supprimant ainsi l'opération de modélisation requise par l'approche en position.

Les structures de commande référencées « dynamic look and move » font appel à un retour articulaire (Figure 3.4 et Figure 3.5), tandis que les structures d'asservissement visuel avec approche fondée sur la position et l'image (Figure 3.6 et Figure 3.7) s'en affranchissent.
Cette classification réalisée en 1980 par Sanderson et Weiss reste d'actualité dans ces grandes lignes, bien que la discipline ait beaucoup évolué ces dernières années. Nous la complèterons en tenant compte des développements les plus récents.
Pour ce faire, nous allons effectuer une classification en prenant en compte deux critères :

- La configuration caméra-robot (c'est-à-dire le positionnement de la caméra par rapport au robot),
- Le type des grandeurs asservies (la dimension de l'espace de référence, de mesure et de commande).

5. Classification des systèmes d'asservissements visuels

5.1. Selon la position de capteur visuel

Dans la robotique en générale on distingue grossièrement deux types de positionnement selon l'application souhaitée :

- Capteur visuel attaché à l'organe terminal du robot. Ce type de positionnement permet des applications nécessitant un grand volume de travail (surtout si le robot est mobile comme notre cas) et se prête bien à des applications de suivis de cibles.
- Capteur visuel fixe et observant la scène. Ce type de positionnement permet des applications nécessitant un faible volume de travail et se prête bien à la robotique chirurgicale où il est faut observer à la fois l'outil et l'objet.

5.2. Selon la nature des grandeurs asservies

- **Asservissement visuel 3D**

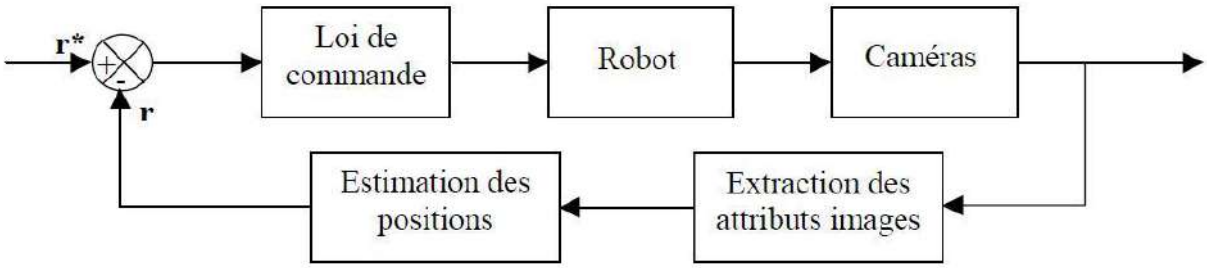


Figure 3.8 : Principe de l'asservissement visuel 3D.

Dans un asservissement visuel 3D, l'objectif est de contrôler le déplacement du robot dans l'espace cartésien. La grandeur à asservir correspond directement à l'attitude d'un repère lié rigidement à l'effecteur du robot par rapport à un repère attaché à l'objet d'intérêt. L'inconvénient majeur de ce type d'asservissement est la nécessité d'une étape de reconstruction 3D permettant de fournir une mesure d'attitude. Cette dernière peut par exemple être obtenue par triangulation en utilisant un système de vision stéréoscopique calibré, ou par des techniques de reconstruction de pose dans le cas d'un capteur monoculaire. Les techniques de reconstruction de pose utilisent des primitives extraites de l'image et sont généralement basées sur la connaissance d'un modèle de la scène observée. Malheureusement quelle que soit

la méthode de reconstruction de pose utilisée, l'estimation de l'attitude est très sensible aux bruits de mesure et aux erreurs de calibrage du capteur visuel. Il en résulte, après convergence de l'asservissement lorsque le régime permanent est atteint, un biais entre la position du robot et l'objet d'intérêt à atteindre. Cet inconvénient constitue le problème majeur de l'asservissement visuel 3D.

- **Asservissement visuel 2D**

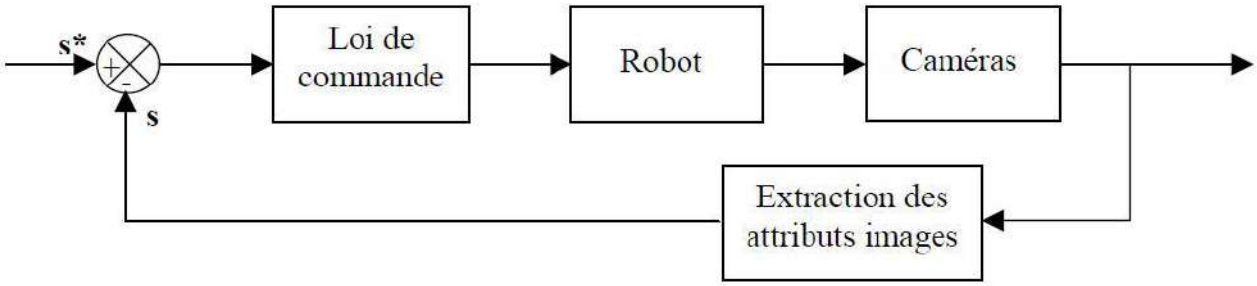


Figure 3.9 : Principe de l'asservissement visuel 2D.

Dans un asservissement visuel 2D, la grandeur à asservir est exprimée sous la forme de primitives visuelles dans l'image. Pour ce type d'asservissement, on parle également "d'asservissement référencé image". Contrairement à un asservissement visuel 3D, le contrôle du robot n'est pas effectué dans l'espace cartésien, mais directement dans l'image. Les primitives extraites de l'image sont généralement des formes géométriques élémentaires qui proviennent de l'objet d'intérêt. Différents types de primitives peuvent être utilisés. Des coordonnées de points dans l'image sont les plus couramment utilisées. Ces primitives peuvent également être des droites, des ellipses, des cylindres, des invariants projectifs, etc... Pour des scènes complexes qui ne contiennent aucune primitive géométrique simple, l'information du mouvement dans l'image ou directement l'information photométrique d'une région d'intérêt de l'image peuvent être utilisée comme primitives visuelles.

- **Asservissement visuel dynamique d2D/dt**

Pour des scènes complexes qui ne contiennent aucune primitive géométrique simple à extraire, l'information du mouvement dans l'image peut être choisie comme primitive visuelle [mahdi]. Dans ce cas, une mesure du flot optique représentant le mouvement 2D apparent dans l'image est régulée à un champ de vitesses désiré.

- **Asservissement visuel hybride 2D 1⁄2**

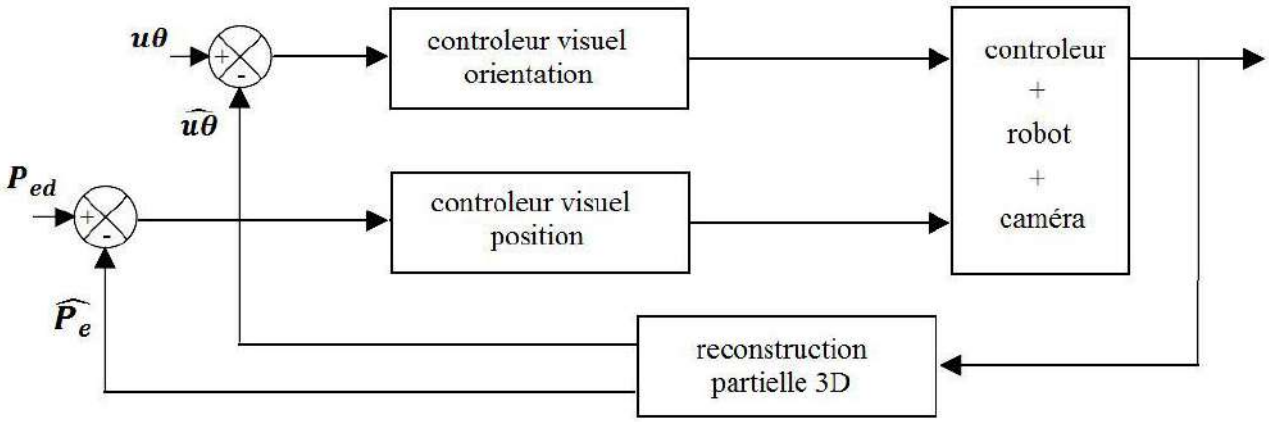


Figure 3.10 : Asservissement visuel 2D1/2.

L'asservissement visuel 2D 1/2 tire partie des avantages des approches 3D et 2D sans en prendre leurs inconvénients. Un vecteur s de dimension 6 est établi à la fois à partir de grandeurs exprimées dans le plan image et dans le repère de la caméra. L'approche la plus connue [mahdi] permet d'obtenir un découplage partiel entre les mouvements de translation et de rotation de la caméra. Elle génère une trajectoire correcte dans l'espace tout en garantissant le contrôle de la projection du centre de l'objet dans l'image afin de maintenir sa visibilité. De plus, cette approche ne requiert pas de calcul de pose et la condition de stabilité asymptotique globale du système a été démontrée [12].

5.3. Selon le type de commande

- **Asservissement visuel séquentiel**

La commande séquentielle qui effectue une approche de type «Static Look & Move» ou encore « Look then Move » est un asservissement en boucle ouverte (donc pas vraiment un asservissement) qui consiste à acquérir et analyser un ensemble d'images issues du capteur puis en générer une trajectoire à atteindre par le robot en boucle ouverte selon le but recherché.

Ce type de commande est utilisé pour des capteurs permettant de faibles fréquences d'acquisition [12] et peut être utilisé pour des missions de re-planification de tâches nécessitant des déplacements lents en utilisant la vision. Cependant, l'absence de retour visuel rend la

convergence de l'asservissement très sensible aux incertitudes de modélisation du robot et du capteur ainsi qu'au aux méthodes de reconstruction de pose.

- **Asservissement visuel cinématique**

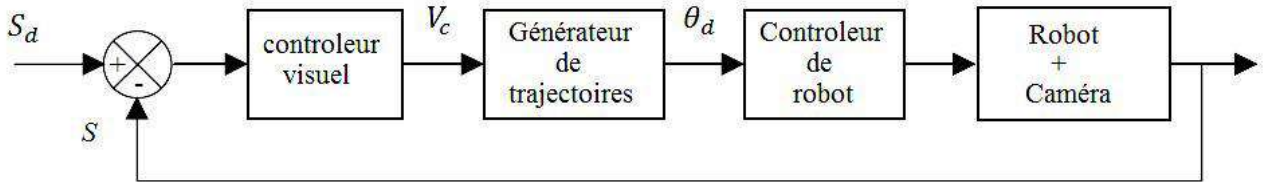


Figure 3.11 : Asservissement visuel cinématique.

La commande cinématique qui effectue une approche de type « Dynamic Look & Move » ou encore « Look and Move » est un asservissement en boucle fermée, où le contrôleur visuel intervient dans une boucle de haut niveau et fourni un état de consigne en vitesse assurant une convergence asymptotique qui est transformé par le Générateur de Trajectoires en consignes angulaires transmises au contrôleur interne bas niveau du robot qui les réalise.
Le robot est considéré comme un simple système de positionnement dont la dynamique est approximée par un gain et les performances du système dépendent essentiellement de la synthèse du contrôleur visuel et du retard introduit par celui-ci dans la boucle de commande. Ce type de commande dit « indirecte » est utilisé pour des asservissements visuels lents dont le but recherché est la robustesse [12].

- **Asservissement visuel dynamique**

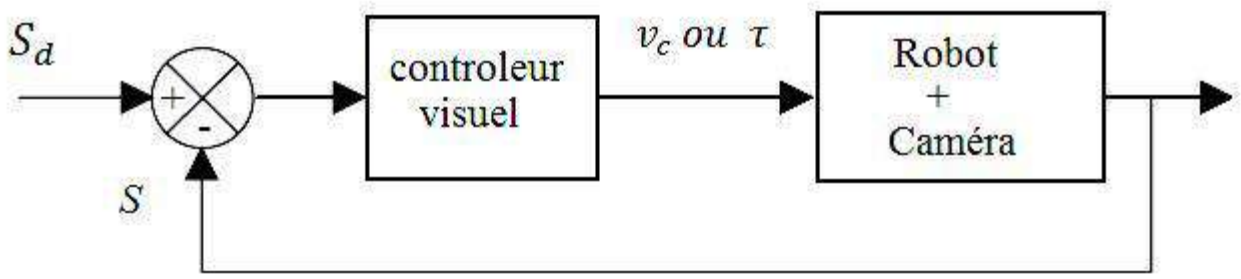


Figure 3.12 : Asservissement visuel dynamique.

La commande dynamique est aussi une alternative de l'approche « Dynamic Look & Move » ou encore « Look and Move » d'asservissement en boucle fermée, où cette fois-ci le contrôleur visuel se substitue au contrôleur interne du robot en envoyant directement les consignes en vitesses ou en couples aux variateurs des moteurs.
La conception du contrôleur visuel tient en compte d'un comportement plus réaliste de la dynamique du robot et parfois même de celui du capteur visuel et/ou des actionneurs pour améliorer les performances dynamiques du système. Ce type de commande dit « directe » est utilisé pour des asservissements visuels rapides [12].

6. Conclusion

Dans ce chapitre, nous avons décrit les déférentes architectures qui existent en asservissement visuel ainsi qu'une classification de ces systèmes de commande. Dans notre travail, nous avons utilisé une de ces architectures de commande référencées « ***Dynamic look and move*** » qui est la structure avec approche fondée l'image.

Grâce à l'utilisation du nouveau système de vision Pixy, nous allons pouvoir améliorer les performances de la plate-forme robot mobile, en réduisant les efforts de calcul du Raspberry. Dans notre application, Le système Pixy fourni directement les informations visuelles de l'objet cible à la carte Raspberry. Ces caractéristiques visuelles sont le contour de la cible ainsi que la position de son centre de gravité dans l'espace image. A partir de ces informations visuelles, une loi de commande est directement calculée est transmise à l'étage de puissance composé de l'ensemble Arduino + Shield.

Dans le prochain chapitre, nous détaillerons les développements que nous avons apporté à l'architecture de la plate-forme. Nos choix quant à la modification de l'architecture et l'utilisation des cartes électroniques embarquées de dernières générations ont considérablement améliorer les performances des algorithmes de traitement d'image et d'asservissement visuel. Ce gain en performances seront mis en exergue in situ au travers d'applications de suivi de cible avec évitement d'obstacles.

Chapitre 4 :
Description de la plate-forme robot mobile

1. Introduction

Comme nous avons motionné avant, nous avons constaté que pour réaliser la commande de la robotique mobile nous avons besoin d'une stratégie de commande. Pour cela, nous allons utiliser un asservissement visuel afin de contrôler la position d'un robot mobile. Cet objectif doit être atteint en utilisant uniquement les données visuelles issues d'une caméra CCD. D'autre part, nous avons intégré une deuxième loi de commande qui permet au robot mobile d'éviter des obstacles. Ce contrôleur se sert de données fournies par 4 capteurs ultrason disposés sur le châssis du robot mobile.

Notre projet vise d'une part à améliorer l'architecture architecture matérielle composée d'un Raspberry et d'un ensemble Arduino /shield et d'autre part la programmation d'un algorithme de contrôle d'un robot mobile capable de réaliser un suivi de cible avec évitement d'obstacle. Une deuxième évolution a été apporté à cette plate-forme est l'introduction du système de vision Pixy. En effet, ce dispositif permet d'acquérir et de traiter les images issues de la camera et par conséquent, nous fournir, à une cadence de 50 images par secondes, les informations visuelles caractéristiques de l'objet cible. C'est une évolution majeure de cette plate-forme robotique dans le sens où nous déchargeons le Raspberry de cette opération d'acquisition et de traitement d'image qui consommait beaucoup de ressources du Raspberry.

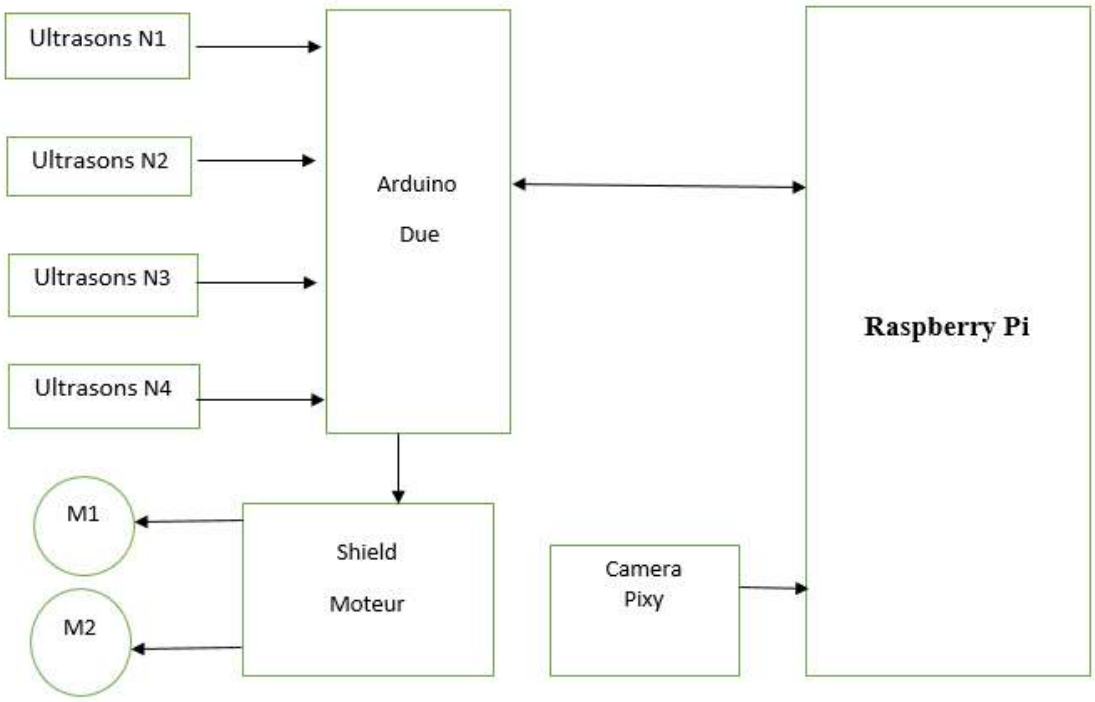


Figure 4.1 : Schéma bloc du projet.

La figure 4.1 représente le schéma synoptique de l'ensemble robot mobile et le système de commande embarquée que nous avons réalisé au cours de notre projet. Ce dernier s'articule autour de 4 blocs principaux. Les deux premiers blocs représentent l'association du robot mobile (avec ces moteurs électriques à courant continu) et de la carte Arduino qui sert d'une part à gérer les capteurs proprioceptifs (capteurs ultrason) et la commande bas niveau des deux moteurs du robot. Tandis que les deux derniers blocs regroupés au sein de la carte raspberry et qui ont pour rôle le calculer des lois de commandes nécessaire à la poursuite de la cible et l'évitement d'obstacles à partir des données fournies par le système de vision Pixy et les capteurs ultrason.

Dans le premier chapitre nous avons parlé sur la robotique mobile en général, nous avons abordé les grandes classes du robot mobile et leur modèle, Il existe plusieurs classes de robots à roues déterminées, principalement, par la position et le nombre de roues utilisées.

Le robot mobile que nous avons utilisé dans notre projet, est un robot mobile de type tricycle, ce robot est techniquement un Arduino robot, nous avons le choisi grâce à sa plate-forme simple, prête-à-utilisé, libre et facile à utiliser, mais nous avons utilisé seulement la plate-forme qui contient les moteurs et les roues. Car au fur et à mesure que nous progressions dans

notre travail, nous nous sommes orientés vers une architecture matérielle universelle capable de commander n'importe quel robot à condition d'avoir accès à l'alimentation de ces moteurs électriques qu'ils soient à CC, pas à pas ou de type servomoteurs. Il suffit juste d'apporter quelques adaptations mineures au programme de la carte Arduino.

2. Robot mobile Arduino

2.1. Description

Notre robot mobile Arduino est de type tricycle, représenté à la figure de la figure 4.2. Il est constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe longitudinal du robot. Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et l'orientation de la roue orientable. De ce point de vue, il est donc très proche d'une voiture. [12]

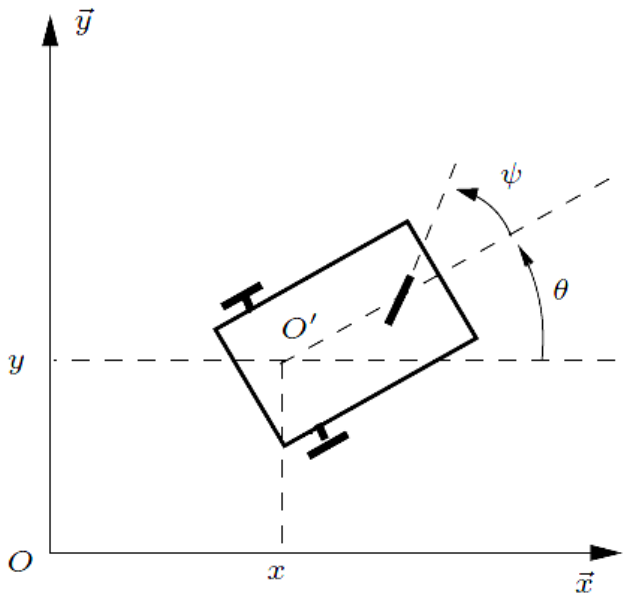


Figure 4.2 : Robot mobile de type tricycle.

2.2. Modélisation

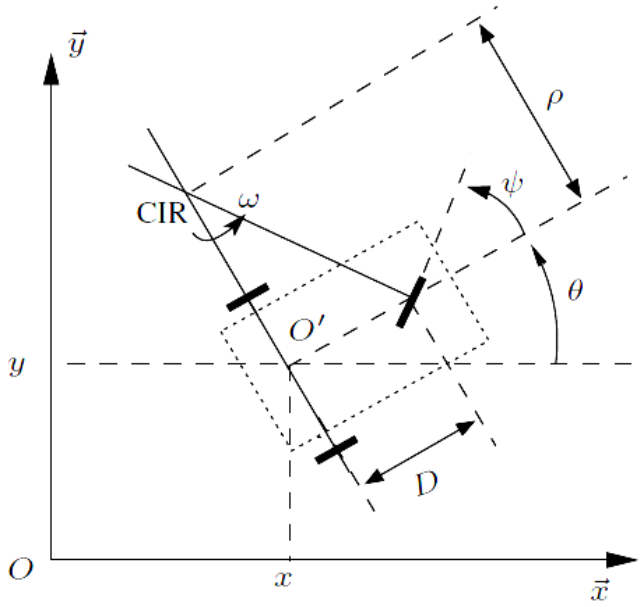


Figure 4.3 : Un robot mobile de type tricycle et son CIR.

Le CIR du robot mobile de type tricycle se situe à la rencontre des axes des roues fixes et de la roue orientable, comme cela est représenté à la figure 4.3. On peut déterminer ρ de manière géométrique à partir de l'angle d'orientation de la roue avant et ω à partir de la vitesse linéaire v du véhicule (vitesse en O') et de ρ :

$$\rho = \frac{D}{\tan\psi}$$

$$\omega = \frac{v}{D}\tan\psi$$

Ce type de robot peut se diriger en ligne droite pour $\psi = 0$ et théoriquement tourner autour du point O' (on pourrait dire sur place) pour $\psi = \frac{\pi}{2}$. Néanmoins, le rayon de braquage de la roue orientable, généralement limité, impose le plus souvent des valeurs de ψ telles que $-\frac{\pi}{2} < \psi < \frac{\pi}{2}$, interdisant cette rotation du robot sur lui-même.

L'écriture des contraintes sur chacune des roues et un raisonnement similaire à celui suivi dans le cas de l'unicycle permettent de déterminer les modèles cinématiques des robots de type tricycle. Toutefois, par un simple raisonnement géométrique, on établit les équations :

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \frac{v}{D}\tan\psi$$
$$\dot{\psi} = \eta,$$

Où $u = (v\ \eta)^T$ est le vecteur de commande cinématique, η représentant la vitesse d'orientation imposée à la roue orientable. Ces équations sont celles du modèle cinématique en configuration simplifié, la configuration associée $q = (xy\theta\psi)^T$ ne décrivant pas les rotations propres des différentes roues.

Comme on l'a vu précédemment, l'existence d'un CIR unique n'impose que les axes des roues du robot soient concourants. [12]

3. Le schéma d'asservissement visuel du robot mobile Arduino

L'élaboration de la loi de commande dans l'espace des images est la partie la plus importante de notre projet. Dans notre application, nous avons placé une caméra numérique sur le robot pour repérer la cible que nous avons désigné au préalable. Le schéma de commande que nous avons adopté est un asservissement visuel avec approche fondée sur l'image tel décrit par la figure 4.4

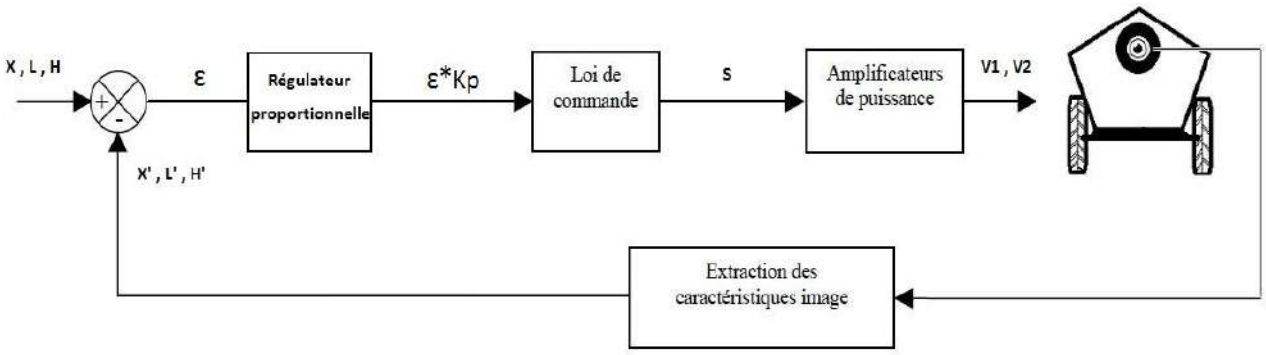


Figure 4.4 : structure d'asservissement visuel utilisé.

Cette structure consiste à calculer directement les valeurs des commandes à partir des caractéristiques images. L'erreur de positionnement est calculer directement dans l'espace images et c'est cette variable que le contrôleur devra réduire.

Ces caractéristiques sont extraites à partir des informations issues de la Cam Pixy. Lorsque la cible désignée apparaît dans le champs de la caméra, le traitement

d'images va repérer cette cible est va nous fournir un certains nombres de données sur cette objet en mouvement.

L'utilisation de la vision monoculaire, ne permet pas d'estimer la profondeur de la cible. Pour contourner cette limitation (en évitant d'introduire une deuxième caméra), nous avons calculé la surface de la cible. Si la valeur de surface grandie, la cible est proche de la caméra et donc du robot et inversement si, la valeur de la surface diminue, la cible s'éloigne du robot. Par conséquence, cette approche nous permettra d'asservir et de suivre la cible quand elle se déplace en perpendiculaire du plan image.

Quant aux déplacements de la cible sur le plan parallèle au plan image, nous utiliserons les coordonnées (estimées en pixels) X, Y du centre de gravité de la cible.

En réalité, nous utilisons uniquement la coordonnée X pour maintenir le centre de gravité de la cible au centre de l'image.

Par conséquent, grâce à l'estimation de la position de la cible dans l'espace cartésien (faite à partir des données visuelles L, H et X), nous avons contraint le robot à maintenir la cible à une distance donnée et surtout au centre de l'image. C'est le principe même de l'asservissement visuel avec une approche basée sur l'image.

4. L'architecture matérielle de gestion et de commande de la plate-forme robot mobile

4.1. L'extension (Shield) de commande des moteurs

Cette extension (Adafruit Motor Shield) est basée sur le circuit CI L293D, qui est un double pilote à pont complet conçu pour gérer des charges inductives telles que les relais, les solénoïdes, les moteurs DC et les servomoteurs. Il vous permet de commander 4 moteurs DC avec votre carte Arduino, en contrôlant la vitesse et la direction de chacun d'eux indépendamment.

Adafruit Motor Shield ne doit être alimenté que par une alimentation externe. Parce que le CI L293D monté sur le blindage comporte deux connexions d'alimentation séparées, l'une pour la logique et l'autre pour l'alimentation des charges inductives. Le courant requis du moteur dépasse souvent la note maximale du courant USB.

L'alimentation externe (non-USB) peut provenir d'un adaptateur AC-DC ou d'une batterie. L'adaptateur peut être branché en branchant une prise centrale positive de 2,1 mm dans la prise

d'alimentation de la carte Arduino sur laquelle le blindage du moteur est monté ou en connectant les fils qui alimentent l'alimentation des bornes à vis (la pin Vin et GND).

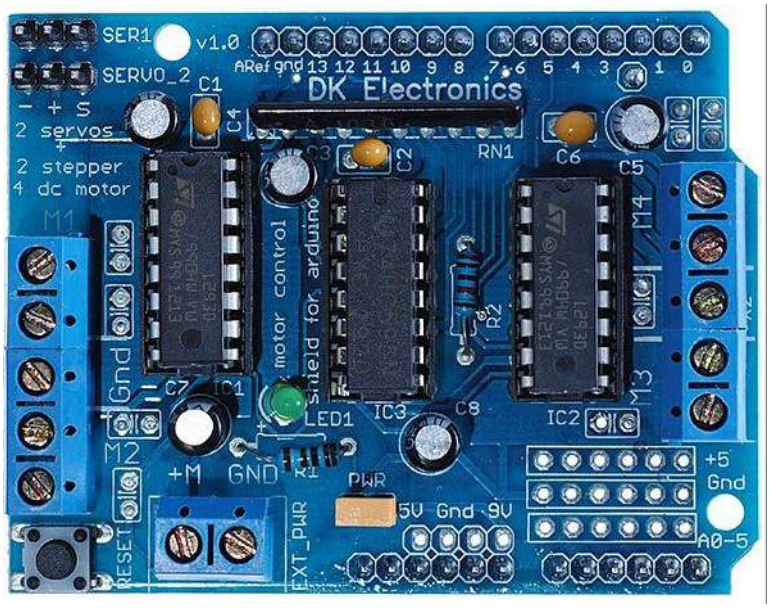


Figure 4.5 : Adafruit Motor Shield.

a) Principales caractéristiques :

- Tension d'alimentation du circuit logique : 5V
- Tension d'alimentation du circuit puissance : 0V→ 8V
- Tension minimale de sensibilité MLI : 2.3V
- Courant maximal du circuit puissance : 3A

b) Les moteurs et le registre à décalage

Le blindage du moteur peut contrôler jusqu'à quatre moteurs à courant continu à l'aide de deux circuits intégrés de moteur à moteur à double pont L293D. Vous pouvez lire la fiche technique pour plus de détails. Comme le montre la Figure 4.6, chaque CI possède quatre entrées numériques qui commandent bidirectionnellement deux moteurs à courant continu. Les entrées sont par paires, et chaque paire contrôle un moteur. Par exemple, si la broche 2 est réglée en haut et que la broche 7 est bas, le moteur 1 est allumé. Si les entrées sont inversées, le moteur est à nouveau activé, mais son sens de rotation est inversé, et si les deux sont bas, le moteur est éteint. Il existe d'autres configurations possibles, mais celle-ci est supportée par le bouclier du moteur. [17]

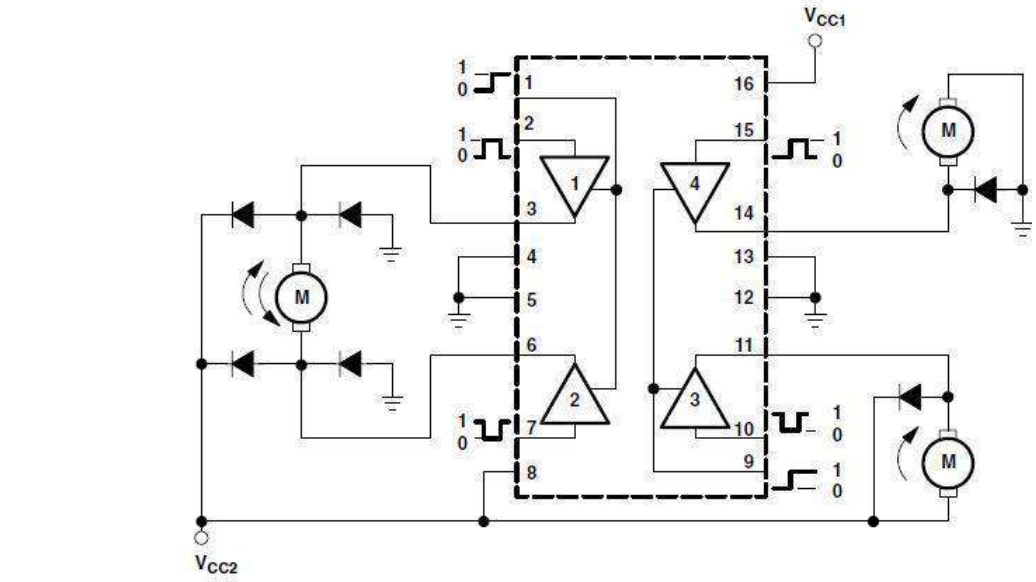


Figure 4.6 : fonctionnement de L293D.

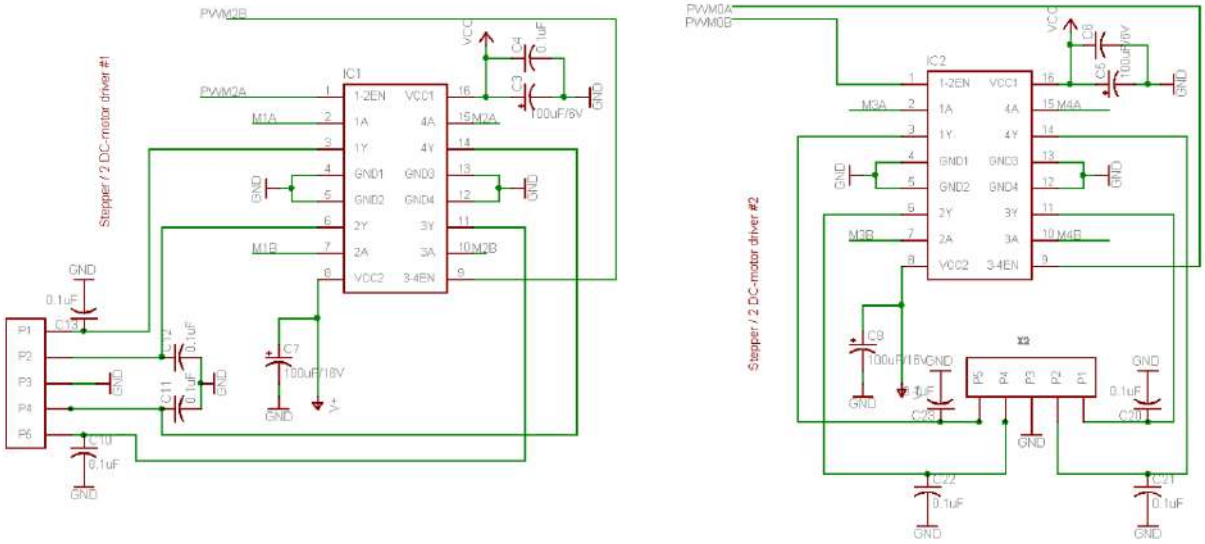


Figure 4.7: montage L293D avec Shield.

Les entrées des moteurs sont effectivement câblées sur les sorties d'un registre à décalage 74HCT595N à 8 bits Figure 4.8 Cet IC vous permet d'alimenter en bits en série, mais il les présente ensuite comme des sorties parallèles. Il est contrôlé par quatre entrées numériques, de sorte qu'il vous permet de contrôler huit entrées aux moteurs (donc quatre moteurs) en utilisant quatre broches de l'Arduino. [17]

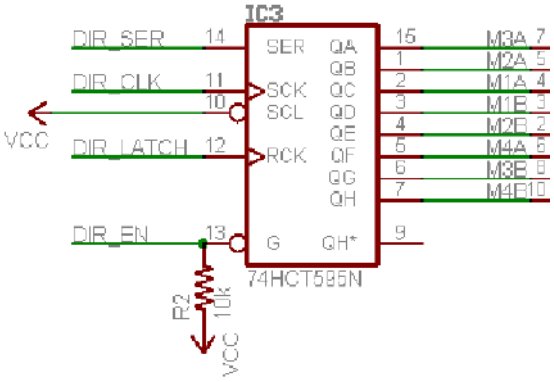


Figure 4.8 : IC 74HCT595N.

Les broches étiquetées QA à QH sont les broches de sortie parallèles, et vous pouvez les considérer comme des bits 0 à 7. Les chiffres immédiatement à droite du CI sont des chiffres de broches plutôt que des nombres de bits, bien qu'ils coïncident surtout. QA est un bit zéro, broche 15. La fiche technique les appelle Q0 à Q7, de sorte que je ne sais pas pourquoi le schéma utilise une convention différente.

Comme vous pouvez le voir, le moteur 1 est contrôlé par les bits 2 et 3, le moteur 2 par les bits 1 et 4, etc.

Les broches illustrées à gauche sur Tableau4.1 et Figure 4.8 "74HC595" sont les quatre broches d'entrée plus la réinitialisation et la validation de sortie. La convention de dénomination d'Adafruit est en contradiction avec celle fournie dans la fiche technique 74HCT595N à 8 bits de registre à décalage Philips. Voici un tableau de référence croisée. [17]

Tableau 4.1 : brochage d'IC 74HCT595N.

ARDUINO BROCHES	IC 74HCT595N BROCHES		
8	DS	SERIE DATA	14
4	SH_CP	CLK	11
12	ST_CP	LACH	12
7	OE	SELECTION	13

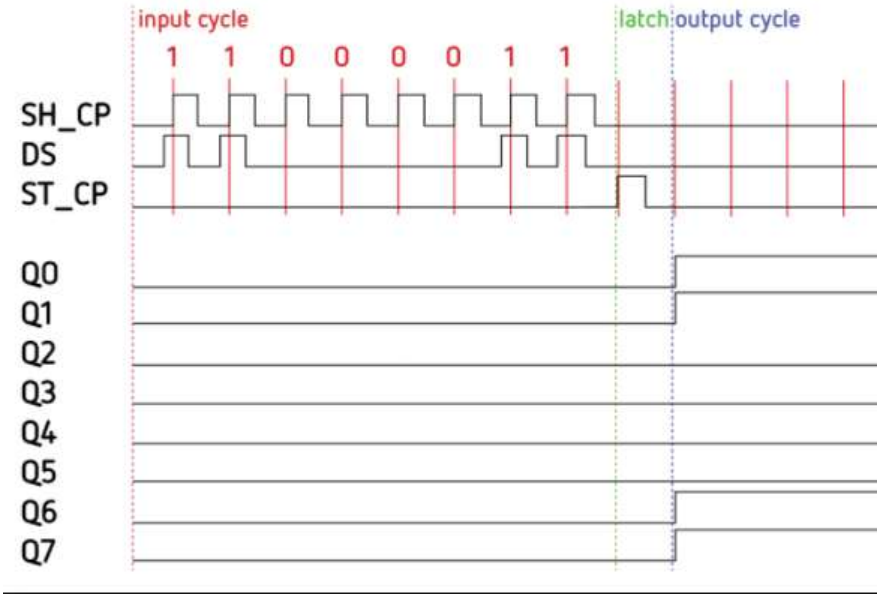


Figure 4.9 : fonctionnement de IC 74HCT595N.

4.2. Les capteurs ultrasons

Un capteur à ultrasons transmet des ondes ultrasonores dans Air et détecte les ondes réfléchies à partir d'un objet. Là sont nombreuses les applications pour les capteurs à ultrasons, comme dans Systèmes d'alarme d'intrusion, ouvre-portes automatiques et Capteurs de secours pour automobiles. Accompagné du développement rapide de l'information Technologie de traitement, nouveaux champs d'application, tels que l'équipement d'automatisation industrielle et l'électronique automobile, sont croissant et devrait continuer à le faire. En utilisant sa fabrication unique de céramique piézoélectrique technologie développée depuis de nombreuses années, Murata a développé différents types de capteurs à ultrasons qui sont compact et pourtant très performant. [18]

Les capteurs à ultrasons (ou télémètres ultrasonores) sont aujourd'hui les capteurs les plus communément employés dans les systèmes de robotique mobile autonome en raison de leurs bas prix, de leur faible consommation, de leur simplicité et de leur compacité. Ils ont été exploités dans des domaines tels que la navigation, la modélisation d'environnement, l'évitement d'obstacle ou l'estimation de la position.

Les capteurs à ultrasons mesurent le temps de vol propagation d'une onde ultrasonore (de 20 Khz à 200 Khz) entre son émission et sa réception après réflexion. A partir de ce temps et

connaissant la vitesse de propagation de l'onde, la distance séparant le capteur de l'obstacle qu'il perçoit est directement disponible sans analyses complémentaires. Les acquisitions des mesures sont rapides, ils peuvent donc être utilisés pour les processus qui demandent un traitement en temps réel comme l'évitement d'obstacles. Il existe néanmoins des erreurs potentielles comme la variation de la vitesse de propagation de l'onde qui peut être influencée par les changements de température, par les courants d'air et dans une moindre mesure par l'humidité. Une autre caractéristique de ces capteurs est leur cône d'émission assez large (de 20 à 30°) qui ne permet pas de connaître précisément la position de l'obstacle dans le cône d'émission du capteur. [8]

a) Caractéristiques des ondes ultrasonores

Les ondes ultrasonores sont des sons qui ne peuvent être entendus par les humains et sont normalement des fréquences supérieures à 20 kHz. Les caractéristiques de base des ondes ultrasonores sont expliquées ci-dessous.

- **Longueur d'onde et rayonnement**

La vitesse de propagation des ondes s'exprime par multiplication de fréquence et de longueur d'onde. La vitesse d'un l'onde électromagnétique est de 3 × 108 m / s, mais la vitesse de la propagation de l'onde sonore dans l'air est aussi lente que d'environ 344 m /S (à 20 ° C). À ces vitesses plus lentes, les longueurs d'onde sont court, ce qui signifie qu'une plus grande résolution de la distance et La direction peut être obtenue.

- **La réflexion**

Afin de détecter la présence d'un objet, les ondes ultrasons se reflètent sur les objets.

Comme le métal, le bois, le béton, le verre, le caoutchouc et le papier, etc. reflètent environ 100% des ondes ultrasonores, Ces objets peuvent être facilement détectés.

Le tissu, le coton, la laine, etc. sont difficiles à détecter car ils absorbent les ondes ultrasonores. Il peut être souvent difficile, aussi, pour détecter les objets ayant une grande ondulation de surface, À cause d'une réflexion irrégulière.

- **Effets de la température**

La vitesse de propagation des ondes sonores "C" est exprimée par la formule suivante :

$$C=331.4 + (0.606 \times T) + (0.0124 \times H)$$

T = Température (° C)

H= humidité (g/m^3)

C'est comme si la vitesse du son varie selon la Température et Humidité initiale, il est nécessaire de vérifier la température, tout le temps pour mesurer la distance à l'objet avec précision. [18]

5. HC-SR04 - Module de détection aux ultrasons

Le capteur HC-SR04 Figure 4.10 utilise les ultrasons pour déterminer la distance d'un objet. Il offre une excellente plage de détection sans contact, avec des mesures de haute précision et stables. Son fonctionnement n'est pas influencé par la lumière du soleil ou des matériaux sombres, bien que des matériaux comme les vêtements puissent être difficiles à détecter.



Figure 4.10 : HC-SR04 ultrasons.

5.1. Caractéristiques

- Dimensions : 45 mm x 20 mm x 15 mm
- Plage de mesure : 2 cm à 400 cm
- Résolution de la mesure : 0.3 cm
- Angle de mesure efficace : 15 °
- Largeur d'impulsion sur l'entrée de déclenchement : 10 µs (Trigger Input Pulse width)

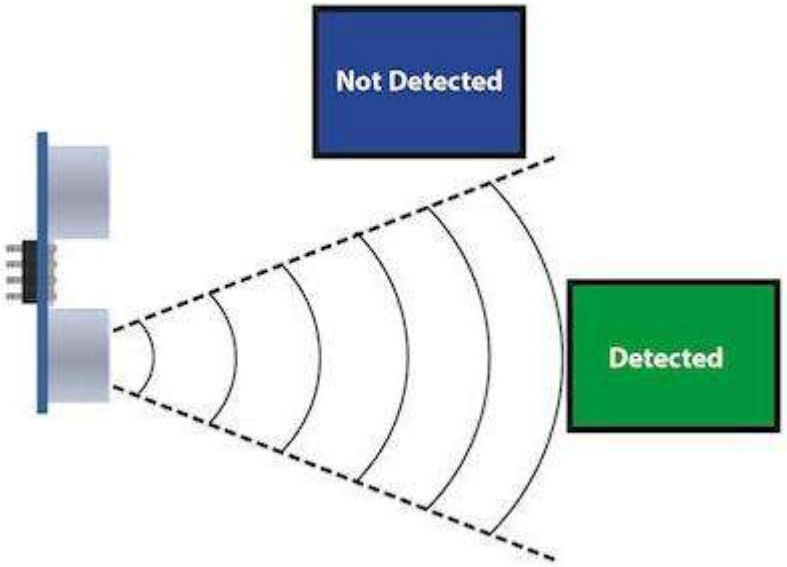


Figure 4.11 : Angel de détection.

Broches de connections

- Vcc = Alimentation +5 V DC
- Trig = Entrée de déclenchement de la mesure (Trigger input)
- Echo = Sortie de mesure donnée en écho (Echo output)
- GND = Masse de l'alimentation

Tableau 4.2 : Spécifications et limites.

Paramètre	Min	Type	Max	Unité
Tension d'alimentation	4.5	5.0	5.5	V
Courant de repos	1.5	2.0	2.5	mA
Courant de fonctionnement	10	15	20	mA
Fréquence des ultrasons	-	40	-	KHz

5.2. Fonctionnement de ce capteur

Pour déclencher une mesure, il faut présenter une impulsion "high" (5V) d'au moins 10 µs sur l'entrée "Trig". Le capteur émet alors une série de 8 impulsions ultrasoniques à 40 kHz, puis il attend le signal réfléchi. Lorsque celui-ci est détecté, il envoie un signal "high" sur la sortie "Echo", dont la durée est proportionnelle à la distance mesurée

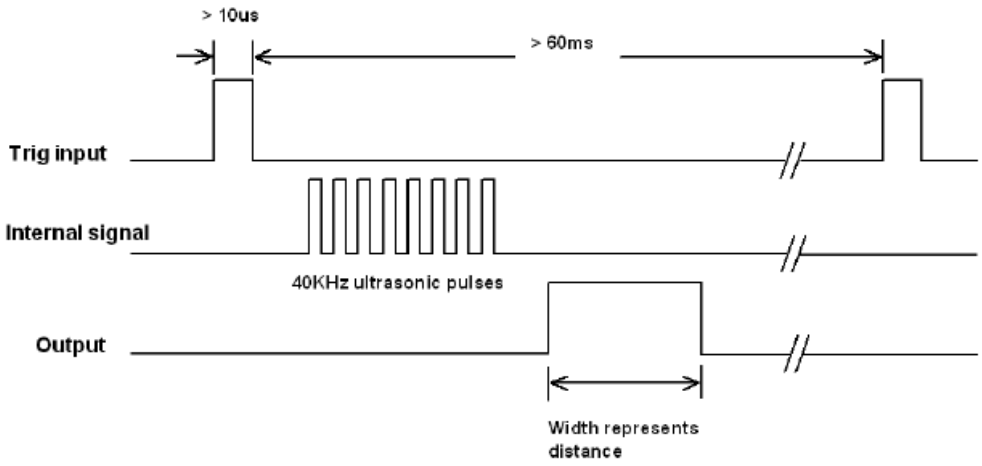


Figure 4.12 : chronogramme de HC-SR04 ultrasons.

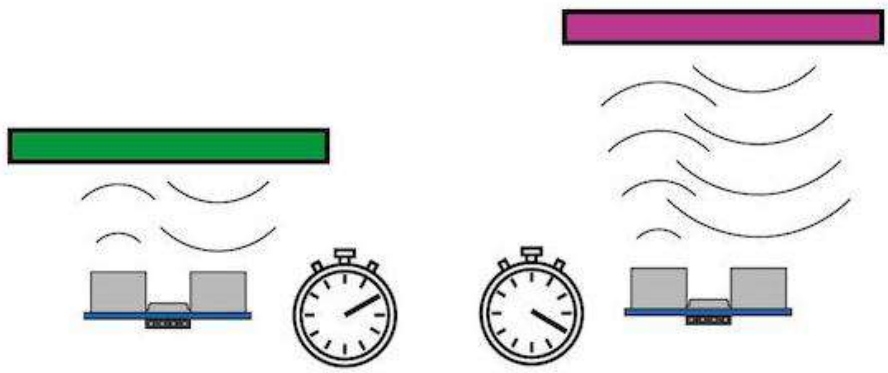


Figure 4.13 : le temps de propagation.

5.3. Distance de l'objet

La distance parcourue par un son se calcule en multipliant la vitesse du son, environ 340 m/s (ou 34'000 cm/1'000'000 µs) par le temps de propagation, soit :

$$d = v \cdot t \text{ (distance = vitesse} \cdot \text{temps)}$$

Le HC-SR04 donne une durée d'impulsion en dizaines de µs. Il faut donc multiplier la valeur obtenue par 10 µs pour obtenir le temps t. On sait aussi que le son fait un aller-retour. La distance vaut donc la moitié.

$$d = 34'000\ cm/1'000'000\ \mu s \cdot 10us \cdot valeur / 2$$

En simplifiant

$$d = 170'000 /1'000'000\ cm \cdot valeur$$

Finalement

$$d = 17/100\ cm \cdot valeur$$

La formule d = durée/58 cm figure (4.13) aussi dans le manuel d'utilisation du HC-SR04 car la fraction 17/1000 est égale à 1/58.8235. Elle donne cependant des résultats moins précis.

Si l'objet absorbe le son, ou s'il n'y a rien pour une longue distance, ou si l'objet rebondit le son à un angle, il se peut qu'il n'y ait pas assez de son pour revenir à mesurer quelque chose.

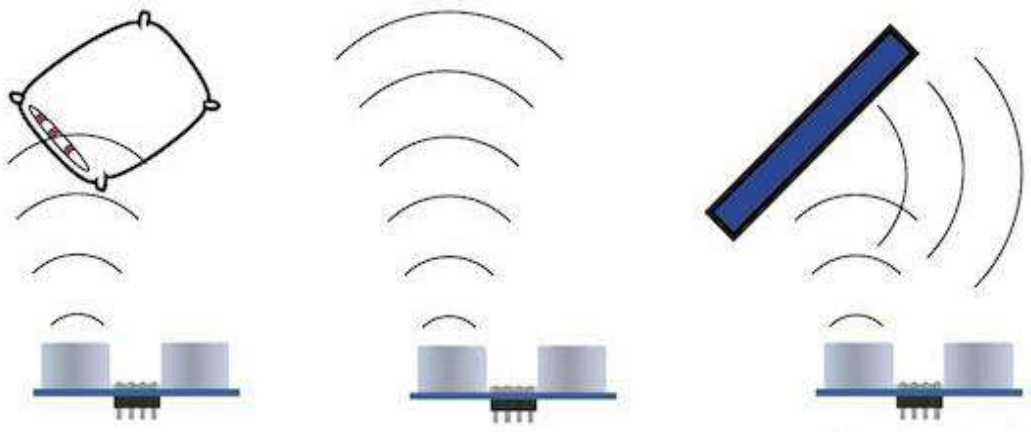


Figure 4.14 : les objets difficiles à détecter.

6. Le système de vision Pixy

Comme nous l'avons mentionné dans le chapitre précédent sur les caractéristiques de la Cam Pixy, Nous présentons maintenant les étapes nécessaires pour détecter un objet à Pixy.

6.1. La détection d'objet objet avec Pixy

Enseigner à Pixy un objet est trés facile, mais abordons d'abord les objets qui fonctionneront bien avec Pixy. Pixy utilise un algorithme de filtrage des couleurs basé sur la teinte pour détecter les objets. Puisque Pixy utilise la teinte (couleur), l'objet doit avoir une teinte distincte. Voici quelques objets qui sont bons parce qu'ils ont des teintes bonnes et distinctes.



Figure 4.15 : des objets avec des teintes bonnes et distinctes.

Voici de mauvais objets car il n'y a pas de teinte (noir, blanc ou gris) ou la teinte n'est pas distincte.



Figure 4.16 : des objets avec teintes non distinctes.

Gardant ces remarques à l'esprit, choisissez un objet pour que Pixy le détecte. Lorsque vous allumez Pixy, il passera par une série de flashs LED. Attendez que la LED s'éteigne avant de presenter à Pixy un objet. Maintenez le bouton en haut de Pixy enfoncé. Après environ 1

seconde, la LED s'allume - premièrement blanche, puis rouge, puis d'autres couleurs - mais quand elle devient rouge, relâchez le bouton.

Lorsque vous relâchez le bouton, Pixy entrera dans ce que l'on appelle le mode "tube léger", où la couleur LED est la couleur de l'objet sur lequel Pixy a été "verrouillé". Pixy va se verrouiller sur les objets au centre de son cadre vidéo, alors maintenez l'objet directement devant Pixy, entre 6 et 20 cm de l'objectif.

Pixy utilise un algorithme de croissance de région pour essayer de déterminer quels pixels font partie de votre objet et quels pixels font partie de l'arrière-plan. À l'aide de ces pixels, Pixy tentera de créer un modèle statistique de votre objet afin qu'il puisse le détecter de manière fiable dans différentes conditions d'éclairage. Utilisez la couleur LED comme retour pour déterminer si Pixy a un bon verrou sur l'objet, et utilisez les instructions suivantes pour juger :

1. Lorsque Pixy a verrouillé sur votre objet, la couleur LED doit correspondre à la couleur de votre objet.

2. Plus la LED est lumineuse, meilleure est la serrure. Déplacez l'objet un peu et voyez si vous pouvez optimiser la luminosité de la LED.

3. Regardez la grille de la région dans la fenêtre PixyMon et voyez comment la taille de la grille correspond à la couleur et à la luminosité de la LED. Une grille qui couvre plus de votre objet est supérieure à une grille qui ne couvre qu'une partie de votre objet et aboutit à une LED plus lumineuse. L'image ci-dessous montre une grille qui couvre la majeure partie de l'objet (bon). [19]

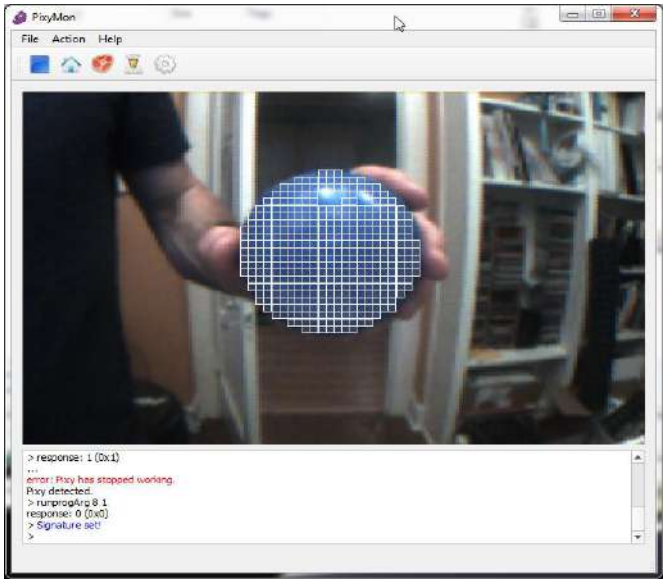


Figure 4.17 : la majeure partie de l'objet (bon) couvre par une grille.

Lorsque vous êtes satisfait, appuyez et relâchez le bouton sur Pixy, comme si vous aviez cliqué sur votre souris. La LED clignotera quelques fois pour indiquer que Pixy a maintenant appris votre objet. Il va maintenant commencer à suivre cet objet.

6.2. Objets à signatures multiples

Pixy peut apprendre jusqu'à sept signatures de couleurs. Nous pouvons apprendre à Pixy la 1ère signature couleur en relâchant le bouton lorsque la LED devient rouge. Si nous continuons d'appuyer sur le bouton, la LED devient orange, jaune, etc., indiquant les signatures de couleur restantes. Voici les signatures dans l'ordre :

1. rouge 2. Orange 3. Jaune 4. Vert 5. Cyan (bleu clair) 6. Bleu 7. Violet

Le numéro de signature couleur est déterminé lorsque vous relâchez le bouton. Relâchez le bouton lorsque la LED est jaune et que vous êtes en train d'enseigner la signature 3. Relâchez le bouton lorsque la LED est bleue et que vous enseignez la signature 6. Ces couleurs ne sont pas liées à la teinte de l'objet. Les couleurs sont utilisées uniquement pour indiquer le numéro de signature. Ainsi, par exemple, la signature 1 peut être un objet jaune, même si la signature 1 est indiquée par une LED rouge, et la signature 2 pourrait être un objet rose même si la signature 2 est indiquée par une LED orange.

Après avoir enseigné la signature de Pixy, elle enregistre la signature en flash. Par conséquent, lorsque vous mettez le cycle de votre Pixy, il se souviendra des signatures que vous lui avez enseignées et continuerez à suivre les objets qui correspondent à ces signatures. [19]

6.3. Balance des blancs

Certains types d'éclairage (comme les incandescents) ont une teinte rougeâtre et d'autres (comme les fluorescents) ont une teinte bleutée. L'éclairage peut affecter vos signatures de couleur. Par exemple, si vous apprenez un objet sous un éclairage incandescent et que vous vous déplacez dans une pièce avec un éclairage fluorescent, les signatures des couleurs ne fonctionneront probablement plus. Vous pouvez soit ré-enseigner toutes les signatures, soit ajuster la balance des blancs. [19]

7. L'utilisation de l'application PixyMon

Vous pouvez également enseigner à Pixy un objet à travers PixyMon. Cela peut être utile si l'objet que vous souhaitez enseigner est petit ou si vous souhaitez plus de contrôle sur les pixels utilisés pour l'enseignement. Commencez par brancher le câble USB entre Pixy et votre ordinateur et exécutez PixyMon.

Maintenez maintenant l'objet que vous souhaitez enseigner devant Pixy et sélectionnez **Action➜Set signature 1** dans le menu déroulant.

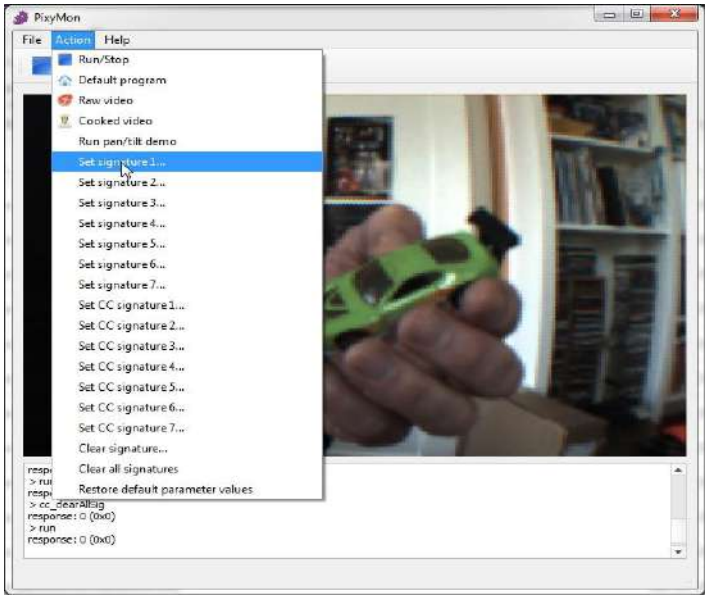


Figure 4.18 : maintenez la signature de l'objet que vous souhaitez enseigner.

À l'aide de la souris, cliquez et faites glisser pour sélectionner la région que vous souhaitez que Pixy utilise pour apprendre l'objet.

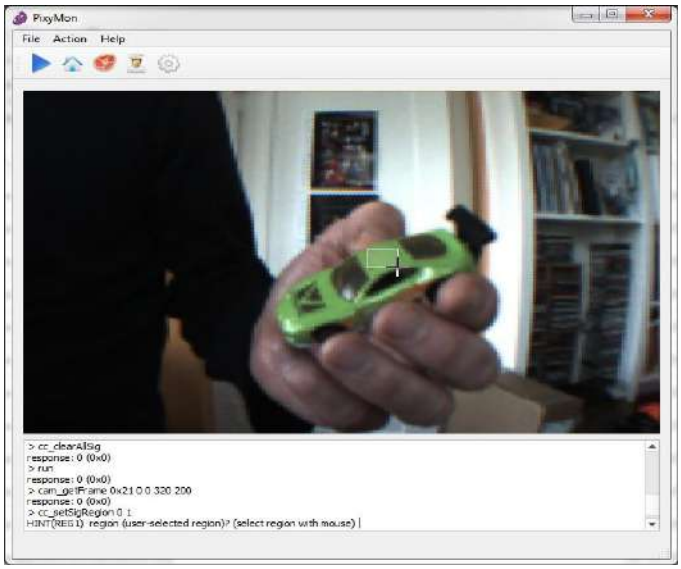


Figure 4. 19 : sélectionner la région que vous souhaitez que Pixy utilise pour apprendre l'objet.

C'est tout ! Après avoir sélectionné la région, Pixy "apprendra" l'objet et entrera automatiquement dans le mode vidéo "cuit" afin de vérifier à quel point votre signature couleur fonctionne.

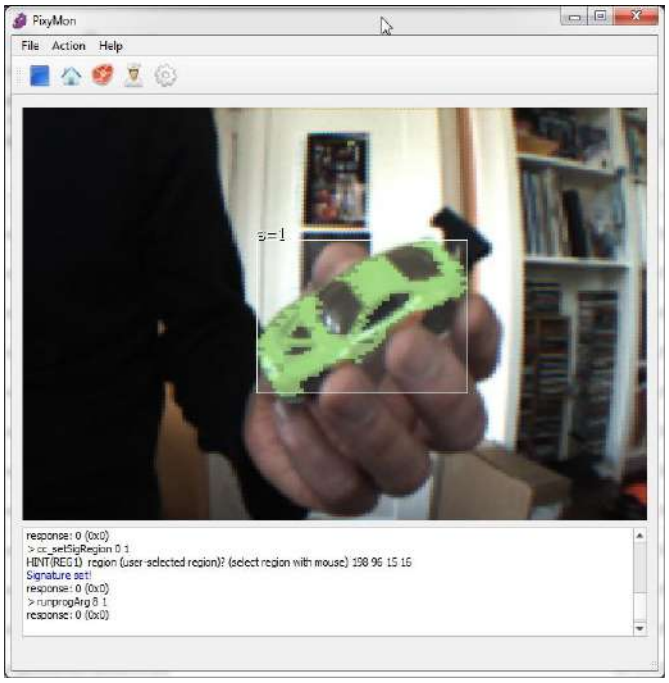


Figure 4.20 : Pixy "apprendra" l'objet.

7.1. Signature tuning

Parfois, les signatures de couleurs que vous enseignez à Pixy doivent être "tordues" - c'est-à-dire que vous obtenez des faux positifs (Pixy détecte des objets qui ne sont pas les objets que vous vouliez) ou des faux négatifs (Pixy ne détecte pas l'objet Vous l'avez appris, ou il détecte l'objet par intermittence). Ceci peut etre resolu grâce à la modification d'un certains nombres de paramètres. Il suffit de faire apparaître la boîte de dialogue, sélectionnez File➜Configure), puis sélectionnez le **Signature Tuning** sous **Pixy Parameters**. [19]

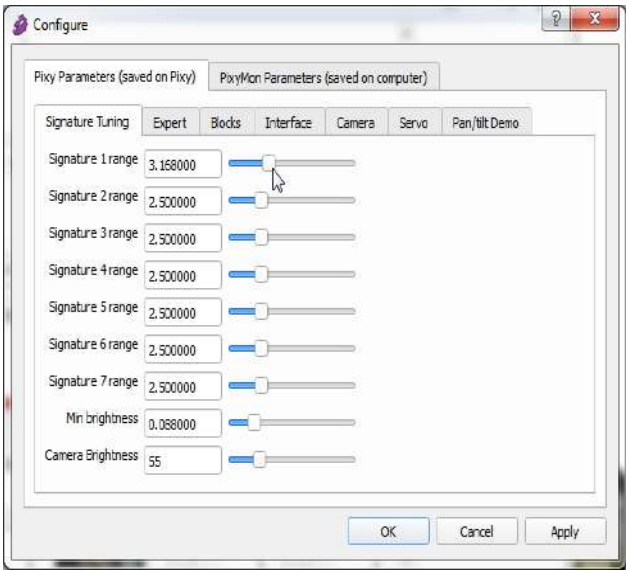


Figure 4.21 : Pixy Paramètres**.**

Utilisez le curseur pour **Signature 1 range** pour ajuster l'inclusivité de la signature 1 (en supposant que c'est la signature 1 que vous souhaitez ajuster). Faites-le glisser vers la gauche si vous voulez être moins inclusif (c'est-à-dire si vous voyez des faux positifs, comme l'image ci-dessous):

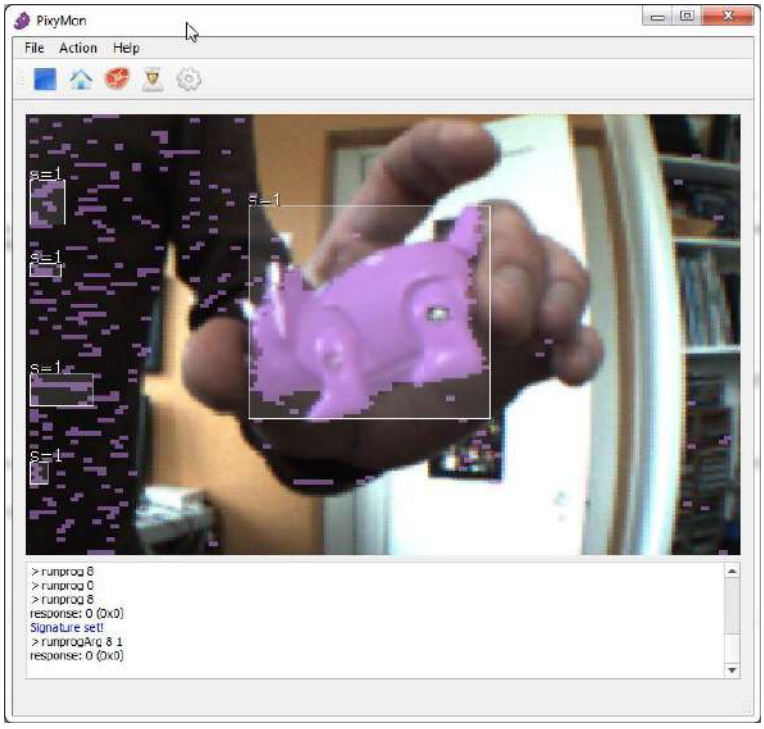


Figure 4.22 : ajuster l'inclusivité de la signature 1.

Ou glissez-le vers la droite si vous voulez être plus inclus (c'est-à-dire que vous voyez des faux négatifs) ou que la détection est intermittente ou clairsemée (comme dans l'image ci-dessous) :

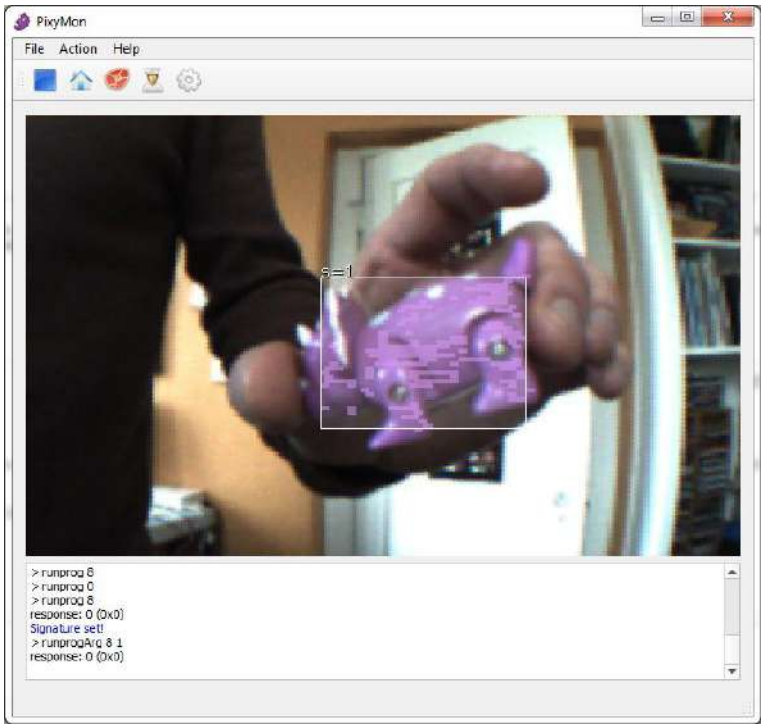


Figure 4.23: la détection intermittente ou clairsemé.

Choisissez une valeur de curseur qui procure une bonne détection forte, comme l'image ci-dessous :

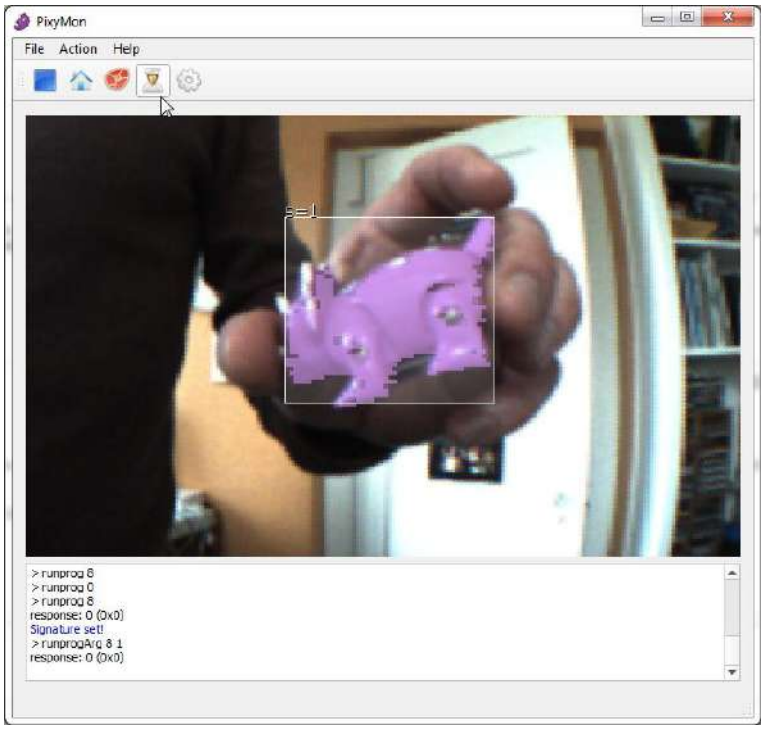


Figure 4.24 : valeur de curseur qui procure une bonne détection.

Vous pouvez ajuster toutes les signatures de sept couleurs de cette façon pour maximiser la précision de détection pour toutes les signatures. Assurez-vous d'appuyer sur **Apply** ou sur **OK** pour enregistrer les plages de curseurs ! Les valeurs ajustées ne seront pas sauvegardées si vous appuyez sur **Cancel** ou désistez la boîte de dialogue. [19]

8. La carte Raspberry Pi: le cerveau

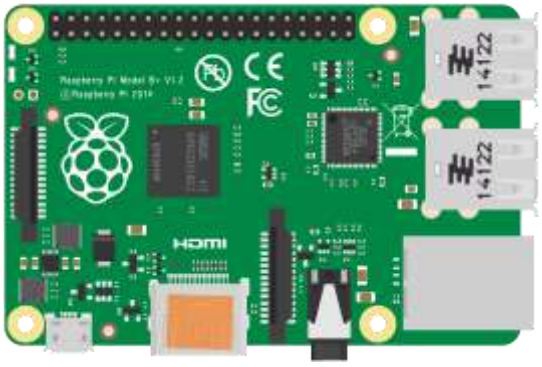


Figure 4.25 : la carte Raspbery Pi utilisé.

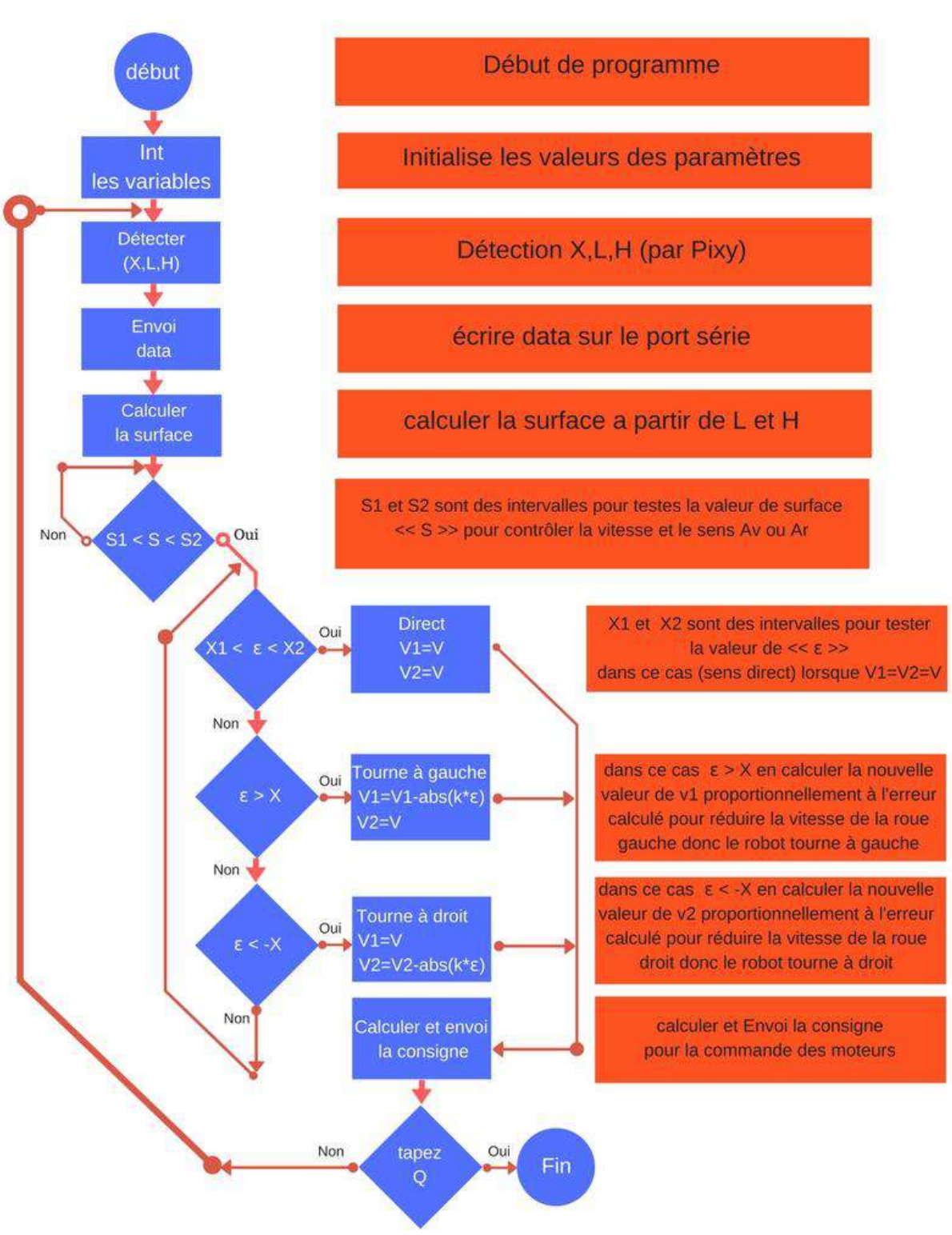
Cette carte, de taille d'une carte de crédit, est considérée comme étant le « cerveau » de notre robot parce qu'elle est en charge de l'acquisition, du calcul des lois de commande. Ce mini PC, doté d'une mémoire importante et d'un Processeur puissant fonctionne sous le système d'exploitation LINUX.

Le modèle de Raspberry Pi 2 utilisé le modèle B+, nous avons choisi ce modèle grâce à ces caractéristiques, (voir Chapitre 2 (4. La carte Raspberry Pi))

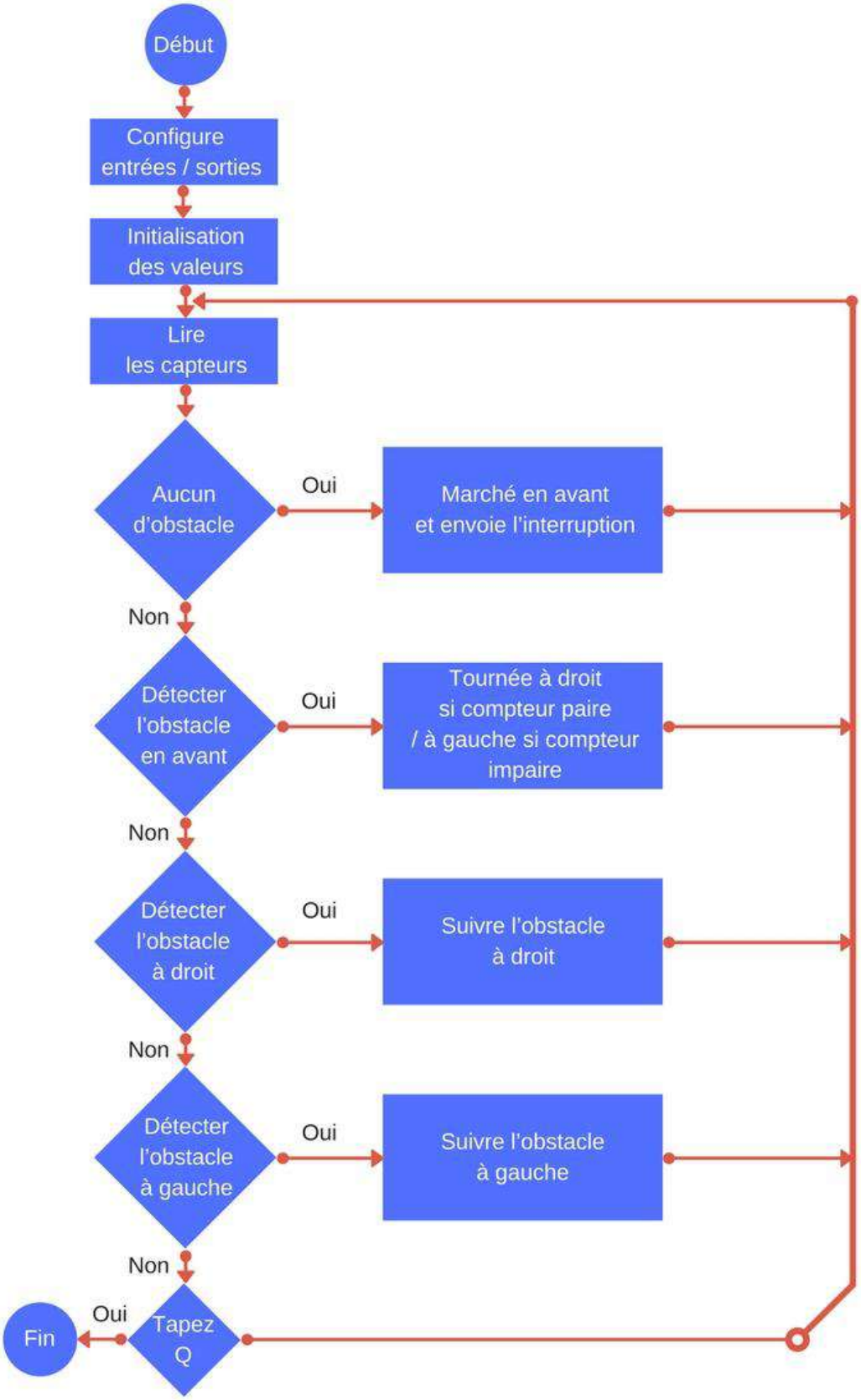
Grace au système Linux, nous avons pu programmé l'ensemble de nos algorithmes en langage Python et exploité les fonctions précompilées de la Cam Pixy. Dans la suite de ce chapitre, nous décrirons brièvement les techniques que nous avons mis en œuvre pour réaliser notre asservissement visuel à savoir : le traitement d'image basé sur l'exploitation des ressources fournie par Pixy, les organigrammes écrits en langage Python. Ces programmes contiennent aussi une procédure pour transmettre, via le port USB, les consignes au module Arduino / Shield en charge de les convertir en taisions nécessaires aux mouvements des moteurs du robot mobile.

9. Les organigrammes des différents contrôleurs

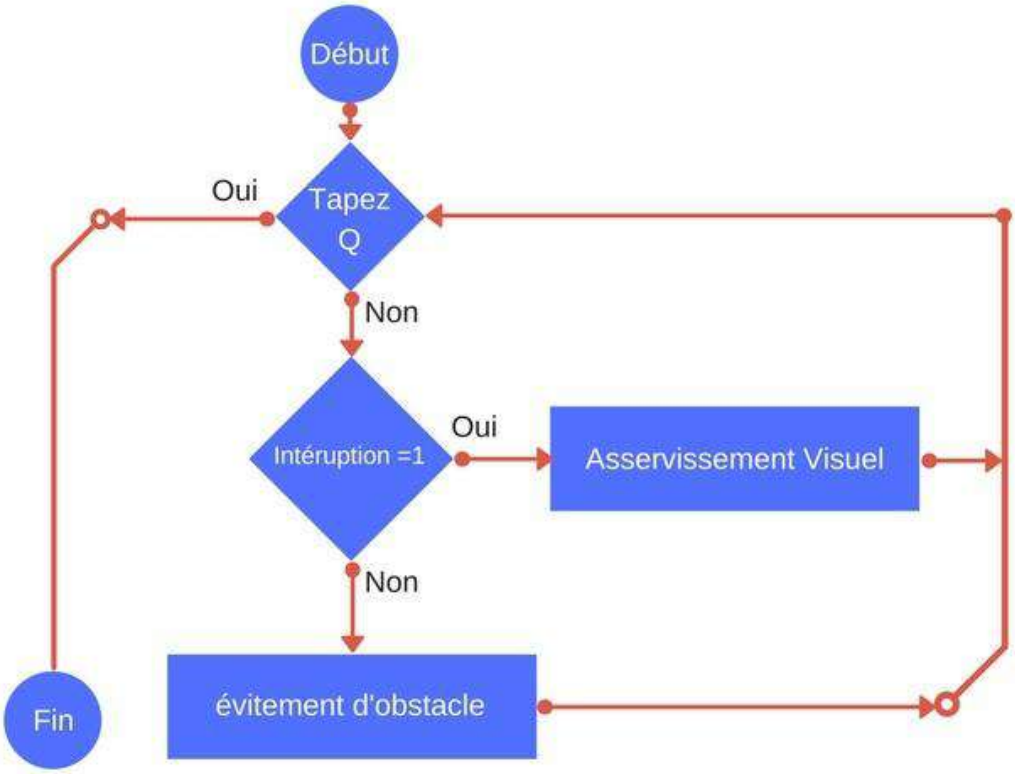
9.1 L'organigramme du contrôleur visuel



9.2. L'organigramme du Contrôleur en charge de l'évitement d'obstacles



9.3. L'organigramme du contrôleur hiérarchique d'asservissement visuel avec évitement d'obstacle



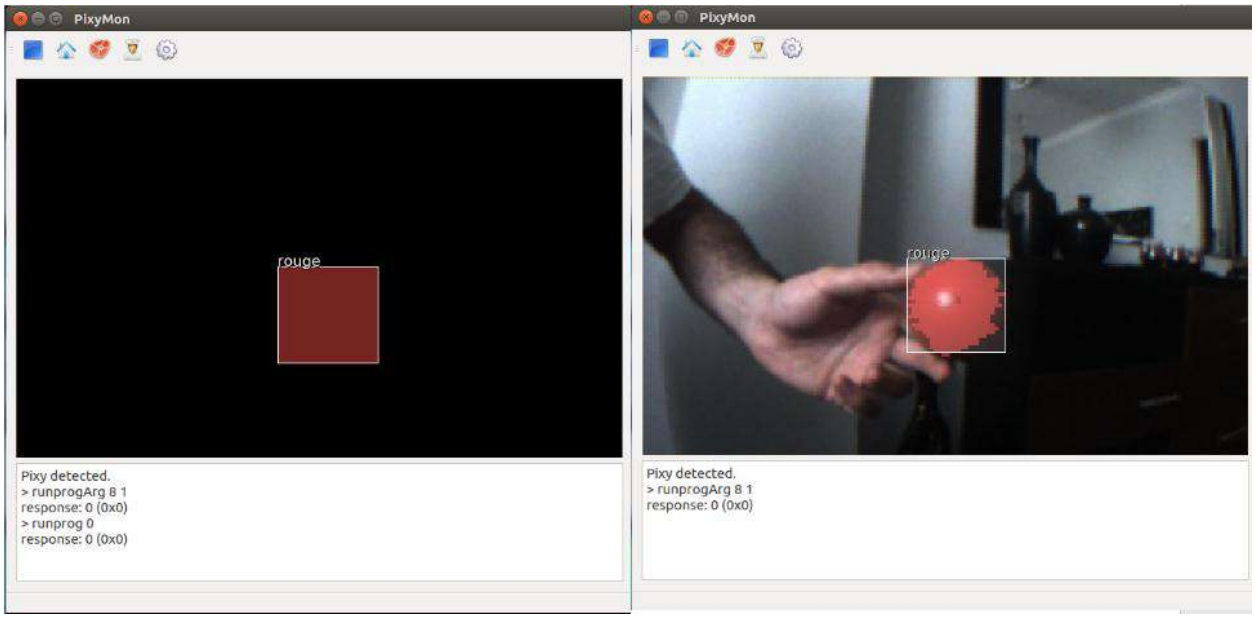
10. Description du programme d'asservissement visuel

L'organigramme précèdent décrit le fonctionnement du programme que nous avons mis en œuvre pour réaliser l'asservissement visuel du robot mobile. Dans un premier temps, nous initialisons les valeurs des paramètres. Par la suite, on va définir les limites inférieures et supérieures de la surface et la valeur X de notre cible. Ces limites vont nous permettre de commander le robot mobile vers l'avant, arrière, gauche, droite et stoppé ce dernier. Au cours de ses mouvements pour suivre la cible, si le robot rencontre un obstacle, le contrôleur visuel cesse de fonctionner et donne la main au contrôleur d'évitement d'obstacles. Ce dernier va gérer totalement le robot afin de lui faire éviter cet obstacle et le mener vers une zone dégagée. Une fois cette zone atteinte, le contrôleur visuel reprend la main pour retrouver l'objet cible en réalisant une rotation sur son axe. Une fois la cible détectée, le contrôleur visuel fournira les consignes nécessaires pour amener le robot vers celle-ci. La permutation entre les deux

contrôleurs se fait par des interruptions électroniques entre la carte Raspberry et la carte Arduino.

Note : la partie traitement d'image et la couleur de l'objet cible est réaliser indépendamment la cam Pixy.

Dans la suite de ce chapitre, nous vous présentons une chronologie de la détection d'objet cible grâce à notre système Raspberry / Pixy.



(b) (a)

Figure 4.26 : Voir ce que voit pixy.

La figure (a) à droit montre ce que Pixy voit lorsqu'il détecte l'objet cible.

La figure (b) à gauche montre ce que Pixy voit lorsqu'il détecte l'objet cible et envoi ses caractéristiques visuelles, à savoir la largeur L et la hauteur H et de sa position X dans l'image.

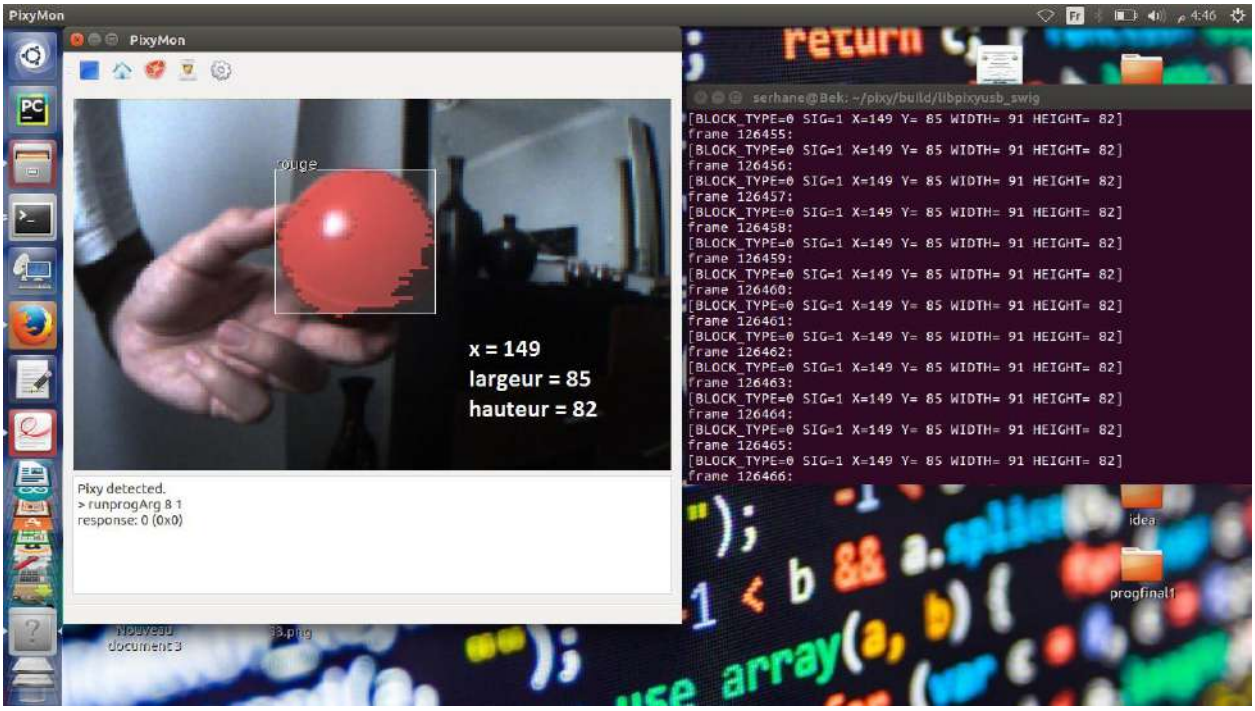


Figure 4.27 : test de détermination de largeur L, hauteur H et de X cas 1.

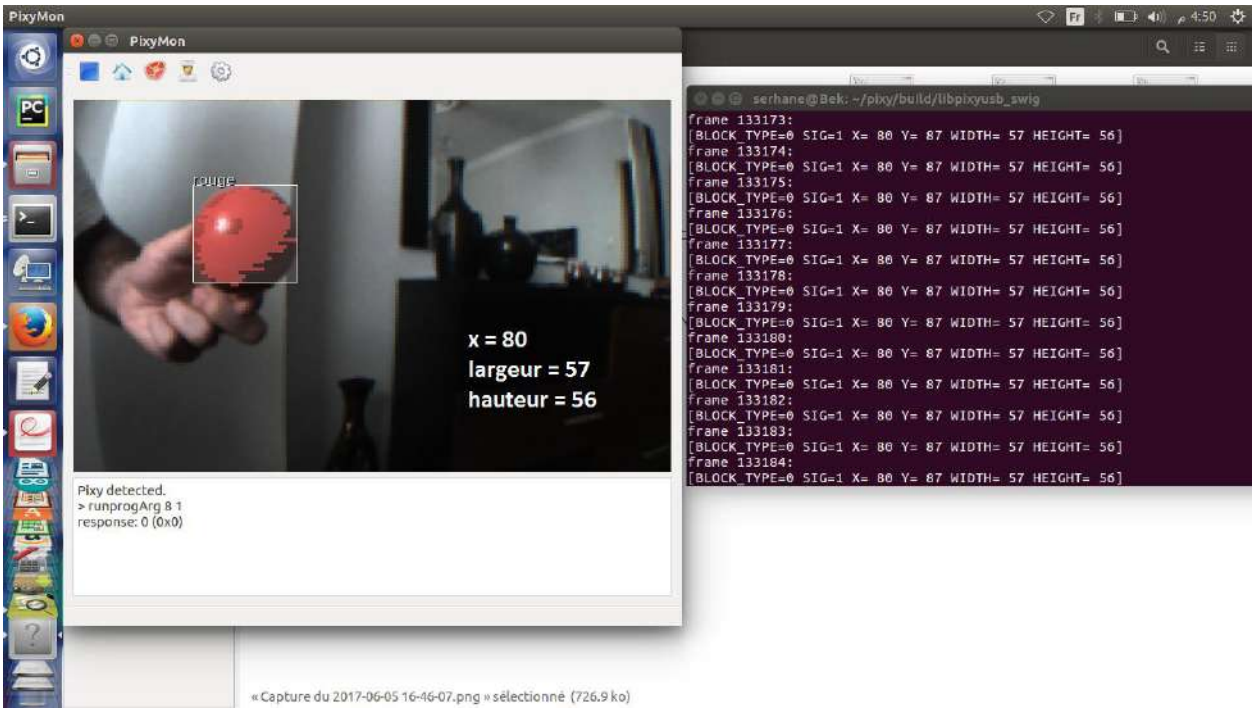


Figure 4.28 : test de détermination de largeur L, hauteur H et de X, cas 2.

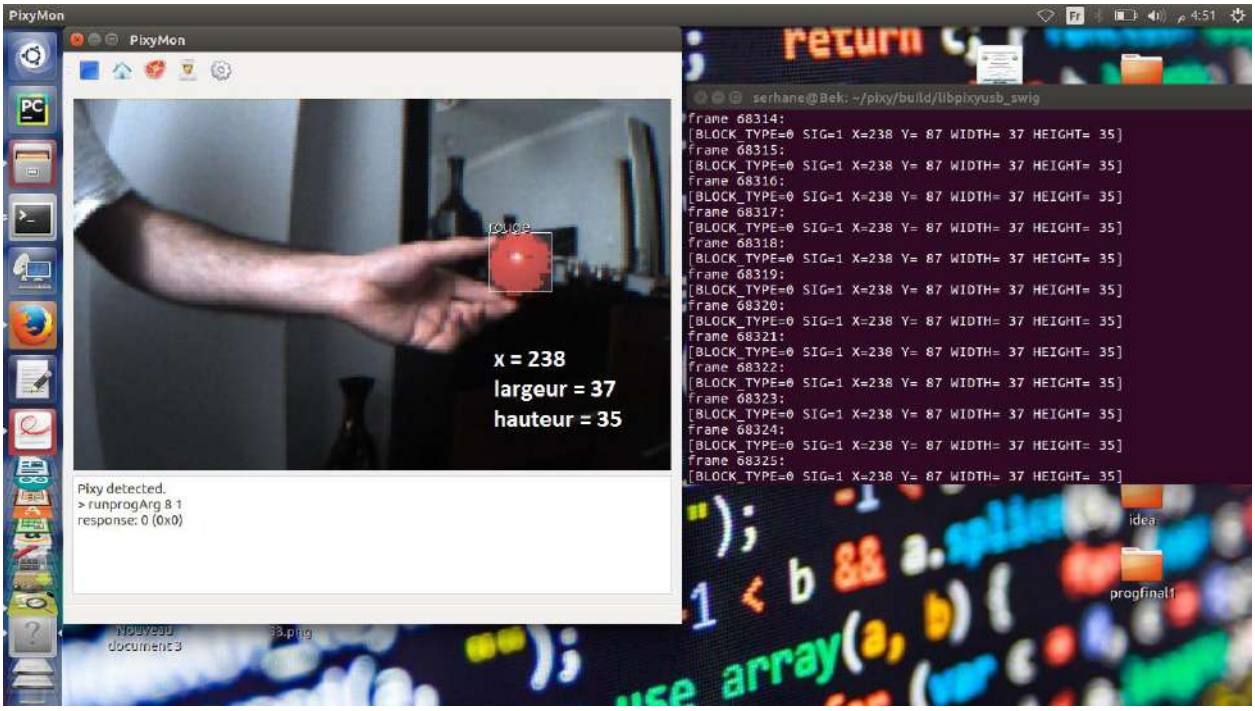


Figure 4.29 : test de détermination de largeur L, hauteur H et de X, cas 3.

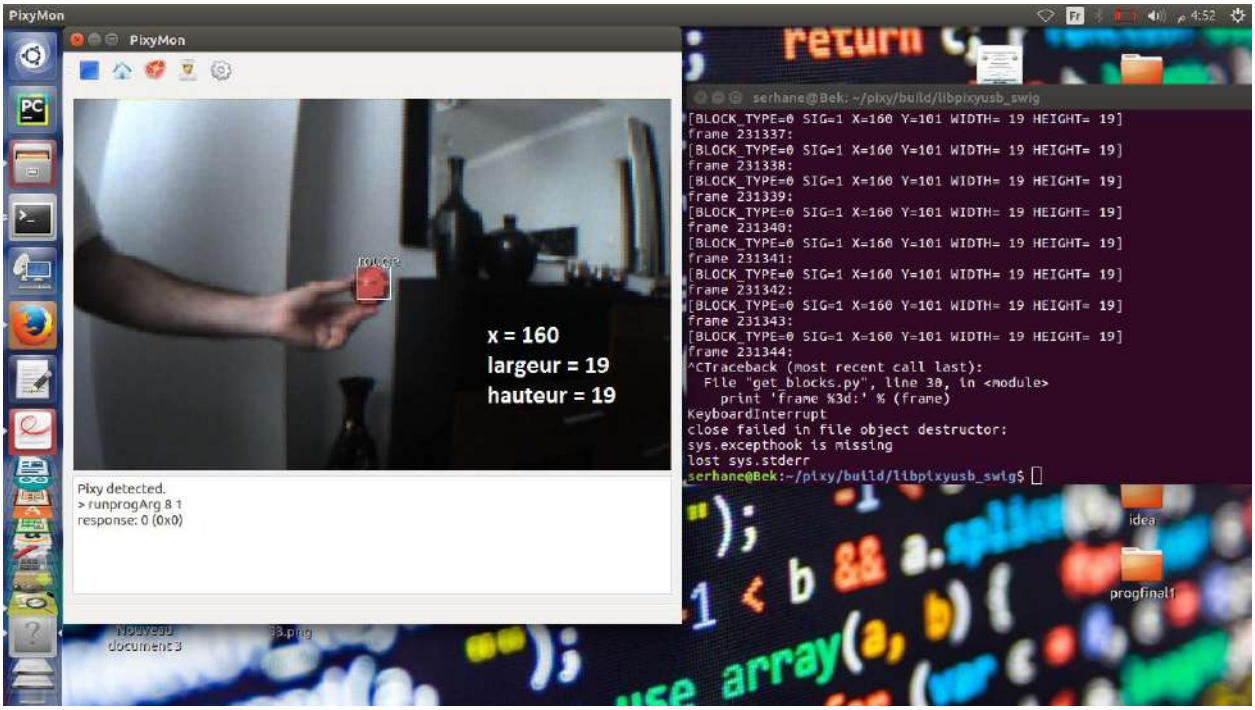


Figure 4.30 : test de détermination de largeur L, hauteur H et de X, cas 4.

A partir de ces trois données visuelles, nous pouvons commander les mouvements du robot pour maintenir la cible d'une part dans le centre de l'image (grâce à l'utilisation de la donnée X) et d'autre part maintenir la cible à une distance du robot prédéfinie (grâce à l'exploitation

du deux données L et H en calcule la surface s). En effet, plus la cible est loin du robot plus la surface de cette cible est faible, donc le contrôleur fournira aux deux moteurs une vitesse et un sens de rotation pour augmenter, dans l'espace image, la valeur de S. Une fois la valeur de référence de S atteinte, le contrôleur génèrera des consignes d'arrêt aux moteurs du robot. A la cadence vidéo, ces consignes sont automatiquement calculées est transmises via le port USB à la carte Arduino. Celle-ci est munie d'une carte extension (Shield). Cet ensemble aura pour tâche de traduire ces consignes en tensions nécessaires aux mouvements des moteurs du robot mobile.

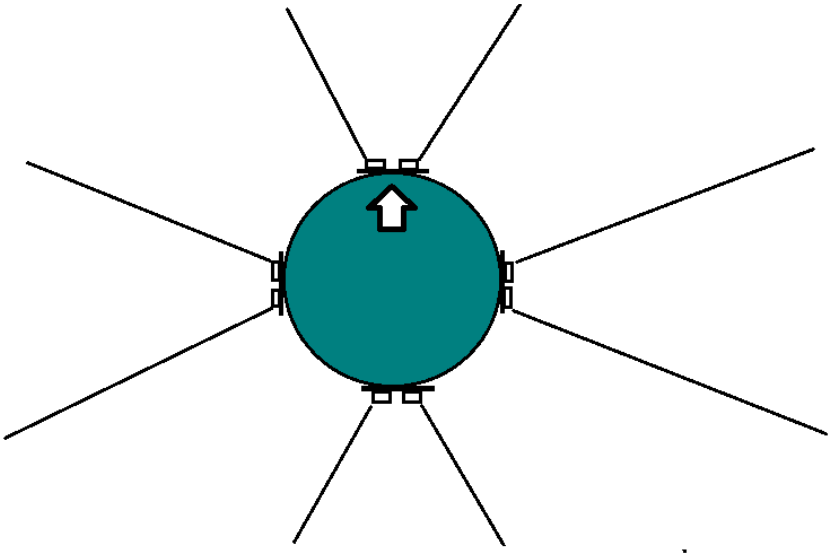


Figure 4.31 : les angles des détections dans le robot arduino mobile.

A contrario, si durant cette phase de poursuite, un obstacle apparait sur sa trajectoire, les quatre capteurs ultrason le détecterons et enclencherons automatiquement le contrôleur d'évitement d'obstacles qui calculera les consignes nécessaires à son évitement et les transmettra via le port USB à la carte Arduino.

Ce contrôleur hiérarchique a été testé in situ avec succès sur notre robot mobile. Plusieurs cibles, de couleurs différentes, ont été utilisées. Le robot a pu les suivre à chaque fois. Par ailleurs, au cours de sa poursuite de cibles, plusieurs obstacles de tailles et de textures différentes ont été introduits et notre contrôleur hiérarchique a pu efficacement les éviter et reprendre la poursuite de la cible.

Pour illustrer les performances de notre plate-forme robot mobile, nous avons réalisé plusieurs simulations et testes expérimentaux dont voici quelques résultats :

10.1. Résultat

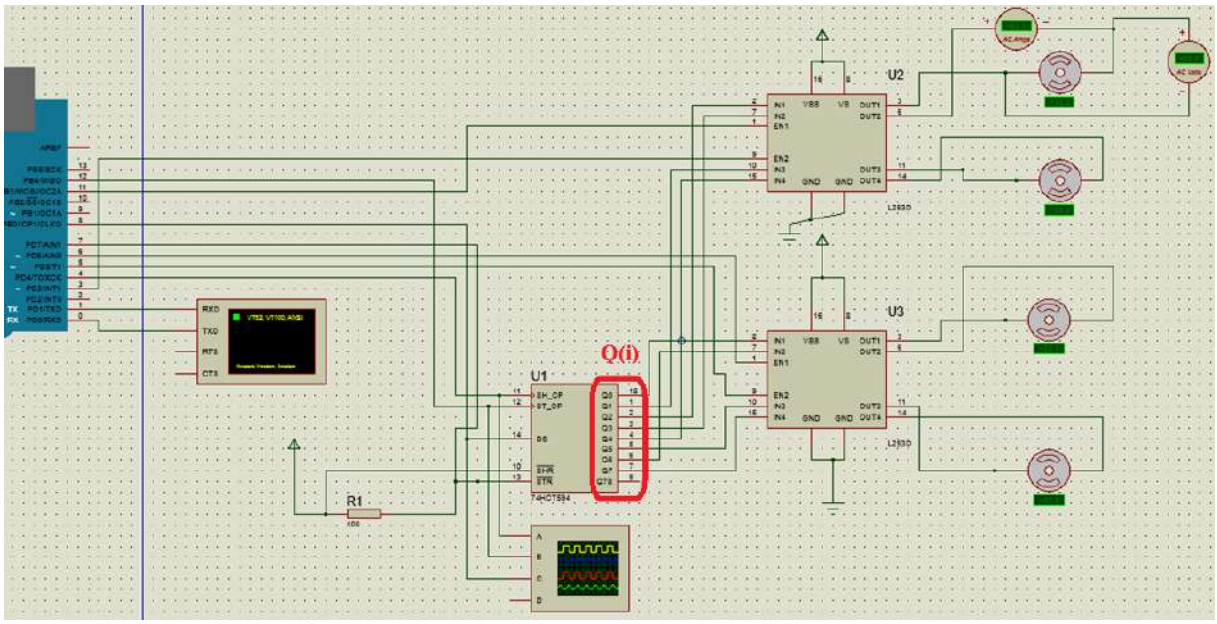
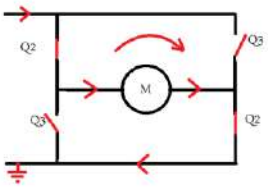
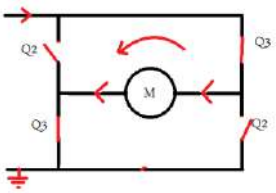
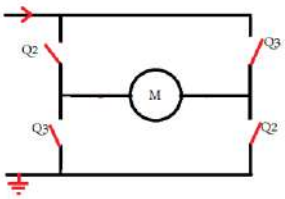


Figure 4.32 : simulation de motor shield .

Tableau 4.3 : Synoptique du système d'alimentation en pont des moteurs du robot.

	Q2=1, Q3=0	Q2=0, Q3=1	Q2=0, Q3=0
Moteur 1	 Sens de rotation (-)	 Sens de rotation (+)	 Lâcher moteur

Lors de la mise en œuvre de la commande des moteurs du robot à travers le shield, nous avons du étudié ce système pour pouvoir maitriser leur mise en marche et surtout commander les sens de rotation des deux moteurs du robot mobile.

Le tableau 4.3 montre comment en agissant sur les entrée $\mathbf{Q_i}$ du shield, nous arrivons à imposer un ses de rotation à un moteur donné.

Le tableau ci-dessous résume la gestion des sens de rotations des moteurs par les shield L293D.

Tableau 4.4: Résultat de la simulation.

Moteur 1	Q2=1	Q3=0	En avant
Moteur 1	Q2=0	Q3=1	En marche arrière
Moteur 1	Q2=0	Q3=0	Lâcher moteur
Moteur 2	Q1=1	Q4=0	En avant
Moteur 2	Q1=0	Q4=1	En marche arrière
Moteur 2	Q1=0	Q4=0	Lâcher moteur
Moteur 3	Q0=1	Q6=0	En avant
Moteur 3	Q0=0	Q6=1	En marche arrière
Moteur 3	Q0=0	Q6=0	Lâcher moteur
Moteur 4	Q5=1	Q7=0	En avant
Moteur 4	Q5=0	Q7=1	En marche arrière
Moteur 4	Q5=0	Q7=0	Lâcher moteur

Avant de commencer les expérimentations avec la plate-forme robot mobile, nous avons réalisé un certains nombres de simulation avec le logiciel Proteus.

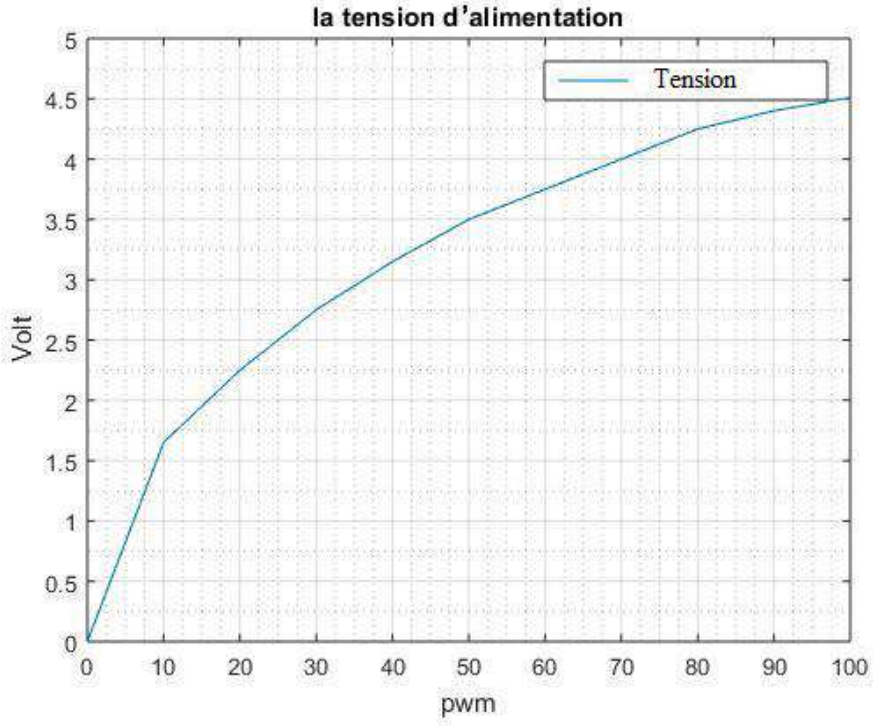


Figure 4.33 : graphe de la tension d'alimentation du moteur en fonction de PWM.

Dans le premier graphique illustré par la figure 4.33, la simulation nous montre les caractéristiques du hacheur série ou sa tension de sortie est fonction de la PWM et la tension d'entrée 5v. Nous constatons que sont évolution n'est pas linéaire et que lors de la conception de notre loi de commande nous l'avons pris en compte.

De même pour le courant qui traverse le moteur du robot, nous pouvons constater sont évolution en fonction de la PWM (figure 4.34).

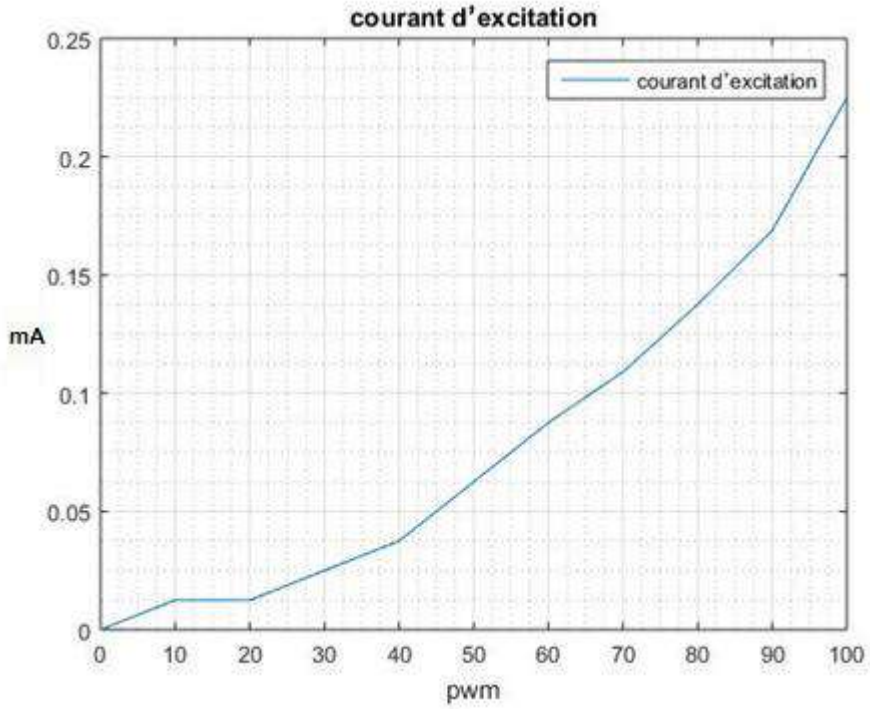


Figure 4.34 : graphe du courant d'excitation du moteur en fonction de PWM.

Ces caractéristiques ont une incidence sur le fonctionnement de la commande des deux moteurs du robot mobile. En effet, pour vaincre l'inertie du moteur au démarrage, nous devons appliquer une consigne supérieure à 30. Les tests effectués sur le robot nous ont permis de déterminer ces paramètres qui sont explicites sur le graphe de (la figure 4.35)

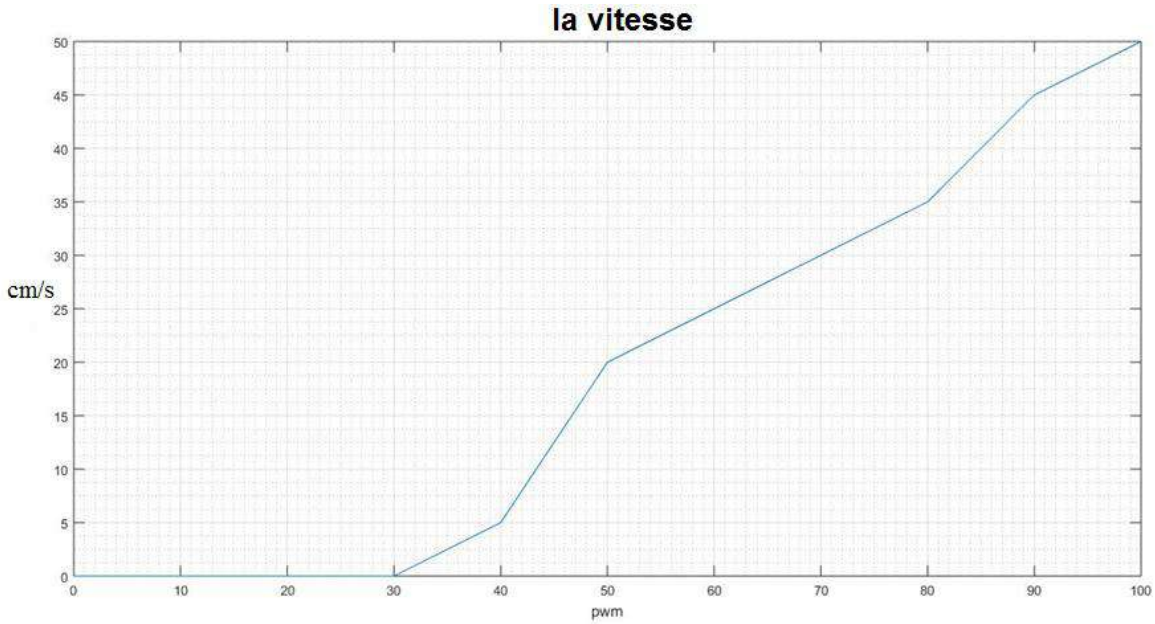


Figure 4.35 : graphe de vitesse du véhiculé on fonction de PWM.

Par ailleurs, Pour évaluer les performances de notre contrôleur visuel, nous avons réalisé une expérimentation qui consiste à présenter la cible au centre de l'image puis la faire déplacée vers la gauche de 150 pixels puis vers la droite de 300 pixels.

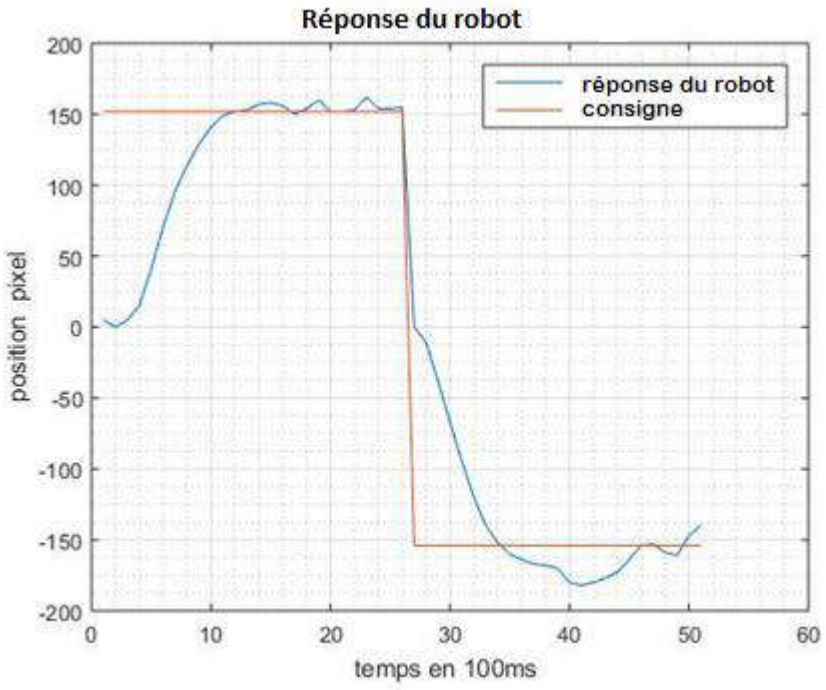


Figure 4.36 : l'évolution du suivi de cible avec le contrôleur visuel.

La figure 4.36, illustre les résultats obtenus lors de cette expérimentation. En effet, nous constatons la réponse du système robotique coïncide avec celle de la consigne tout en sachant

que nous utilisant un simple proportionnel pour corriger le comportement du système. Ces performances peuvent vraiment être augmentées en réalisant des correcteurs plus évolués.

11. Conclusion

Dans ce chapitre, nous avons présenté le fonctionnement détaillé et les modifications majeures apportées à la plate-forme robot mobile pour réaliser le suivi d'une cible à partir de données visuelles en evitant d'eventuels d'obstacles qui peuvent se présentés sur sa trajectoire. Dans un premier temps, nous avons explicité l'agencement et l'organisation de notre système embraqué composé du système de vision Pixy de la carte Raspberry Pi et de la carte Arduino / Shield. Celles-ci grâce à leur interconnexion, via le port USB, communiquent et s'échangent les données nécessaires à la commande du robot mobile.

Dans un second temps, nous avons détaillé le rôle de chaque étage. Le plus important est l'acquisition, le traitement des données visuelles avec le système Pixy ainsi que le fonctionnement des capteurs ultrason. Ces opérations ainsi que la loi de commande ont été regroupées dans un seul programme écrit en Python et exécute sur la carte Raspberry. Les performances de cette carte ont permis la réalisation d'un contrôleur hiérarchique visuel + évitement d'obstacle performant.

Enfin, l'application expérimentale que nous avons mise en œuvre a permis de mettre en exergue d'une part, nos choix quant à l'utilisation de ces cartes embarquées et d'autre part notre algorithme de commande basée sur un schéma d'asservissement visuel fondé sur l'approche image. Bien entendu, d'autres tests sont à prévoir pour étudier de plus près les performances de notre contrôleur. Mais en tout état de cause, et au vu des résultats obtenus, nous avons rempli les conditions qui nous ont été imposées, à savoir la conception et la réalisation d'une plate-forme évolutive, puissante et embraquée pour asservir un robot mobile à partir de données visuelles.

Conclusions et Perspectives

Au terme de ce travail, nous avons apporté des modifications majeures à l'architecture matérielle embarquée sur la structure mécanique du robot mobile. L'augmentation de ses performances a pu être obtenu grâce à l'utilisation d'un nouveau système de vision Pixy et la réorganisation du fonctionnement des cartes électronique Raspberry Pi et Arduino. En effet, la combinaison de ces deux cartes nous a permis de développer une plate-forme capable de commander tout type de robot tricycle à condition d'avoir accès aux entrées de ces moteurs qu'ils soient à courant continu, pas à pas ou de type servomoteur. De plus, l'introduction de nouveaux capteur proprioceptifs (capteurs ultrason) a permis à ce robot mobile d'éviter les obstacles présents sur sa trajectoire.

Par ailleurs, notre utilisation de la carte Raspeberry dans l'asservissement visuel de notre robot mobile a permis de démontrer ses performances en termes de puissance de calcul, de mémoire et de connectivité (USB, I2C, WIFI,).

D'autre part, l'application expérimentale que nous avons mise en œuvre nous a valider le fonctionnement de cette nouvelle version de la plate-forme mais aussi de mettre en évidence les performances de l'algorithme de commande que nous avons élaboré. En effet, nous avons mis en œuvre un contrôleur capable d'asservir les mouvements d'un robot mobile uniquement à partir d'informations visuelles. L'introduction du nouveau système de vision Pixy, nous a permis d'économiser des ressources informatiques au niveau du Raspbery. Ce dernier pourra désormais consacrer toutes ses ressources à l'exécution des lois de commandes complexes. Cette première évolution ouvre, à l'équipe du laboratoire **L.S.E.A** de nouvelles perspectives dans le domaine de la conception de lois de commandes robustes ou intelligentes.

La deuxième évolution majeure que nous avons apportée et l'introduction de quatre (4) capteurs ultrason fournissant les informations nécessaires à l'évitement d'obstacles. Ces modifications roderont le robot mobile plus autonome.

Eu égard des résultats obtenus lors de nos divers tests, nous avons pu valider le fonctionnement du nouvel ensemble, algorithme et plate-forme matérielle.

Enfin, ce travail nous a permis d'aborder les différentes problématiques inhérentes à la robotique mobile. Mais surtout, de nous initier à l'utilisation de cartes électroniques embarquées, à la programmation en langage Python et aux techniques de traitements d'images.

Quant aux perspectives futures que nous envisageons pour compléter ce travail :

- La conception d'une tète robotique motorisée capable d'orienter le système de vision Pixy.

- Remplacer la carte Raspberry par une autre carte plus performante encore (carte FPGA par exemple).
- Créer une interface de type web pour gérer et récupérer des informations sur le fonctionnement du robot.

Bibliographie

[1] Damien LHOMME-DESAGES, "*Commande d'un robot mobile rapide à roues non directionnelles sur sol naturel* ", Thèse de doctorat, l'Université Paris 6, 12 Apr 2013

[3] Alain COURCELLE, "*Localisation d'un robot mobile* : *Application à l'aide à la mobilité des personnes handicapées moteur*", Thèse de doctorat, l'Université de Metz, France, janvier 2000.

[4] M. Drum heller, "*Mobile Robot Localization Using Sonar*", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 9, n°2, march 1987, pp. 325-332.

[5] Cyril Drocourt, "*Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels*", Thèse doctorat, l'Université de Technologie de Compiègne, février 2002.

[7] David FILLIAT, "*Robotique Mobile*", Cours C10-2, ENSTA, France, Octobre 2004.

[8] Oualid Djekoune. " *Localisation Et Guidage Du Robot Mobile Atrv2 Dans Un Environnement Naturel Robotique*", Thèse doctorat, Université des Sciences et de la Technologie Houari Boumediene, 23 Feb 2012

[12] BENAISSA el mahdi, BENYAIHIA Dalal, " *Asservissement Visuel d'un robot mobile*", Projet de fin d'étude, Université Yahia Fares de Médéa, année 2016

[15] William JACQUET, "*Paradigme connexionniste pour l'apprentissage de fonctions robotique : application aux problèmes d'asservissement visuel de bras manipulateurs*", Thèse doctorat, l'Université de Haute-Alsace, année 2001

[16] Zoulikha BOUHAMATOU, "*Navigation visuelle d'un robot mobile*", mémoire de magister, l'Université de Batna, 22 apr 2015

Webographie

[2] Mazen Al Homsi, " *main humaine, main robotique, qu'elle est la différence ?* ", UTBM, 2016
http://http://mrmhutbm.e-monsite.com/pages/apercu-historique.html

[6] Arduino, "*Introduction*", https://www.arduino.cc/en/Guide/Introduction

[9] locoduino, " *Alimentation d'arduino*", http://www.locoduino.org/spip.php?article16

[10] Raspberry pi, "*Raspberry pi 2 model B*, raspberry pi foundation UK registred charity 112909
https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[11] Arduino, "*Arduino Board Uno*", https://www.arduino.cc/en/main/arduinoBoardUno

[13] Arduino, "*Arduino Due*", https://www.arduino.cc/en/Main/ArduinoBoardDue

[14] Anthony Rowe, Rich LeGrand, Scott Robinson, "*Aperçu sur CMUcam5 Pixy*", http://cmucam.org/projects/cmucam5

[17] ATMEL, "8-*bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash*"
http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdfasd Chapter 21 "2-wire Serial Interface"

[18] Higashi Kotari, Nagaokakyoshi, "*Ultrasonic Sensor*", Kyoto japan, http://www.murata.com/

[19] Edward Getz, "*Teach Pixy an Object*", 08/10/2015
http://cmucam.org/projects/cmucam5/wiki/Teach_Pixy_an_Object_