Université ABBES LAGHROUR Khenchela
Faculté des Sciences et de la Technologie
Département de Génie Industriel
جامـــعة عبـــاس لـــــغرور خـنـشـلة
كليـــة العلـــوم والتكنـــولـــوجيـا
قسم الهنـدسة الصنـاعية

N° Série :.....

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master

Filière : Télécommunications
Spécialité : Systèmes des Télécommunications

THEME

Etude et conception d'un détecteur d'objet en utilisant la communication sans fil, équipé d'un Raspberry *Pi* et une caméra

Réalisé par **:** - *ABIDI Abdelatif*

- *ZEROUAL Samira*

Soutenu le : 29/06/2019 ***Devant le jury composé de:***

Mme. MEDJALDI Malika	***Présidente***	***Université Abbes Laghrour-Khenchela***
Mr. DOUAK Fouzi	***Encadreur***	***Université Abbes Laghrour-Khenchela***
Mr. SAIGAA Mohamed	***Examinateur***	***Université Abbes Laghrour-Khenchela***

Promotion 2018/2019

Je dédie ce travail à :

Ma mère,

Mon père,

Mes frères et mes sœurs

Tous mes collègues de promotion,

Abdelatif

Je dédie ce mémoire à :

Mes parents, Mes frères et mes sœurs

Tous mes collègues de promotion,

Samira

Remerciements

Nous tenons à remercier ALLAH tout puissant qui nous a donné la santé, le courage, et la volonté pour réaliser ce travail.

Nous remercions notre encadreur Monsieur **Fouzi Douak**, *enseignant à l'université de Khenchela, pour son suivi, pour son aide précieux.*

Nous adressons nos plus sincères remerciements aux examinateurs, pour avoir accepté de juger ce travail, et qui nous font l'honneur d'être membres de ce Jury.

Nous tenons à remercier particulièrement tous les enseignants et les responsables de notre département qui ont veillé sur notre formation.

Nos remerciements vont également à tout le personnel du département du génie industriel pour leur gentillesse et leur disponibilité.

Enfin, nous exprimons notre remerciement à tous ceux qui ont contribué de prés ou de loin à l'aboutissement de ce travail.

Résumé

Dans ce travail nous présentons l'étude et la réalisation d'un suiveur visuel d'objets par un capteur d'image. Pour atteindre cet objectif, nous avons utilisés plusieurs techniques de prétraitement et seuillage. Le prétraitement des images (transformation d'espace couleur) sont recommandés pour ce genre de détection d'objets. Dans ce contexte, nous avons proposé une méthode basée sur la projection des données dans les différentes espaces couleurs (RGB, YCbCr, YIQ, HSV, HSI).

Comme application, nous avons utilisé un robot mobile qui fonctionne en mode autonome. Ensuite la conception et programmation des circuits ont été faites en utilisant deux types des cartes : la carte Arduino (Mega ou Nano) qui est programmé dans le logiciel Arduino IDE, et la carte Raspberry *Pi* 3, dans cette dernière l'environnement de programmation est le logiciel MATLAB-SIMULINK. Le robot mobile est muni d'une communication Wi-Fi pour accéder au réseau local, une caméra pour transmettre une vidéo en temps réel à un ordinateur qu'on peut accéder à distance, et une transmission radiofréquence.

Mots clés: Détection d'objets, Suiveur d'objets, Changement d'espace couleur, Seuillage, Arduino Mega, Arduino Nano, Raspberry *Pi* 3.

Abstract

In this work we present a study and realization of the visual objects tracking by an image sensor. To achieve this goal, we used several preprocessing and thresholding techniques. Image preprocessing (color space transformation) is recommended for this kind of object detection. In this context, we proposed a method based on the projection of data in the different color spaces (RGB, YCbCr, YIQ, HSV, and HSI).

As an application, we used a mobile robot that works in stand-alone mode. Then the design and programming circuits were made using two types of cards: the Arduino board (Mega or Nano) that is programmed in the Arduino IDE software and the Raspberry *Pi* 3 cards, in the latter the programming environment is the MATLAB-SIMULINK software. The mobile robot is equipped with Wi-Fi communication to access of the local network, a camera to transmit a video in real time to a computer that can be accessed remotely, and a radio frequency transmission.

Keywords: Object detection, Object tracking, Color space transformation, Thresholding, Arduino Mega, Arduino Nano, Raspberry *Pi* 3.

ملخص

في هذا العمل قمنا بدراسة نضريه و تطبيقية للمتابع البصري للأشياء بواسطة مستشعر للصورة، لتحقيق هذا الهدف استخدمنا العديد من تقنيات معالجة الصورة المسبقة بالإضافة لاستعمال الحد الأدنى للصورة. في هذا المجال يحبذ باستعمال تحويلات الألوان إلى فضاء أخر أكثر ملائمة . في هذا السياق، اقترحنا عدة طرق تعتمد على إسقاط الألوان في فضاء مختلف (YIQ, HSI, HSV, YCbCr, RGB).

كتطبيق لما ذكر سابقا، استعملنا آلة متحكمة يعمل في وضع مستقل، ثم تم تصميم برامج لهذا الأخير باستخدام نوعين من البطاقات المبرمجة أردوينو و الراسبيري iP 3، البطاقة الأولى نستخدم برنامج الأردوينو IDE و الثانية MATLAB-SIMULINK . في هذه الآلة توجد تقنية Wi-Fi للاتصال بالشبكات، و كاميرا لنقل الفيديو إلى جهاز الكمبيوتر الذي يمكننا بالتحكم به عن بعد، واستعملنا تقنية أخرى للاتصالات مبنية على ترددات الراديو.

الكلمات المفتاحية: المتتبع البصري، تحويل فضاء الألوان، قيمة الحد الأدنى للصورة، الأردوينو،راسبيري.

Liste des tableaux

Liste des figures

Liste des symboles

AI Adobe illustrator.

BMP BitMap.

CIE Commission internationale de l'éclairage.

CE Chip Enable.

CSN Chip Select Not.

DC Direct Current.

dpi dots per inch (points par pouce).

EPS Encapsulated PostScript.

E/S Entrée/Sortie.

EEPROM Electrically Erasible Programmable Read-Only Memory.

GIF Graphical Interchange Format.

GPIO General Purpose Input Output.

HSV teinte (H), saturation (S), valeur (V).

HSI teinte (H), saturation (S), intensité (I).

I2C (IIC) Inter Integrated Circuit.

IDE Integrated Development Environment.

IP Internet Protocol.

IRCC International Radio Consultative Committee.

JPEG Joint Photographic Experts Group.

LCD Liquid Crystal Library.

Mac OS Macintosh Operating System.

NTSC National Television System Committee.

PAL Phase Alternating Line.

PWM Pulse Width Modulation.

PNG Portable Network Graphics.

PDF Portable Document Format.

PICT Picture Format.

PPM Portable Pix Map Format.

RGB Reed Green Blue.

RVB Rouge Vert Bleu.

RVBA ou RGBA (Rouge Vert Bleu Alpha).

RF Radio Fréquence.

RAM Random Access Memory.

SDA Serial Data Line.

SCL Serial Clock Line.

SRAM static random access memory.

SPI Serial Peripheral Interface.

SSH Serveur Secure Shell.

SVG Scalable Vector Graphics.

SRAM Static random access memory.

SCLK Serial Clock, Horloge.

MOSI Master Output, Slave Input.

MISO Master Input, Slave Output.

SS Slave Select.

TTL Transistor-Transistor Logic.

UART Universal Asynchronous Receiver/Transmitter.

USB Universal Serial Bus.

TIFF Tagged Image File Format.

Wi-Fi Wireless Fidelity.

YCbCr Luminance (Y), Chrominance (Red-Yellow); Chrominance (Blue-Yellow).

YUV luminance (Y), Saturation (U), Teinte (V).

Table des matières

Chapitre II

Conception des shields Arduino pour la détection d'objets

Chapitre III

Traitement d'images sur Raspberry Pi

Chapitre IV

Réalisation pratique

Introduction générale

Introduction générale

Le suivi d'objets dans une séquence d'images occupe une place prépondérante dans plusieurs domaines en relation avec la vision artificielle : surveillance, la compression, l'imagerie médicale, la robotique, l'interaction homme machine, …etc [1]. Le but principal de la vision artificielle est de permettre à une machine de comprendre ce qu'elle «voit» lorsqu'on la connecte à une ou plusieurs caméras. Elle peut servir entre autre à la reconnaissance de formes, qui consiste à reconnaitre une forme dans une image après l'avoir enregistrée.

L'analyse de détection d'objets est un vaste sujet qui englobe un certain nombre de problématiques: (*i*) la détection des objets en mouvement, c'est-à-dire la détection d'un ensemble de régions d'intérêt en mouvement dans la scène observée, (*ii*) le suivi de primitives ou de régions, dont le but est de déterminer la position de chaque région dans l'image à chaque instant [2].

La problématique de détection des objets en mouvement, est en général une première étape pour des outils automatiques de la vision artificielle. Ces outils peuvent avoir pour vocation, soit uniquement de détecter, soit de détecter et reconnaître, soit de détecter et suivre des objets. Dans ce mémoire notre objective est la détection et le suivi d'objets couleur. Pour cela nous avons utilisé comme application un robot mobile. La construction principale comprend le Raspberry Pi 3, un Arduino Nano, un capteur ultrasons et une caméra. Le robot est muni d'une communication Wi-Fi pour accéder au réseau local et une caméra pour transmettre une vidéo en temps réel à un ordinateur qu'on peut accéder à distance. D'autre part, nous avons utilisé une communication sans fil à base des ondes radio fréquence pour transmettre les données (distance entre l'objet et le robot, la détection d'objet, la couleur de l'objet détecté).

Un des problèmes majeurs de la robotique mobile est la planification de mouvement et aussi comprendre l'environnement (image ou les objets). L'autre problématique est aussi importante et nécessite la mise en place des méthodes simples et robustes pour la détection d'objets. Dans notre travail, nous avons étudié deux méthodes de détection des objets dans une sciène vidéo, ces méthodes basées sur un simple seuillage, et un seuillage adaptative. Nous sommes intéressés par la méthode de détection des objets selon leur couleur et leur espace couleur.

Les images couleurs sont traités sous le format RGB (Red Green Blue : Rouge Vert Bleu). Or le système RGB n'est pas le mieux adapté pour le processus de détection d'objets, puisqu' il y a une forte corrélation entre les couleurs rouge, vert et bleu; mais avec un changement d'espace de couleur RGB vers un autre espace moins corrélé, par exemple HSV. Cette

transformation permet une représentation plus efficace des données. Il est clair que les résultats obtenus après application de la transformation de couleur en terme de taux de classification ou reconnaissance sont mieux que ceux obtenus par application directe de la détection d'objet sur une image RGB, mais la question qui mérite d'être posée : est-il de même pour les autres transformations YCbCr [3, 4], HSV [5], HIS [6], YIQ [7], O1O2O3 [8],… etc. Pour répondre à cette question, nous avons utilisé plusieurs types des espaces couleur avec l'intermédiaire d'une stratégie de seuillage, et ensuite trouvé le meilleur espace couleur adapté à la détection d'objet couleur.

L'organisation générale du mémoire est décrite ci-dessous :

Le chapitre I est consacré à la présentation des différents types d'images (binaires, niveaux de gris, couleur) et aussi le changement d'espace couleur.

Le chapitre II sera consacré à une étude approfondie sur les cartes Arduino, sa construction son environnement de programmation et son principe de fonctionnement afin de simplifier son utilisation. Ensuite, on mettra la lumière sur les cartes d'extension (shields) utilisé dans notre travail.

Le chapitre III traite deux parties. Dans la première partie nous présentons les différentes caractéristiques de la carte Raspberry *Pi* 3, dans la deuxième partie, nous présentons deux stratégies de détection d'objets couleurs basé sur le seuillage et le changement d'espace couleur.

Le chapitre IV conclut ce mémoire en présentant l'ensemble des résultats expérimentaux obtenus pour la détection et le suivi d'objets.

La dernière partie de ce mémoire est consacrée à la conclusion générale.

Chapitre I

Etat de l'art sur le traitement d'image

I.1. Introduction

La technologie numérique moderne a rendu possible la manipulation de données multidimensionnelles des signaux avec des systèmes allant de simples circuits numériques à des systèmes parallèles avancés des ordinateurs [9]. Il existe une demande croissante de traitement d'images dans divers domaines d'application, tels que l'informatique multimédia, la communication sécurisée de données d'images, l'imagerie biomédicale, la biométrie, la télédétection [10], la compréhension de la texture, la reconnaissance de formes, la récupération d'images basée sur le contenu, la compression [11], etc.

Nous sommes au milieu d'un monde enchanté visuellement, qui se manifeste avec une variété de formes, de couleurs et de textures, de mouvement et de tranquillité. La perception humaine a la capacité d'acquérir, d'intégrer, d'interpréter toutes ces informations visuelles abondantes autour de nous. Il est difficile de transmettre de telles capacités à une machine afin d'interpréter les informations visuelles intégrées aux images fixes, graphiques et vidéo ou aux images en mouvement dans notre monde sensoriel. Il est donc important de comprendre les techniques de stockage, traitement, transmission, reconnaissance, et enfin interprétation de ces images des scènes. Ce qui nous rend capable de travailler sur ce qu'on appelle «Le traitement d'image».

I.2. Image numérique

Une image numérique est une représentation d'une image réelle sous la forme d'un ensemble de nombres pouvant être stockés et gérés par un ordinateur numérique. Pour traduire l'image en chiffres, elle est divisée en petites zones appelées pixels (éléments d'image). Pour chaque pixel, le dispositif d'imagerie enregistre un nombre, ou un petit ensemble de nombres, décrivant certaines propriétés de ce pixel.

Le traitement d'image numérique consiste à utiliser des algorithmes informatiques pour effectuer le traitement d'image sur des images numériques. Comme un sous-domaine du traitement du signal numérique, le traitement des images numériques présente de nombreux avantages par rapport au traitement des images analogiques ; il permet d'appliquer une gamme beaucoup plus large d'algorithmes aux données d'entrée et d'éviter des problèmes tels que l'accumulation du bruit et de la distorsion du signal pendant le traitement. Puisque les images sont définies sur deux dimensions (peut-être plus), le traitement numérique des images peut être modélisé sous la forme de systèmes multidimensionnels.

I.3. Caractéristiques d'une image numérique

On distingue généralement plusieurs caractéristiques qui définissent l'image numérique que l'on veut traiter.

I.3.1. Dimensions

La taille d'une image numérique est déterminée par ses dimensions (*M*x*N*) multipliées par le nombre de bits *b* nécessaires pour stocker le niveau d'intensité ($L = 2^b$).

Les valeurs typiques de *b* sont:

- *b* =1 ➔ image en noir et blanc (binaire).
- *b* =8 ➔ images en niveaux de gris (256 niveaux de gris) ou en couleur.
- *b* =24 ➔ images couleur RVB.

Les images peuvent être affichées dans leur intégralité en agissant sur un facteur de zoom, mais leur taille réelle n'est pas modifiée.

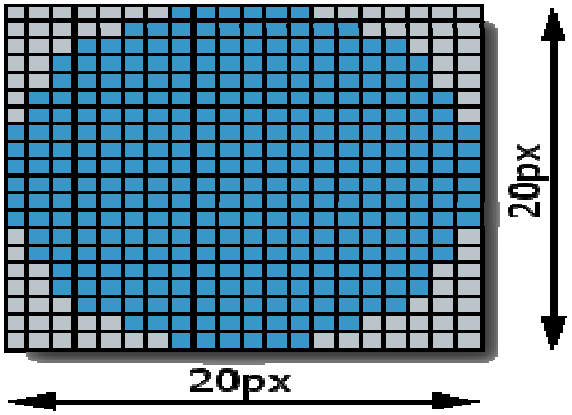


Figure I. 1 La dimension d'une image.

I.3.2. Résolution

La résolution est le détail d'une image. Le terme s'applique aux images numériques matricielles, aux images de film et à d'autres types d'images.

C'est ce qui lie la taille à la dimension. Cela correspond à la quantité de pixels qu'il y a sur une longueur donnée. Cette valeur s'exprime en ppp (point ou pixel par pouce) traduit en Anglais par dpi (*dot* per *inch*) [12]. Un pouce mesure 2.54 cm. Plus cette valeur est élevée, plus la densité de pixels est importante et plus l'image à du détail. Afin de réduire la place de stockage d'une image, on peut réduire sa résolution, c'est-à-dire diminuer le nombre de pixels.

La façon la plus simple d'effectuer cette réduction consiste à supprimer des lignes et des colonnes dans l'image de départ.

La résolution *Rs* est calculée à partir de cette formule :

$$Rs = \frac{D}{DM} \tag{I.1}$$

où *D* : définition et *DM* dimension.

La résolution permet ainsi d'établir le rapport entre le nombre de pixels d'une image et la taille réelle de sa représentation sur un support physique (impression d'une photo).

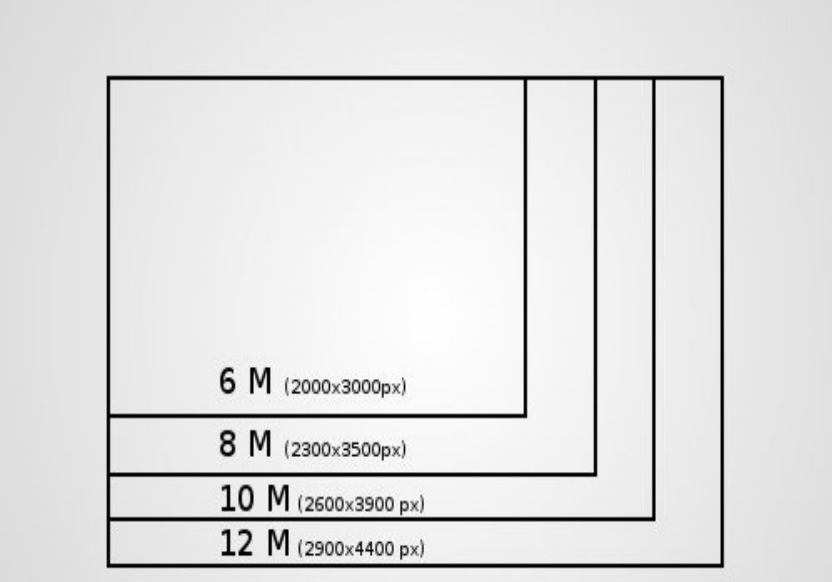


Figure I. 2 La résolution d'une image.

I.3.3. Histogramme

Un histogramme est une courbe statistique indiquant la répartition des pixels selon leur valeur. Il est très utile pour contrôler l'exposition d'une image.

Un histogramme d'image est un type d'histogramme qui agit comme une représentation graphique de la distribution tonale dans une image numérique. Il trace le nombre de pixels pour chaque valeur tonale. Pour le traitement, il permet de corriger ou modifier l'exposition de l'image, ainsi que l'échelle des couleurs.

L'axe horizontal du graphique représente les variations de mesurant, tandis que l'axe vertical représente le nombre de pixels de ce mesurant particulier, donc ce dernier représente la taille de la zone capturée dans chacune de ces zones.

Pour des images en niveau de gris, l'histogramme indique pour chaque valeur entre le noir (0) et le blanc (255), combien il y a de pixels de cette valeur dans l'image; en abscisse (axe x) le niveau de gris (de 0 à 255) et en ordonnée (axe y) le nombre de pixels. Les pixels sombres

apparaissent à gauche de l'histogramme, les pixels clairs à droite de l'histogramme et les pixels gris au centre de l'histogramme. Et pour les images couleurs (Figure I.3), plusieurs histogrammes sont utilisés comme l'histogramme des luminances et les 3 histogrammes de chacune des composantes R, V et B [13].

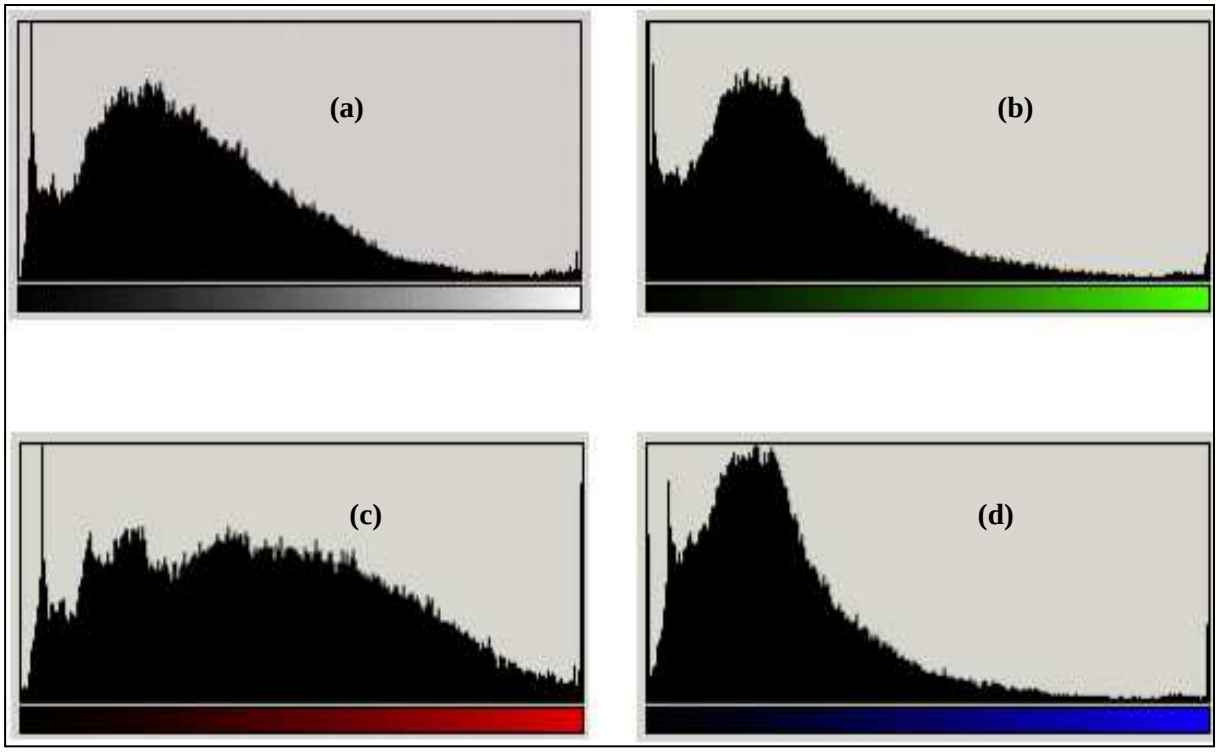


Figure I. 3 Exemple d'histogrammes pour une image couleur, (a) luminances et (b), (c), (d) les 3 histogrammes de chacune des composantes R, V et B.

I.3.4. Luminance

La luminance est une grandeur correspondant à la sensation visuelle de luminosité d'une surface. Une surface très lumineuse présente une forte luminance, tandis qu'une surface parfaitement noire aurait une luminance nulle. Luminance ou brillance d'une image est définie comme la moyenne de tous les pixels de l'image. Pour augmenter la luminance, il suffit de décaler l'histogramme : $I_0(i, j)=I(i, j)+b$. Dans les deux images représentée dans la Figure I.4, seule la luminance est différente [14]**.**

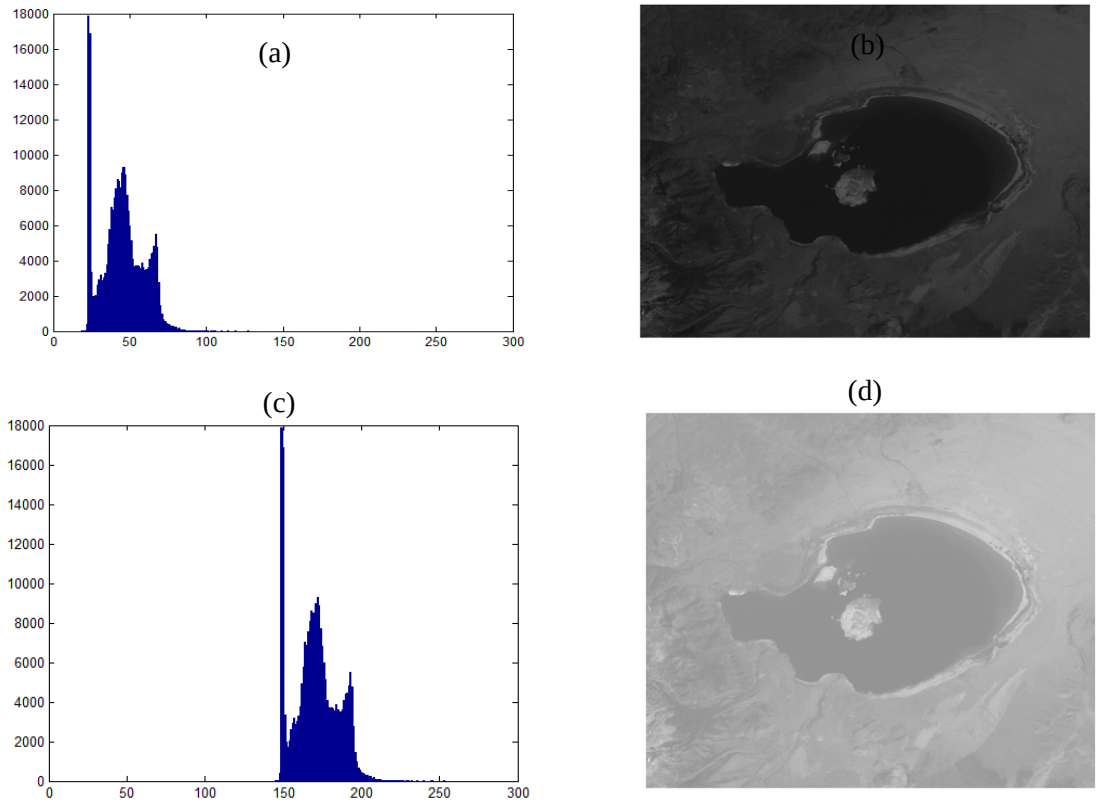


Figure I. 4 Histogramme représentant la répartition lumineuse d'une image, (a) l'histogramme de l'image original, (b) l'image originale, (c) l'histogramme décalé, (d) luminance modifie.

I.3.5. Contraste

Le contraste est la différence de propriétés visuelles qui rend un objet (ou sa représentation dans une image) distinguable, plus sombre ou plus lumineux des autres objets et de l'arrière-plan. En visuel perception du monde réel, le contraste est déterminé par la différence dans la couleur et la luminosité de l'objet et des autres objets dans le même champ de vision. Parce que le système visuel humain est plus sensible au contraste qu'à la luminance absolue, nous pouvons percevoir le monde de la même manière indépendamment des énormes changements dans l'éclairage sur la journée ou d'un lieu à l'autre. Donc, le contraste est une propriété intrinsèque d'une image qui quantifie la différence de luminosité entre les parties claires et sombres d'une image.

Les images ayant un niveau de contraste plus élevé présentent généralement un degré de variation de couleur ou de niveaux de gris plus élevé que celles présentant un contraste plus faible. On peut calculer le contraste *C* par la formule :

$$C = \frac{DL}{L_m} \qquad \text{(I.2)}$$

ou *DL* est la différence de luminance, et L_m la luminance moyenne.

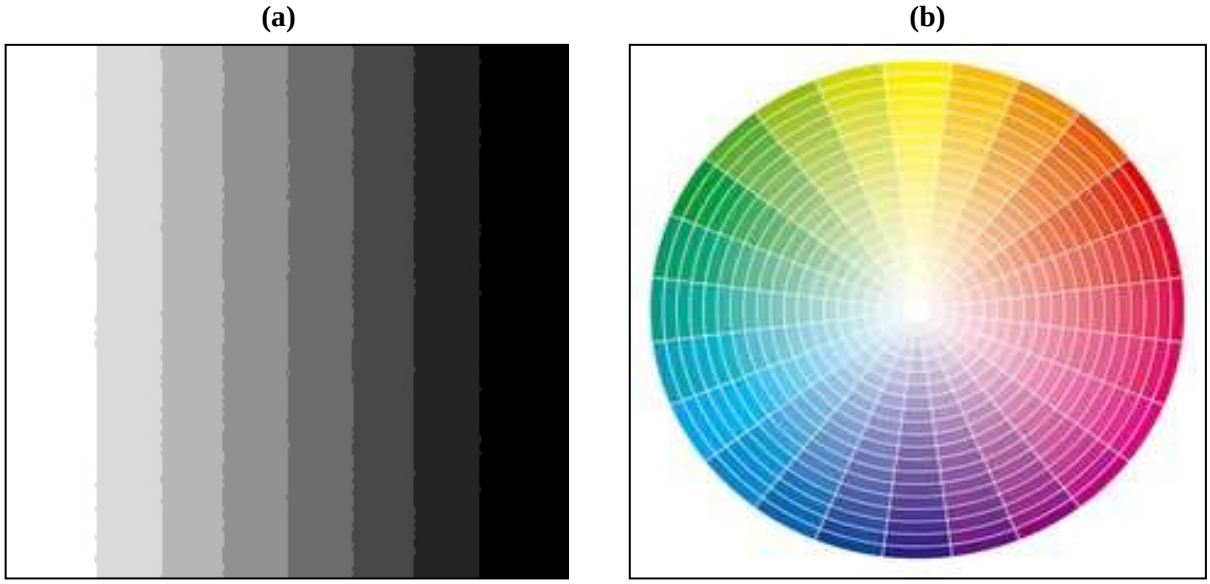


Figure I. 5 Echelle de luminosité d'une image au niveau de gris et (b) couleur.

I.3.6. Profondeur

La profondeur définit la quantité d'informations chromatiques disponibles pour chaque pixel dans une image. La profondeur de couleurs, dont l'unité est le bit par pixel (bpp), décrit le nombre de bits utilisés pour représenter la couleur d'un pixel dans une image.

Le nombre de couleurs disponibles et la précision de la représentation des couleurs dans une image sont proportionnels au nombre de bits d'informations par pixel. Ainsi, dans une image d'une profondeur de 1 bit par pixel, les pixels peuvent prendre deux valeurs possibles : noir et blanc. Une image avec une profondeur de 8 bits par pixel compte 256 valeurs possibles. Une image en mode niveaux de gris avec une profondeur de 8 bits par pixel compte 256 valeurs possibles de gris.

Les images RVB sont constituées de 3 couches de couleur. Une image RVB de 8 bits par pixel a 256 valeurs possibles pour chaque couche, soit plus de 16 millions de valeurs chromatiques possibles.

Les images RVB 8 bits par couche sont parfois appelées images 24 bits (8 bits x 3 couches = 24 bits de données pour chaque pixel).

I.3.7. Poids de l'image

Le poids d'une image est exprimé comme pour tous les fichiers en octets (ou kilo-octets, méga-octets…), il est directement proportionnel à sa définition. En règle générale, plus notre image est grande (définition élevée), plus son poids sera élevé.

Pour connaître le poids (en octets) d'une image, il est nécessaire de compter le nombre de pixels que contient l'image, cela revient à calculer le nombre de cases du tableau, soit la hauteur de celui-ci que multiplie sa largeur. Le poids de l'image est alors égal à son nombre de pixels que multiplie le poids de chacun de ces éléments.

Le poids des images est évidemment en fonction de deux critères:

- Superficie de l'image ➔ Nombre de pixels de l'image.
- Résolution tonale ➔ Nombre de bit d'information couleur par pixels.

On le calcule par :

$$P = N \times R \tag{I.3}$$

ou N : le nombre de pixels dans l'image, R : la résolution totale (bits par pixels).

I.3.8. Contours et textures

Les contours représentent les frontières entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative [14], les textures décrivent la structure de ceux-ci. L'extraction de contour consiste identifier dans l'image les points qui séparent deux textures différentes [15].

Dans la figure suivante un exemple de la détection de contour.

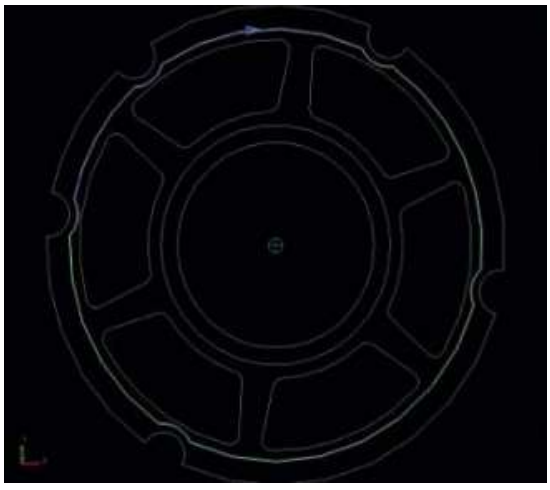


Figure I. 6 Exemple d'une image après la détection de contour.

La détection de contour est très utile en traitement d'images, c'est par exemple une étape indispensable à la reconnaissance de formes. Ces algorithmes sont également utilisés en imagerie médicale, cartographie, détection d'objet…etc.

I.4. Types d'images

Il existe plusieurs types d'images numériques en état de l'art de traitement d'image. Parmi lesquels, on peut citer les types suivants :

I.4.1. Image binaire

Les images binaires sont le type d'image le plus simple et peuvent prendre deux valeurs, généralement le noir et le blanc, ou 0 et 1. Une image binaire est appelée image 1 bit car elle ne nécessite qu'un chiffre binaire pour représenter chaque pixel. Ces types d'images sont fréquemment utilisés dans des applications où les seules informations requises sont la forme générale ou les contours. Les images binaires sont souvent créées à partir des images en niveaux de gris via une opération de seuil, où chaque pixel au-dessus de la valeur de seuil devient blanc ('1'), et ceux situés en dessous sont noir ('0').

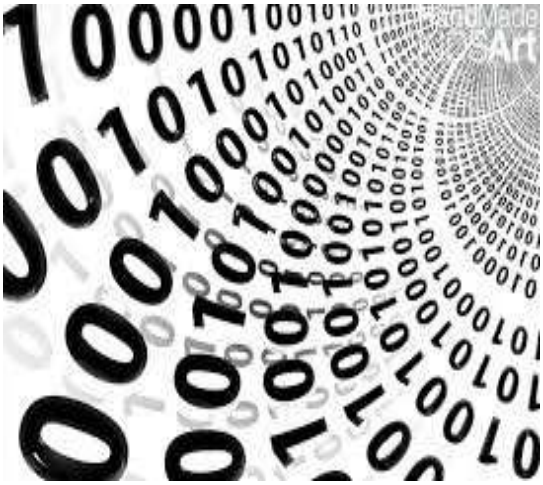


Figure I. 7 Une image binaire.

La couleur de l'objet (généralement le blanc) est appelée couleur de premier plan. Le reste (généralement noir) est appelé couleur de fond. Toutefois, en fonction de l'image à seuiller, cette polarité peut être inversée. Dans ce cas, l'objet est affiché avec 0 et l'arrière-plan avec une valeur non nulle.

I.4.2. Image d'intensité (niveau de gris)

Une image en niveaux de gris est simplement une image dans laquelle les seules couleurs sont des nuances de gris. La raison pour laquelle de telles images se différencient de tout autre type d'image couleur est que moins d'informations doivent être fournies pour chaque pixel. En fait, une couleur «grise» est une couleur dans laquelle les composantes rouge, verte et bleue ont toutes la même intensité dans l'espace RVB. Il est donc nécessaire de spécifier une seule valeur d'intensité pour chaque pixel, par opposition aux trois intensités nécessaires pour spécifier chaque pixel dans une image en couleur.

Souvent, l'intensité des niveaux de gris est stockée sous forme de nombre entier sur 8 bits, ce qui donne 256 niveaux de gris possibles, du noir au blanc. Si les niveaux sont régulièrement

espacés, la différence entre les niveaux successifs est nettement meilleure que le pouvoir de résolution du niveau de l'œil humain.

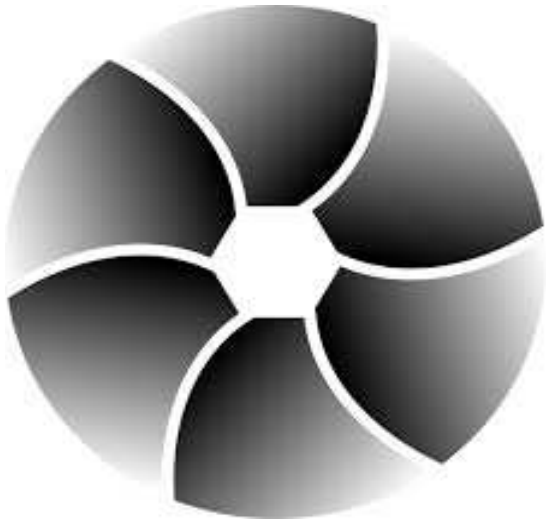


Figure I. 8 Image en niveaux de gris.

Les images en niveaux de gris sont très courantes et tout à fait suffisantes pour de nombreuses tâches telles que la détection des visages. Il n'est donc pas nécessaire d'utiliser des images couleur plus complexes et plus difficiles à traiter. Nous pouvons alors générer des images en niveaux de gris.

I.4.3. Image couleur RVB

L'espace RGB (Red, Green, Blue, pour Rouge Vert Bleu, en français RVB), mis au point en 1931 par la Commission Internationale de l'Eclairage (CIE) consiste à représenter l'espace des couleurs à partir de trois rayonnements monochromatiques de couleurs (rouge, vert et bleu) [5]. Cet espace de couleurs correspond à la façon dont les couleurs sont généralement codées informatiquement, ou plus exactement à la manière dont les tubes cathodiques des écrans d'ordinateurs représentent les couleurs [5].

Une image couleur est une image numérique qui comprend des informations de couleur pour chaque pixel. Une image couleur a trois valeurs (ou canaux) par pixel et mesure l'intensité et la chrominance de la lumière. Les informations réelles stockées dans les données d'image numérique sont les informations de luminosité dans chaque bande spectrale [16, 17].

Les images couleur peuvent être modélisées sous forme de données d'image monochrome à trois bandes, chaque bande de données correspondant à une couleur différente comme le montre la Figure I. 9. Les informations réelles stockées dans les données d'image numérique sont les informations de niveau de gris dans chaque bande spectrale. Les images couleur typiques sont représentées en rouge, vert et bleu (images RVB) en utilisant la norme

monochrome 8 bits comme modèle. L'image couleur correspondante aurait 24 bits / pixel (8 bits pour chacune des trois bandes de couleurs rouge, bleue et verte).

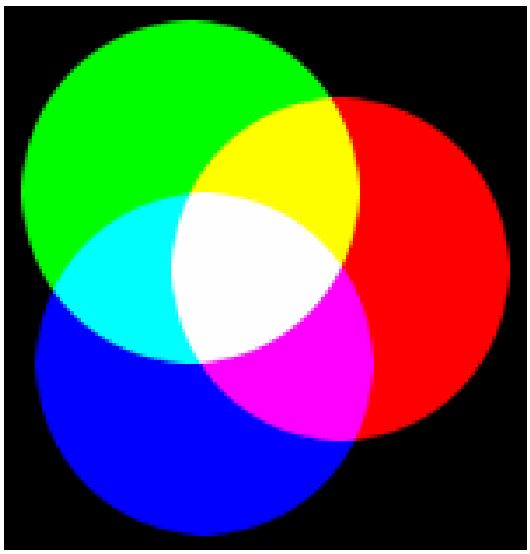


Figure I. 9 Superposition des trois couleurs : rouge, vert et bleu (RGB).

I.5. Formats d'images

Il existe deux types d'images utilisées en informatique et en traitement numérique spécialement :

I.5.1 Images matricielles (ou bitmap)

Une image matricielle (ou bitmap) est une image constituée d'un ensemble de points qui sont les pixels. Chaque point porte des informations de position et de couleur [18]. Lors de l'agrandissement d'une image matricielle, cette dernière devient floue car les pixels ressortent, ce sont les carrés qui apparaissent sur l'écran. Elle est représentée sur une grille dotée de deux axes *X* et *Y*. La définition (ou la précision) d'une image matricielle dépend ainsi du nombre de pixels qui la composent. Plus ils sont nombreux, plus l'image sera nette : c'est la résolution. Les photos numériques et les images scannées sont de ce type.

- **Les formats bitmap sont :**

- ***PPM* (*Portable Pix Map Format*)** : Le format PPM est un format de fichier d'image couleur au plus petit dénominateur commun. Il convient de noter que ce format est extrêmement inefficace. Il est hautement redondant et contient beaucoup d'informations que l'œil humain ne peut même pas discerner.

De plus, le format ne permet que très peu d'informations sur l'image en dehors des couleurs de base, ce qui signifie que nous devrons peut-être associer un fichier de ce format à d'autres informations indépendantes pour en tirer une utilisation décente. Cependant, il est très facile d'écrire et d'analyser des programmes pour traiter ce format .Il convient également de noter

que les fichiers se conforment souvent à ce format sous tous les aspects, à l'exception de la sémantique précise des valeurs de l'échantillon. Ces fichiers sont utiles en raison de la façon dont PPM est utilisé comme format intermédiaire. Ils sont officieusement appelés fichiers PPM, mais pour être absolument précis, vous devez indiquer la variation par rapport à PPM véritable.

- ***TIFF* (*Tagged Image File Format*)** : est un format de fichier informatique permettant de stocker des images graphiques matricielles. Le format TIFF est largement pris en charge par la numérisation, la télécopie, le traitement de texte, la reconnaissance optique de caractères, la manipulation d'images, la publication assistée par ordinateur et les applications de mise en page. Le format de fichier TIFF est un format de fichier flexible et adaptable permettant de traiter des images et des données dans un seul fichier. Il comprend les balises d'en-tête (taille, définition, agencement des données d'image, compression d'image appliquée) définissant la géométrie de l'image. Un fichier TIFF peut également inclure un tracé de détourage basé sur un vecteur (contours, découpages, images). La possibilité de stocker des données d'image dans un format sans perte fait d'un fichier TIFF une archive d'images utile.
- ***GIF* (*Graphics Interchange Format*)** : le Graphics Interchange Format est une animation au format image bitmap qui a été développée par une équipe du fournisseur de services en ligne CompuServe. Le format prend en charge jusqu'à 8 bits par pixel pour chaque image, permettant à une seule image de référencer sa propre palette comprenant jusqu'à 256 couleurs différentes, choisies dans l'espace colorimétrique RVB 24 bits. Il prend également en charge les animations et permet une palette distincte pouvant aller jusqu'à 256 couleurs pour chaque image. Ces limitations de palette rendent le format GIF moins adapté à la reproduction de photographies couleur et d'autres images avec des dégradés de couleurs, mais convient également à des images plus simples, telles que des graphiques ou des logos avec des zones de couleur unies. Les images GIF sont compressées à l'aide de la technique de compression de données sans perte Lempel–Ziv–Welch (LZW) afin de réduire la taille du fichier sans dégrader la qualité visuelle.
- ***JPEG (Joint Photographic Experts Group)*** ➔ Le format JPEG est une méthode couramment utilisée de compression avec perte pour les images numériques, en particulier pour les images produites par la photographie numérique. Le degré de compression peut être ajusté, permettant un compromis sélectionnable entre la taille de stockage et la qualité de l'image.

Il s'agit d'un format d'image standard permettant de contenir des données d'image compressées et avec pertes. Malgré la réduction considérable de la taille du fichier, les images JPEG conservent une qualité d'image raisonnable. Cette fonctionnalité de compression unique permet aux fichiers JPEG d'être largement utilisés sur Internet, sur les ordinateurs et sur les périphériques mobiles. Le partage d'images JPEG est rapide et efficace. En outre, un grand nombre de fichiers image JPEG peuvent être stockés dans un espace de stockage minimal. Les fichiers JPEG peuvent également contenir des données d'image de haute qualité avec une compression sans perte.

- ***BMP* (*Windows Bitmap*) *:*** est un format de fichier image graphique raster utilisé pour stocker des images numériques bitmap, indépendamment du périphérique d'affichage (tel qu'un adaptateur graphique).), en particulier sur les systèmes d'exploitation Microsoft Windows et OS.

  Le format de fichier BMP est capable de stocker des images numériques bidimensionnelles à la fois monochromes et couleurs, dans différentes profondeurs de couleurs et éventuellement avec compression de données, canaux alpha et profils de couleur.
- ***PNG* (*Portable Network Graphics*)** : est un format de fichier d'images raster qui prend en charge la compression de données sans perte. Le format PNG a été mis au point pour remplacer de manière améliorée et non brevetée le format GIF . Le format PNG prend en charge les images basées sur des palettes (avec des palettes de couleurs RVB 24 bits ou RGBA 32 bits), des images en niveaux de gris (avec ou sans canal alpha pour la transparence) et des images RGB / RGBA en couleurs non colorées (avec sans canal alpha).

I.5.2. Les images vectorielles

Les images vectorielles sont composées de formes géométriques qui vont pouvoir être décrites d'un point de vue mathématique. Il s'agit d'un système de proportionnalité et de coordonnées. Grâce à la vectorisation, chaque élément a une place bien définie ce qui empêche la déformation de l'image.

Les professionnels (graphistes, illustrateurs ou concepteurs) réalisent la majorité de leurs visuels en vectoriel afin de pouvoir les modifier à volontés sans les altérer.

L'image vectorielle est surtout utilisée pour la création de cartes, cliparts ou encore pour des animations sur Internet. Elle est par contre peu appropriée pour réaliser des images photoréalistes, qui nécessitent plus de précision dans la nuance des couleurs.

- **Les formats vectoriels sont:**

- ***AI* (*Adobe illustrator*) ➔** AI est une extension de format de fichier graphique vectoriel utilisé dans un dessin Adobe Illustrator qui est un programme de dessin populaire basé sur des graphiques vectoriels. Le format AI est un sous-ensemble de fichiers EPS extrêmement limité et hautement simplifié.
- ***EPS* (*Encapsuled PostScript*) ➔** est un document PostScript conforme à la DSC, assorti de restrictions supplémentaires, destiné à être utilisé en tant que format de fichier graphique. En d'autres termes, les fichiers EPS sont des documents PostScript plus ou moins autonomes, raisonnablement prévisibles, décrivant une image ou un dessin et pouvant être placés dans un autre document PostScript. Simplement, un fichier EPS est un programme PostScript, enregistré en tant que fichier unique comprenant un aperçu basse résolution "encapsulé" à l'intérieur de celui-ci, permettant à certains programmes d'afficher un aperçu sur l'écran.
- ***SVG* (*Scalable Vector Graphics*) ➔** Les fichiers de ce format utilisent un format de texte basé sur XML pour décrire comment l'image doit apparaître.

Comme le texte est utilisé pour décrire le graphique, un fichier SVG peut être redimensionné sans perdre en qualité. En d'autres termes, le format est indépendant de la résolution. C'est pourquoi les graphiques de site Web et d'impression sont souvent conçus au format SVG, de sorte qu'ils peuvent être redimensionnés pour s'adapter à différents modèles à l'avenir.

- ***PDF* (*Portable Document Format*) ➔** est un format de fichier développé pour présenter les documents, y compris le formatage de texte et les images, de manière indépendante des logiciels, du matériel et des systèmes d'exploitation, basé sur le langage PostScript, chaque fichier PDF contient une description complète d'un document plat à disposition fixe, comprenant le texte, les polices, les graphiques vectoriels, les images raster et les autres informations nécessaires à son affichage.

De nos jours, les fichiers PDF peuvent contenir une variété de contenus en plus du texte plat et des graphiques, y compris des éléments de structuration logique, des éléments interactifs tels que des annotations et des champs de formulaire, des calques, du contenu multimédia riche (y compris du contenu vidéo) et des objets tridimensionnels utilisant U3D ou PRC.

- ***PICT* (*Picture Format*) ➔** est un format de fichier graphique. Il permet l'échange de graphiques (bitmap et vectoriels) et d'une prise en charge limitée du texte entre applications Mac. Il s'agissait du format graphique natif de QuickDraw. Dans une application Mac, toute séquence d'opérations de dessin pourrait être simplement enregistrée / encodée au format PICT en ouvrant une "Image", puis en la fermant après avoir exécuté les commandes requises.

I.6. Changement d'espace de couleur

Une gamme de couleurs peut être créée par les couleurs primaires du pigment et ces couleurs définissent ensuite un espace colorimétrique spécifique. Un espace de couleurs est une organisation spécifique des couleurs. En combinaison avec le profilage de périphérique physique, il permet des représentations reproductibles de la couleur, aussi bien analogiques que numériques.

Il y a plusieurs espaces de couleurs qui sont utilisés dans différents domaines. L'espace CIE XYZ sert en photométrie, CIE LUV en visualisation scientifique, CIELAB dans les textiles, RGB pour les moniteurs, CMY pour l'impression, YIQ pour la télévision, HSV, HSI et HLS sont utilisés pour la sélection de couleurs, Munsell en psychologie et Ostwald en peinture [19, 20]. Il existe d'autres espaces encore, mais ceux-ci sont les principaux. Par conséquent, certains offrent une palette de couleurs très réduite (l'espace RAL par exemple utilisé en peinture ne dispose que d'environ 1900 couleurs), alors que les plus complets présentent plusieurs millions de couleurs [21, 22].

I.6.1. L'espace CIE XYZ

Pour certaines valeurs de longueur d'onde la valeur R du RGB est négative et donc certaines couleurs ne sont pas représentées. L'espace RGB ne représente donc pas tout le spectre visible, et c'est pour remédier à cela que l'espace CIE XYZ a été développé. Ce dernier caractérise une couleur en fonction de deux composantes X et Z perpendiculaires représentant la chromacité (tonalité et saturation), et d'une composante Y perpendiculaire aux autres axes qui représente la luminance (clarté), l'ensemble définissant une couleur [23].

Le passage de l'espace RGB vers l'espace XYZ se fait grâce à une matrice 3×3 inversible :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.618 & 0.177 & 0.205 \\ 0.299 & 0.587 & 0.114 \\ 0 & 0.056 & 0.944 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{(I.4)}$$

La transformation inverse de l'espace XYZ vers l'espace RGB est donnée par la relation suivante [24]:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.876 & -0.533 & -0.343 \\ -0.967 & 1.998 & -0.031 \\ 0.057 & -0.118 & 1.061 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \text{(I.5)}$$

I.6.2. L'espace YCbCr

L'espace YCbCr défini par le IRCC (International Radio Consultative Committee), est souvent utilisé dans la compression des images. Cet espace se compose de Y qui est la

composante de luminance (appelée aussi Luma), ainsi que Cb et Cr qui sont les composantes de chrominance (bleue et rouge) voir la Figure I.10 [11].

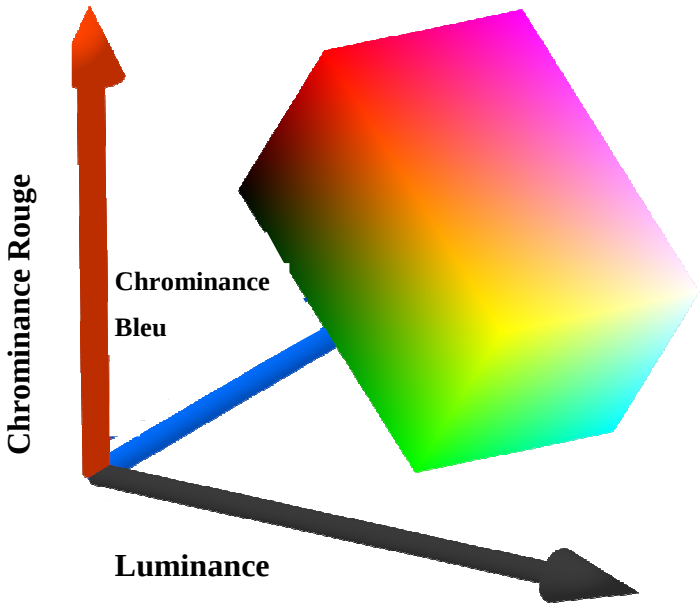


Figure I. 10 La représentation de l'espace de couleur YCbCr dans les cordonnées Luminance, Chrominance bleu et Chrominance rouge.

Puisque la composante Y est plus sensible à l'œil humain, elle doit être plus correcte et Cb et Cr sont moins sensibles à l'œil humain. Par conséquent, il ne faut pas être plus précis. En compression JPEG et MPEG, il utilise ces sensibilités de l'œil humain et élimine les détails inutiles de l'image. Les transformations directe et inverse sont données par les relations suivantes [11]:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.16875 & -0.33126 & -0.50 \\ 0.50 & 0.41869 & 0.0813 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{(I.6)}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.34413 & -0.71414 \\ 1 & 1.772 & 0 \end{bmatrix} \times \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} \quad \text{(I.7)}$$

I.6.3. L'espace HSV

C'est une représentation alternative du modèle de couleur RGB, pour s'aligner plus étroitement sur la façon dont la vision humaine perçoit les attributs de production de couleur. Dans ce modèle, les couleurs de chaque teinte sont disposées en une tranche radiale, autour d'un axe central de couleurs neutres allant du noir en bas au blanc en haut. La représentation HSV modélise le mélange de peintures de différentes couleurs, la dimension de saturation

ressemblant à différentes nuances de peinture de couleurs vives et la dimension de valeur ressemblant au mélange de ces peintures avec des quantités variables de peinture noire ou blanche.

Contrairement aux couleurs RGB, définies en relation avec les couleurs primaires, le HSV est défini de manière similaire à la façon dont les humains perçoivent les couleurs. Elle repose sur trois valeurs: teinte, saturation et valeur (voir la Figure I. 11). Cet espace colorimétrique décrit les couleurs (teinte) en fonction de leur nuance (saturation ou quantité de gris) et de leur valeur de luminosité.

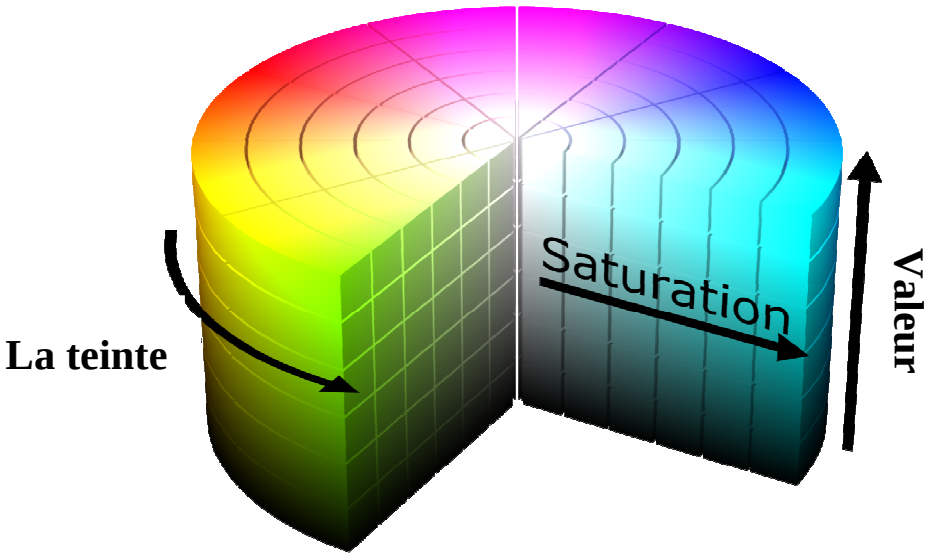


Figure I. 11 La représentation de l'espace de couleur HSV dans l'espace trois dimensions.

- **La teinte (H)** d'une couleur fait référence à la couleur pure à laquelle elle ressemble. Toutes les teintes, les tons et les nuances de rouge ont la même teinte. Les teintes sont décrites par un nombre spécifiant la position de la couleur pure correspondante sur la roue chromatique, sous la forme d'une fraction comprise entre 0 et 1. La valeur 0 correspond au rouge; 1/6 est jaune; 1/3 est vert; et ainsi de suite autour de la roue des couleurs.
- **La saturation (S)** d'une couleur décrit son degré de blancheur. Un rouge pur est complètement saturé, avec une saturation de 1; les teintes de rouge ont des saturations inférieures à 1; et le blanc a une saturation de 0.
- **La valeur (V)** d'une couleur, également appelée luminosité, décrit l'obscurité de la couleur. Une valeur de 0 est noire, avec une légèreté croissante s'éloignant du noir. Le diagramme suivant va nous aider à visualiser la signification des paramètres H, S et V :

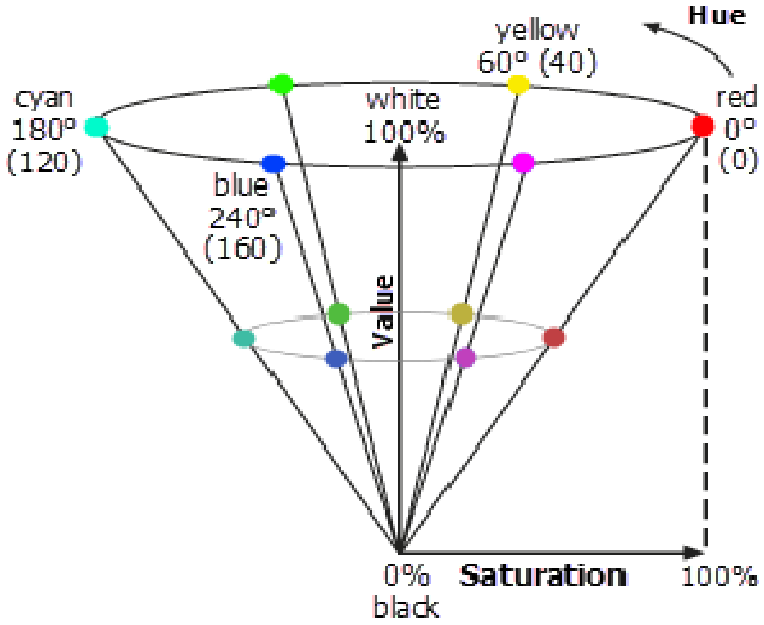


Figure I. 12 Présentation des couleurs dans l'espace HSV.

Le bord extérieur du sommet du cône est la roue chromatique, avec toutes les couleurs pures. Le paramètre H décrit l'angle autour de la roue.

Le S (saturation) est zéro pour toute couleur sur l'axe du cône; le centre du cercle du haut est blanc. Une augmentation de la valeur de S correspond à un mouvement d'éloignement de l'axe.

Le V (valeur ou légèreté) est zéro pour le noir. Une augmentation de la valeur de V correspond à un mouvement d'éloignement du noir vers le haut du cône.

L'espace de transformation est donnée par les relations suivantes [24]:

$$H = \begin{cases} 0, & \text{si} \quad \max = \min \\ (60^0 \times \dfrac{G-B}{\max - \min} + 360^0) \bmod 360^0, & \text{si} \quad \max = R \\ (60^0 \times \dfrac{B-R}{\max - \min} + 120^0), & \text{si} \quad \max = G \\ (60^0 \times \dfrac{B-R}{\max - \min} + 240^0), & \text{si} \quad \max = B \end{cases} \tag{I.8}$$

$$S = \begin{cases} 0, & \text{si} \quad \max = 0 \\ \dfrac{\max - \min}{\max} & \text{ailleurs} \end{cases} \tag{I.9}$$

$$V = \max \tag{I.10}$$

avec $H \in [0 - 360^0], S$ et $V \in [0-1], R, G, B \in [0-1]$.

Passage de HSV à RGB :

$$\begin{cases} H_i = \left\lfloor \frac{H}{60} \right\rfloor \bmod 6 \\ f = \frac{H}{60} - \left\lfloor \frac{H}{60} \right\rfloor \\ p = V \times (1-S), q = V \times (1- f \times S), t = V \times (1-(1- f) \times S) \end{cases} \quad \text{(I.11)}$$

En suite, on calcul le vecteur de couleurs R, G et B par la relation suivante :

$$(R,G,B) = \begin{cases} (V,t,p) & \text{si} \quad H_i = 0 \\ (q,V,p) & \text{si} \quad H_i = 1 \\ (p,V,t) & \text{si} \quad H_i = 2 \\ (p,q,V) & \text{si} \quad H_i = 3 \\ (t,p,V) & \text{si} \quad H_i = 4 \\ (V,p,q) & \text{si} \quad H_i = 5 \end{cases} \quad \text{(I.12)}$$

I.6.4. L'espace HSI

L'espace colorimétrique HSI est un modèle de couleur très important et attrayant pour les applications de traitement d'images car il représente les couleurs de la même manière que l'œil humain perçoit les couleurs.

Le modèle de couleur HSI représente chaque couleur avec trois composantes: teinte (H), saturation (S), intensité (I). La figure ci-dessous montre comment l'espace colorimétrique HSI représente les couleurs :

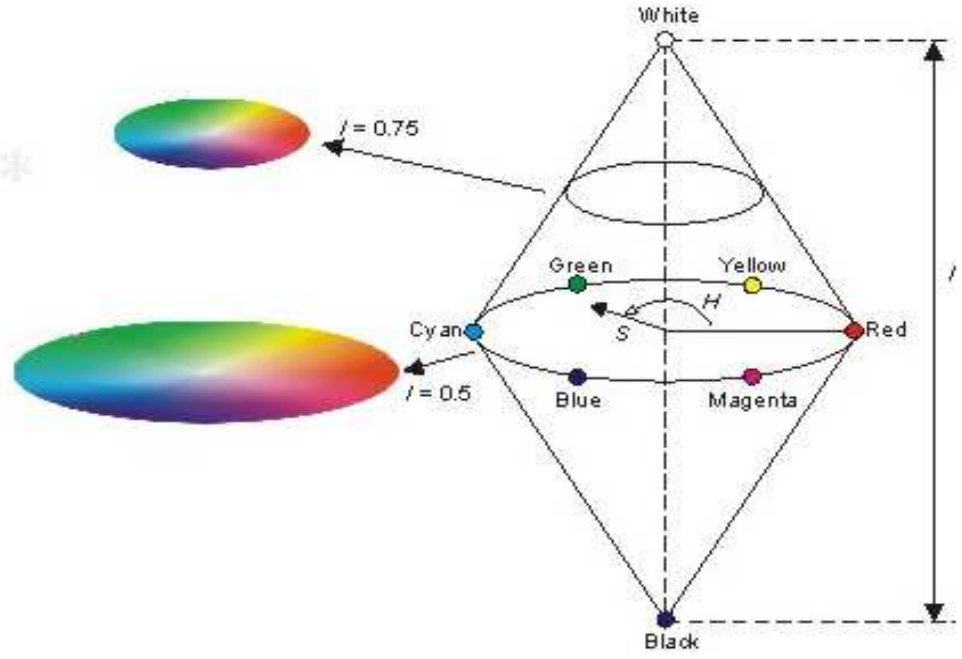


Figure I. 13 Présentation des couleurs dans l'espace HSI.

- La composante Teinte décrit la couleur elle-même sous la forme d'un angle compris entre [0,360] degré. 0 degré signifie rouge, 120 signifie vert 240 signifie bleu. 60 degrés est jaune, 300 degrés est magenta.
- La composante Saturation indique à quel point la couleur est polluée par la couleur blanche. La plage du composant S est [0,1].
- La plage d'intensité est comprise entre [0,1] et 0 signifie noir, 1 signifie blanc.

Le passage de l'espace couleur RGB vers HSI est donné par les équations suivantes [25, 26]:

$$H = \begin{cases} \theta & \text{si} \quad B \le G \\ 360 - \theta & \text{si} \quad B > G \end{cases}, \text{ avec } \theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{\frac{1}{2}}} \right\}$$

$$S = 1 - 3\frac{min(R,G,B)}{(R+G+B)} \tag{I.13}$$

$$I = \frac{1}{3}(R+G+B)$$

I.6.5. L'espace YUV

Le modèle YUV définit un espace colorimétrique en termes de composante luma et deux composantes de chrominance et est similaire à YCbCr. Le modèle de couleur YUV est utilisé dans les normes de vidéo couleur composite PAL, NTSC et SECAM [27].

YUV modélise l'avantage de la perception humaine de la couleur que le modèle RVB standard utilisé dans le matériel informatique. Il code une image couleur ou une vidéo prenant en compte la perception humaine, permettant ainsi une largeur de bande réduite pour les composants de chrominance, permettant ainsi de masquer plus efficacement les erreurs de transmission ou les artefacts de compression par la perception humaine plutôt que d'utiliser une représentation RVB "directe".

Le signal YUV est créé depuis une source RGB (rouge, vert et bleu). Les valeurs de R, G et B sont additionnées selon leur poids relatifs pour obtenir le signal Y. Ce dernier représente la luminance de la source. Le signal U est obtenu en soustrayant le Y du signal bleu d'origine; de façon similaire, le V est obtenu en soustrayant Y du signal rouge.

Les composantes de l'espace YUV généralement est représenté par un cube dont chacun des axes correspond à une composante de couleur (Figure I. 14).

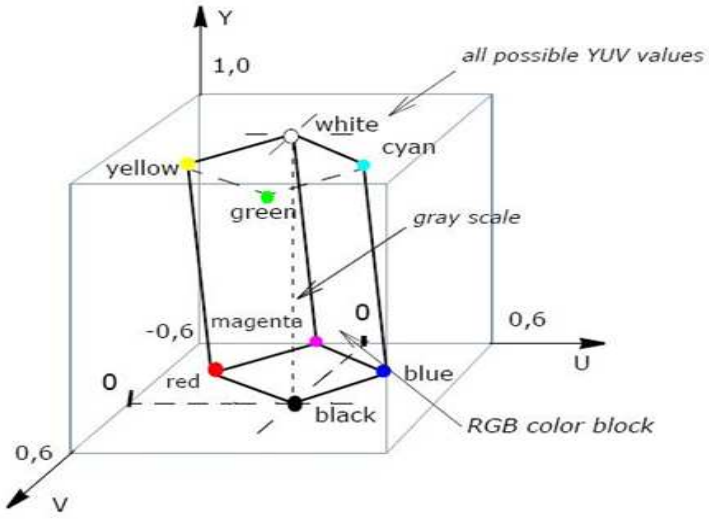


Figure I. 14 Présentation des couleurs dans l'espace YUV.

Le passage de l'espace RGB vers l'espace YUV est défini par l'équation suivante [27]:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.1 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{(I.14)}$$

La transformation inverse de l'espace YUV vers l'espace RGB est donnée par la relation suivante [27]:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.140 \\ 1 & -0.395 & -0.581 \\ 1 & 2.032 & 0 \end{bmatrix} \times \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \quad \text{(I.15)}$$

I.6.6. L'espace YIQ

L'espace YIQ est adopté en premier lieu par le NTSC (National Television System Commitee) pour assurer une diffusion efficace pour les téléviseurs couleurs en garantissant en même temps la compatibilité avec les téléviseurs noir et blanc. La composante *Y* capture l'intensité lumineuse ; *I* et *Q* correspondent à la chrominance [19].

La transformation RGB vers YIQ :

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{(I.16)}$$

La transformation inverse YIQ vers RGB :

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \times \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \quad \text{(I.17)}$$

I.6.7. L'espace YDbDr

YDbDr est l'espace colorimétrique utilisé dans la norme de télévision couleur SECAM. *Y* est la luminance, D_b et D_r sont les différences de couleurs bleue et rouge [19, 24]. Cette espace de couleur est très proche de l'espace YUV. Le passage entre D_bD_r et *UV* se fait grâce aux relations suivantes :

D_b=+3.059*U*, D_r=-2.169*V*.

Le passage entre RGB et YDbDr :

$$\begin{bmatrix} Y \\ D_b \\ D_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.450 & -0.883 & 1.333 \\ -1.333 & 1.116 & 0.217 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{(I.18)}$$

La transformation inverse YDbDr vers RGB :

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.0000923 & -0.5259126 \\ 1 & -0.1291328 & 0.2678993 \\ 1 & 0.6646790 & -0.0000792 \end{bmatrix} \times \begin{bmatrix} Y \\ D_b \\ D_r \end{bmatrix} \quad \text{(I.19)}$$

I.6.8. L'espace YPbPr

L'espace YPbPr est utilisé dans l'électronique visuelle, il est la version analogue de l'espace de couleurs de YCbCr; les deux sont numériquement équivalent, mais YPbPr est conçu pour l'usage dans les systèmes analogues, tandis que YCbCr est prévu pour la vidéo numérique [19]. *Y* est la luminance, P_b est la différence entre le bleu et la luminance, et P_r est la différence entre le rouge et la luminance [24].

-Transformation de l'espace RGB vers un autre espace YPbPr :

$$\begin{cases} Y = \dfrac{R+G+B}{3} \\ P_b = B - Y \\ P_r = R - Y \end{cases} \quad \text{(I.20)}$$

-La transformation inverse de l'espace YPbPr vers l'espace RGB :

$$\begin{cases} R = Y + P_r \\ B = Y + P_b \\ G = Y - P_r - P_b \end{cases} \quad \text{(I.21)}$$

I.6.9. L'espace O1O2O3

L'espace O1O2O3 est un espace de couleurs qui décompose la couleur en trois composantes, O1 c'est l'intensité ou la luminance, tandis que O2 et O3, ensemble, représentent la chrominance à chaque pixel. Les équations suivantes représentent la transformation vers un autre espace O1O2O3 et leur transformation inverse [28] :

$$\begin{cases} O_1 = \left\lfloor \dfrac{R+G+B}{3} + 0.5 \right\rfloor \\ O_2 = \left\lfloor \dfrac{R-B}{2} + 0.5 \right\rfloor \\ O_3 = B - 2G + R \end{cases} \quad \text{(I.22)}$$

$$\begin{cases} R = O_1 + O_2 + O_3 - \left\lfloor \dfrac{O_2}{2} + 0.5 \right\rfloor - \left\lfloor \dfrac{O_3}{3} + 0.5 \right\rfloor \\ G = O_1 - \left\lfloor \dfrac{O_3}{3} + 0.5 \right\rfloor \\ B = O_1 - O_2 + \left\lfloor \dfrac{O_3}{2} + 0.5 \right\rfloor - \left\lfloor \dfrac{O_3}{3} + 0.5 \right\rfloor \end{cases} \quad \text{(I.23)}$$

I.7. Conclusion

Dans ce chapitre essentiellement , nous avons détailler le traitement d'image qui est un domaine très actif où les avancées théoriques se concrétisent sous la forme d'algorithmes rapides de calcul qui ont des applications importantes pour la manipulation des contenus numériques et surtout sur les images et la vidéo .

Nous avons commencé par la citation des caractéristiques de l'image numérique qui englobe les différents paramètres de dimension, résolution, luminance, contraste et autres. Ensuite, nous avons effleuré les types d'image quelque soit une image binaire, niveau de gris ou image couleur avec leurs formats afin de comprendre comment avoir une image dans le format adapté à nos besoins. Finalement, plusieurs représentations d'espace de couleurs sont détailles, reste que l'étude de la mesure des différences de couleur et le choix de l'espace de couleur le plus adéquat sera en fonction du contexte environnant. Si nous étudions maintenant ces caractéristiques couleurs pour le cas de la détection d'objets nous allons bien voir que : Plusieurs espaces couleur sont utilisés dans la détection d'objets, mais aucun d'eux ne peut dominer les autres pour toutes sortes d'images couleur, par conséquence le choix du meilleur espace couleur reste toujours une des difficultés dans la détection d'objets. Dans le chapitre III, nous avons utilisé cinq type d'espace couleur RGB, YCbCr, HSV, HSI, et YIQ.

Chapitre II

Conception des shields Arduino pour la détection d'objets

II.1. Introduction

L'Arduino est une plate-forme informatique physique open source permettant de créer des objets interactifs autonomes ou avec lesquels nous pouvons collaborer des logiciels sur nos ordinateurs. Les cartes Arduino utilisent une variété de microprocesseurs et de contrôleurs. Ses cartes sont équipées d'un ensemble de broches d'entrée/sortie (E/S) numériques et analogiques pouvant être interfacées avec diverses cartes d'extension, cartes de commande et autres circuits. Les cartes disposent d'interfaces de communication série, y compris l'USB sur certains modèles, qui sont également utilisées pour charger des programmes à partir d'ordinateurs personnels. Les microcontrôleurs sont généralement programmés à l'aide d'un dialecte de fonctionnalités issues des langages de programmation C et C ++.

Les cartes Arduino donnent vraiment un potentiel de création quasi infini pourvu de disposer du matériel approprié. Il est possible de fabriquer des robots, de commander des moteurs, de transmettre des données a distance utilisant des shields radiofréquence,...

Dans ce chapitre nous introduisons les parties Matériel et logicielle de la carte Arduino qu'on va utiliser pour programmé notre système de détection d'objet. Ensuite, nous présenterons les différents Shields utilisés dans notre travail.

II.2. Définition du module Arduino

Le module Arduino est un circuit imprimé en matériel libre (open source) dont les plans de la carte elle-même sont publiés en licence libre dont certains composants de la carte : comme le microcontrôleur et les composants complémentaires qui ne sont pas en licence libre. Un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diverses. Arduino est utilisé dans beaucoup d'applications comme l'électrotechnique industrielle et embarquée; le modélisme, la domotique mais aussi dans des domaines différents comme l'art contemporain et le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur, commander des appareils mobiles.

L'environnement de développement de la carte Arduino est basé sur des outils libres comme interface de programmation. L'injection du programme déjà converti par l'environnement sous forme d'un code «HEX» dans la mémoire du microcontrôleur se fait d'une façon très simple par une liaison USB. En outre, des bibliothèques de fonctions "clé en main" sont également fournies pour l'exploitation d'entrées-sorties. Ces cartes sont basée sur un microcontrôleur ATmega et des composants complémentaires. Les cartes Arduino contient une mémoire morte, elles sont dotée par des pins entrées/sorties digitales (ces pins peuvent

être utilisées en tant que sortie PWM), des entrées analogiques et un cristal a 16 MHz, une connexion USB et possède un bouton de remise à zéro et une prise jack d'alimentation.

L'Arduino est une plateforme de contrôle, elle est constituée de deux choses:

Le logiciel (Software) : gratuit et open source, développé en Java, dont la simplicité d'utilisation relève du savoir cliquer sur la souris.

Partie Hardware (Le matériel) : cartes électroniques dont les schémas sont en libre circulation sur internet.

II.2.1. Les différents types de l'ARDUINO

Les différentes versions de l'ARDUINO sont fabriquées par la société italienne Smart Project. Quelques-unes des cartes de marque ARDUINO ont été conçues par la société américaine SparkFun Electronics. Plusieurs versions des cartes ARDUINO ont été produites et vendues dans le monde, on peut citer [29]:

- **SERIAL ARDUINO:** programmé avec une connexion série par un connecteur DB9 et utilisant un microcontrôleur ATmega8.
- **ARDUINO EXTREME:** programmable via une connexion USB et utilisant un ATmega8.
- **ARDUINO MINI:** une version miniature de l'ARDUINO utilisant un ATmega168 de type CMS.
- **ARDUINO NANO:** une version encore plus petite de l'ARDUINO alimenté par USB et utilisant un ATmega168 ou ATmega328 de type CMS.
- **LILYOAP ARDUINO:** une conception de type minimaliste pour permettre une application portable utilisant un ATmega168 de type CMS.
- **ARDUINO NG:** programmable via une connexion USB et utilisant un ATmega8.
- **ARDUINO NG+:** programmable via une connexion USB et utilisant un ATmega168.
- **ARDUINO BLUETOOTH (BT):** programmable via une connexion Bluetooth et utilisant un ATmega328.
- **ARDUINO DIECIMILA:** possède une interface USB et utilise un ATmega168 dans un boitier format DIL28 (16 ko de mémoire flash, 1 ko de mémoire SRAM et 0,5 ko de mémoire EEPROM).
- **ARDUINO DUEMILANOVE (2009):** utilise un Atmega168 et alimenté par le connecteur USB ou une alimentation externe avec commutation automatique. La nouvelle version est équipée d'un ATmega328 (32 ko de flash, 2 ko de SRAM et 1 ko d'EEPROM).

- **ARDUINO MEGA:** est équipé d'un ATmega1280 de type CMS pour avoir des Entrées/Sorties supplémentaires et de la mémoire (128 ko flash, 8 ko SRAM, 4 ko EEPROM).
- **ARDUINO UNO:** utilise un ATmega328 comme les derniers modèles de DUEMILANOVE, mais alors que le DUEMILANOVE utilisait une puce FTDI pour la programmation via un connecteur USB, l'UNO utilise une puce ATmega8U2 programmé comme un convertisseur série.
- **ARDUINO MEGA2560:** est équipé d'un ATmega2560 de type CMS, augmentant la mémoire totale disponible à 256 ko. Il est équipé aussi de la nouvelle puce USB ATmega8U2.
- **ARDUINO ETHERNET:** est une carte ARDUINO UNO intégrant un chip Wiznet W5100 pour rajouter la connectivité Ethernet intégré.
- **ARDUINO LEONARDO:** est une version bas coût de l'ARDUINO UNO à base d'un ATmega32U4.
- **ARDUINO DUE:** est une évolution de l'ARDUINO Mega2560 avec un microcontrôleur 32 bits AtmelSAM3X (ARM 32 bits Cortex-M3).
- **ARDUINO ESPLORA:** est une carte dérivée de l'ARDUINO LEONARDO. Elle a la forme d'une manette de jeu. Contrairement aux autres ARDUINO, c'est une carte «tout-en-un » qui comporte de nombreux capteurs (température, accélération, lumière, microphone, potentiomètre…), ainsi que 4 boutons poussoirs, un potentiomètre et un joystick analogue.

Figure II. 1 Les différentes cartes Arduino selon leurs tailles.

Pour la réalisation de notre travail, on a utilisé deux types des cartes Arduino (Nano et Mega2560).

II.2.1.1. Arduino Nano

La carte Arduino Nano 3.0 est basée sur un ATMega328 cadencé à 16 MHz. Sa mémoire de 32 KB et son grand nombre d'E/S font de ce circuit compatible DIL30 un élément idéal pour les systèmes embarqués ou pour des applications robotiques nécessitant du multitâches. L'Arduino Nano 3.0 peut se programmer avec le logiciel Arduino.

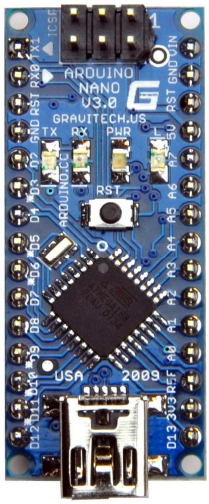


Figure II. 2 Carte Arduino Nano 3.0.

Les spécifications techniques de cette carte Arduino Nano sont les suivantes:

- ✓ Microcontrôleur Atmel ATmega328.
- ✓ Voltage opérationnel (au niveau logique) : 5 V.
- ✓ Pins d'entrées/sorties digitales : 14 (dont 6 proposent une sortie PWM).
- ✓ Pins d'entrée analogique : 8.
- ✓ Courant direct par pin d'entrée/sortie : 40 mA.
- ✓ Mémoire Flash : 32 KO (ATmega328) dont 2 KO sont utilisés par le bootloader.
- ✓ SRAM : 2 KO (ATmega328).
- ✓ EEPROM : 1 KO (ATmega328).
- ✓ Vitesse d'horloge : 16 MHz.

La figure suivante présente l'architecture interne de la carte Arduino Nano version 3.0.

La carte peut être alimentée par la prise mini-USB ou bien par une source d'énergie externe non régulée, à 7-9V (pin 30), ou bien encore par une source d'énergie externe non régulée à 5V (pin 27). La source d'énergie est automatiquement sélectionnée selon le voltage le plus important.

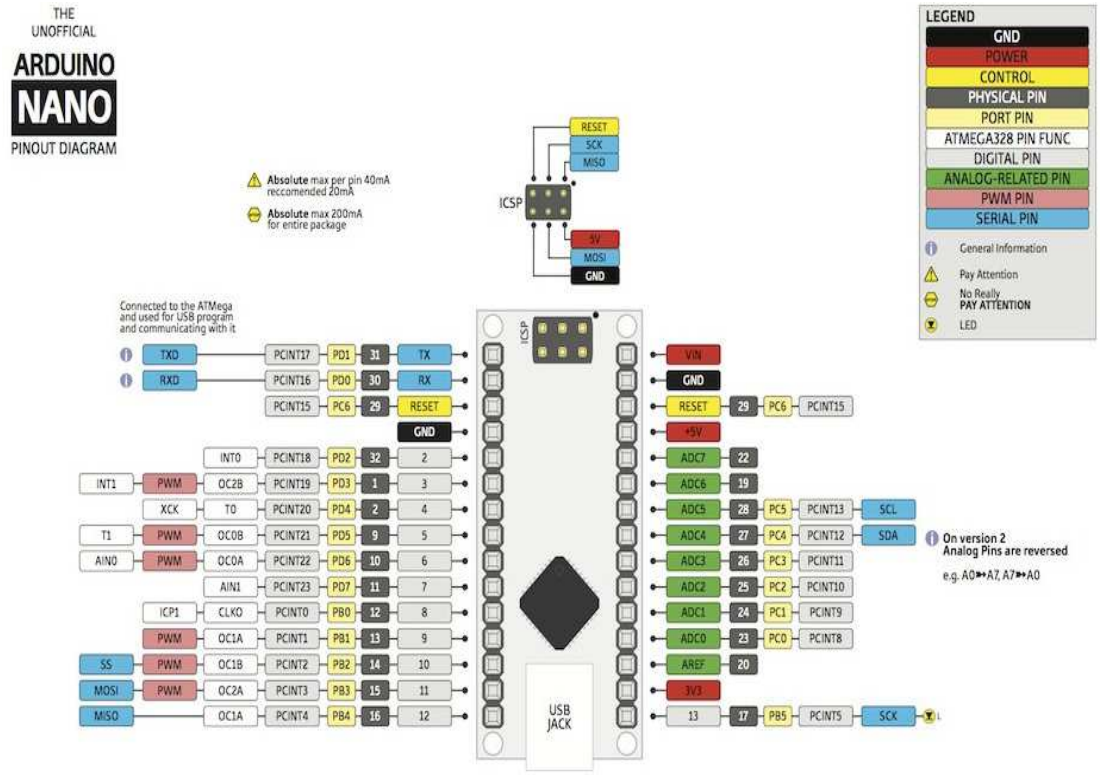


Figure II. 3 Architecture interne de la carte Arduino Nano 3.0.

II.2.1.1.1. ATmega328

Les microcontrôleurs de la famille ATMEGA en technologie CMOS sont des modèles à 8 bits AVR basés sur l'architecture RISC. En exécutant des instructions dans un cycle d'horloge simple, l'ATMEGA réalise des opérations s'approchant de 1 Mb/s par MHZ permettant de réaliser des systèmes à faible consommation électrique et simple au niveau électronique. Pour utiliser ce microcontrôleur sur un montage, nous devons l'alimenter avec une source de tension régulée. Les tensions de 5V ou 3.3V sont populaires, mais l'ATmega328P autorise d'autres tensions d'alimentation (Figures II.4 et II.5). Les microcontrôleurs sont le cerveau des cartes Arduino. Ils vont recevoir le programme que nous allons créer et le stocker dans leur mémoire avant de l'exécuter [30].

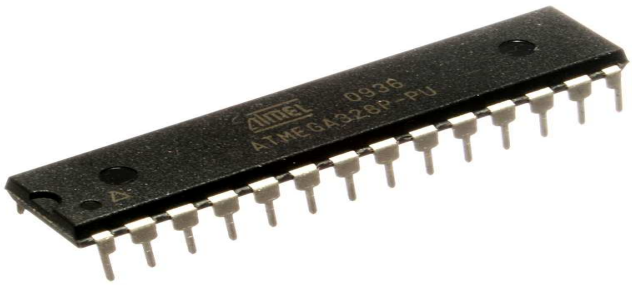


Figure II. 4 Microcontrôleur ATmega328 de la carte Arduino Uno.

Ce type de microcontrôleur est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur. Globalement, l'architecture interne de ce circuit programmable se compose essentiellement sur :

- ✓ **La mémoire Flash:** C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible mémoire programme de 32Ko (dont boot loader de 0.5 ko).
- ✓ **RAM :** c'est la mémoire dite "vive", elle va contenir les variables du programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur. Sa capacité est 2 ko.
- ✓ **EEPROM :** C'est le disque dur du microcontrôleur. On y enregistre des infos qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme [31].

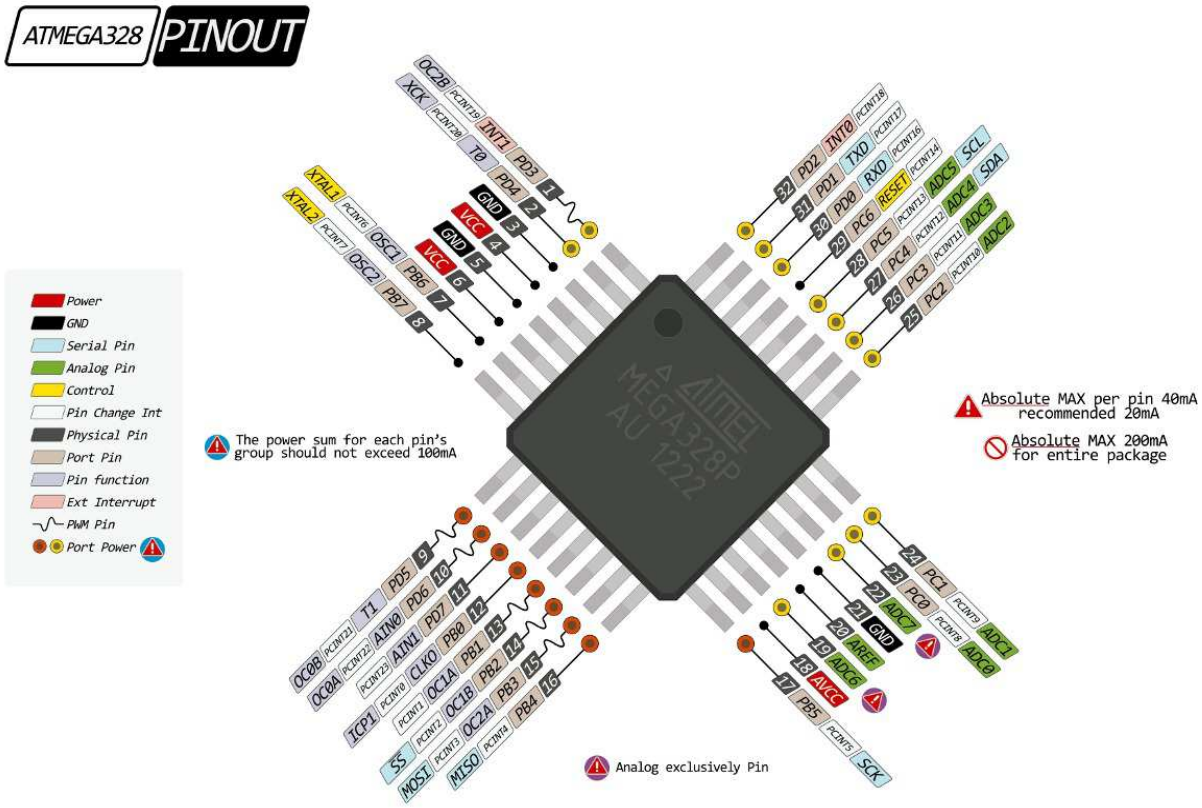


Figure II. 5 Microcontrôleur ATmega328 de la carte Arduino Nano.

II.2.1.2. Arduino Mega2560

Le schéma qui suit montre les déférents composants de la carte Arduino Mega 2560 :

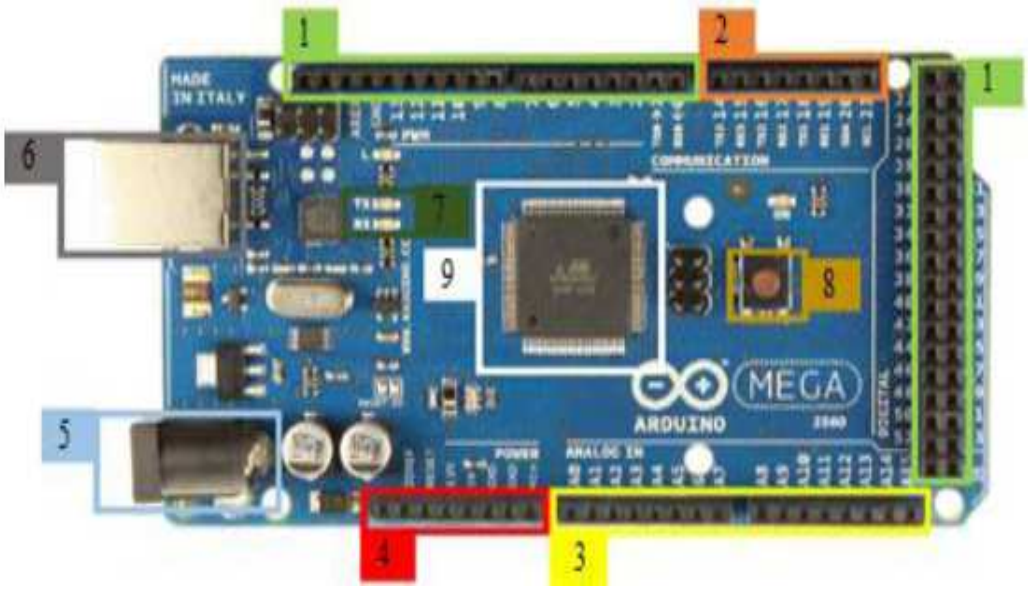


Figure II. 6 Les composants d'Arduino Mega 2560.

1. Entrées/Sorties digitales (54 pin).
2. Pin de communication.
3. Entrées/Sorties Analogique (peuvent aussi servir comme E/S digitales, 16 pin).
4. Broches d'alimentation pour le montage (5V, 3,3V, GND…).
5. Entrée d'alimentation externe pour la carte.
6. Port USB pour la communication entre le PC et la carte.
7. LED indiquant la communication avec l'ordinateur (Tx, Rx)
8. Botton RESET.
9. Microcontrôleur ATmega2560.

La figure II.7 présente l'architecture interne de la carte Arduino Mega. Une telle carte d'acquisition qui se base sur sa construction sur un microcontrôleur doit être dotée d'une interface de programmation comme est le cas de notre carte. L'environnement de programmation open-source pour Arduino peut être téléchargé gratuitement (pour Mac OS X, Windows, et Linux).

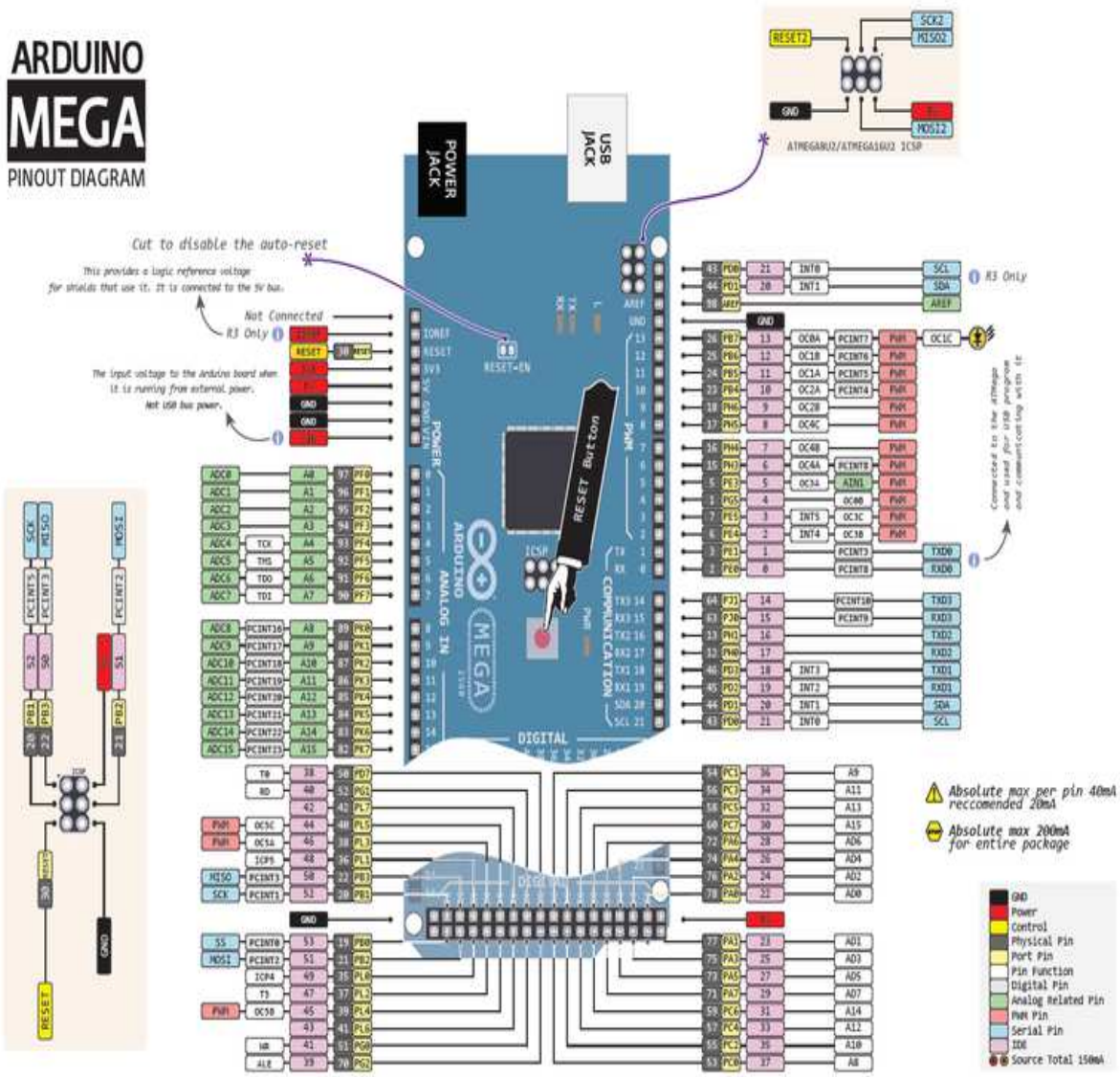


Figure II. 7 Architecture interne de la carte Arduino Mega 2560.

II.2.1.2.1. ATmega2560

Le microcontrôleur RISC Microchip 8 bits à haute performance et faible puissance combine mémoire flash ISP 256KB, SRAM 8KB, EEPROM 4KB, 86 lignes d'E / S à usage général, 32 registres de travail polyvalents, compteur temps réel, six minuteries flexibles / compteurs avec modes de comparaison, PWM, 4 USARTs, interface série à 2 fils orientée octet, convertisseur A / N 10 bits à 16 canaux et une interface JTAG pour le débogage sur puce. L'appareil atteint un débit de 16 MIPS à 16 MHz et fonctionne entre 4,5-5,5 volts.

En exécutant des instructions puissantes dans un cycle d'horloge unique, l'appareil atteint un débit approchant 1 MIPS par MHz, équilibrant la consommation d'énergie et la vitesse de traitement.

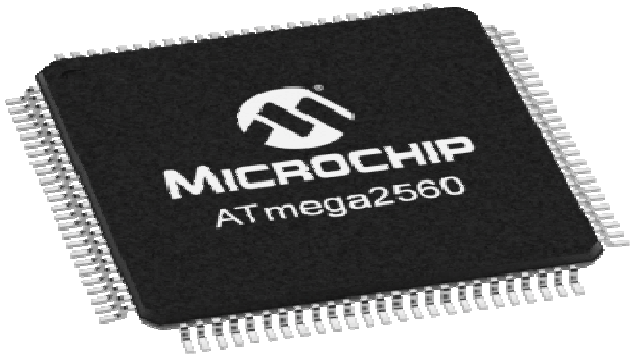


Figure II. 8 ATmega2560.

Les caractéristiques d'un d'ATmega 2560 sont représentées dans le tableau dessous :

Tableau II. 1 Caractéristiques d'un ATmega 2560.

Nom	**Valeur**
Type de mémoire de programme	Flash
Taille de la mémoire du programme	256
Vitesse CPU	16
SRAM Octets	8,192
Données EEPROM / HEF	4096
Périphériques de communication numérique	4-UART, 5-SPI, 1-I2C
Capture / Comparer / Périphériques PWM	4 Input Capture, 4 CCP, 16PWM
Timers	2 x 8-bit, 4 x 16-bit
Nombre de comparateurs	1
Écart de température(C)	-40 to 85
Gamme de tension de fonctionnement(V)	1.8 to 5.5
Nombre de broches	100

II.2.1.3. Les avantage de l'Arduino

- ✓ Elle n'est pas chère.
- ✓ Environnement de programmation simple.
- ✓ Multiplateforme : Windows, Linux.
- ✓ Nombreuses librairies disponibles avec diverses fonctions implémentées.
- ✓ Logiciel et matériel open source et extensible.
- ✓ Nombreux conseils, tutoriaux et exemples en ligne (forums, site personnel etc...)
- ✓ Existence de «shield» : ce sont des cartes supplémentaires qui se connectent sur le module Arduino pour augmenter les possibilités, par exemple : servomoteurs, radiofréquences, afficheur LCD, interface ethernet, GPS, etc..

II.3. Environnement de développement Arduino

Le logiciel de programmation des modules Arduino est une application Java, libre et multiplateforme, servant d'éditeur de code et de compilateur, qui peut transférer le programme dit sketch à travers une liaison série (RS-232, Bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino, et de compiler et uploader les programmes via l'interface en ligne de commande. Le langage de programmation utilisé est le C++, compilé avec avrg++ 3, et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard enrichie le développement de programmes sur les plateformes Arduino, et le rend plus intéressant.

II.3.1. La partie software (Logiciel)

Pour commander un Arduino sur PC on installe le logiciel IDE (Integrated Development Environment) Arduino (par exemple la version 1.6.5 dans notre cas) qui est une application Java et qui a l'interface montré dans la figure II.9. IDE Arduino est une interface souple et simple est exécutable sur n'importe quel système d'exploitation Arduino basé sur la programmation en C ou C++.

Le logiciel Arduino a pour les fonctions principales :

- ✓ De pouvoir écrire et compiler des programmes pour la carte Arduino.
- ✓ De se connecter avec la carte Arduino pour y transférer les programmes.
- ✓ De communiquer avec la carte Arduino.

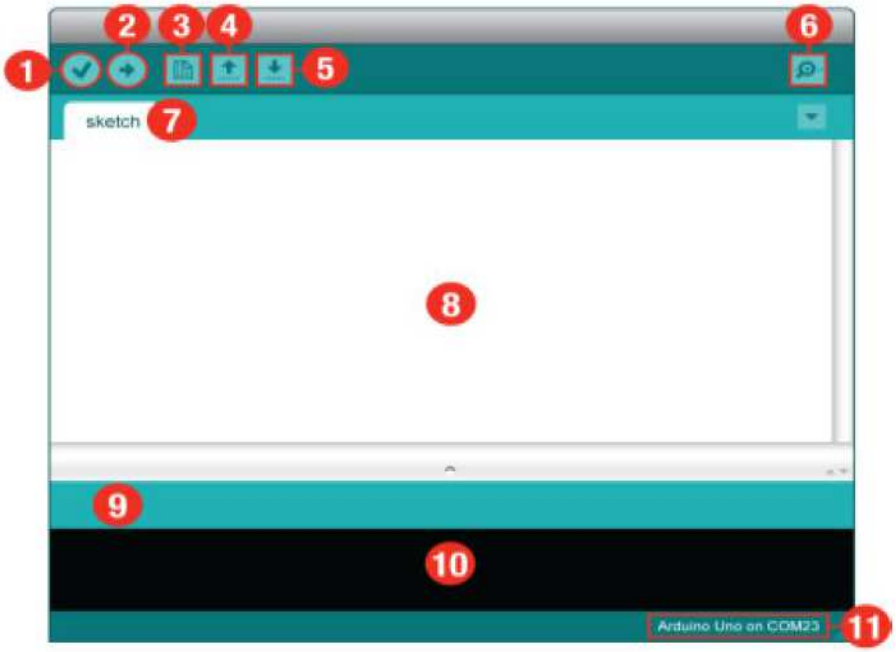


Figure II. 9 Description de l'interface d'Arduino IDE.

1 : Compiler et vérifier le code.

2 : Compiler et téléverser/télécharger le code vers la carte Arduino.

3 : Nouvel onglet de la fenêtre de code.

4 : Ouvrir un projet.

5 : Sauvegarder.

6 : Moniteur série (pour voir les messages qui a été programmes dans le programme).

7 : L'onglet en cours avec le nom de programme.

8 : Editeur de code.

9 : Zone de message.

10 : Le Console des erreurs (zone de notification).

11 : Type de la carte Arduino + port série sur lequel la carte est branchée.

II.3.1.1. Les fenêtres de commande Arduino

Pour programmer l'Arduino on va ouvrir le sketch (Arduino IDE) et après écrire le programme qui fait la commande, ensuite on sélectionne le type d'Arduino et le port USB de notre carte (voir les figures II. 10 et II. 11).

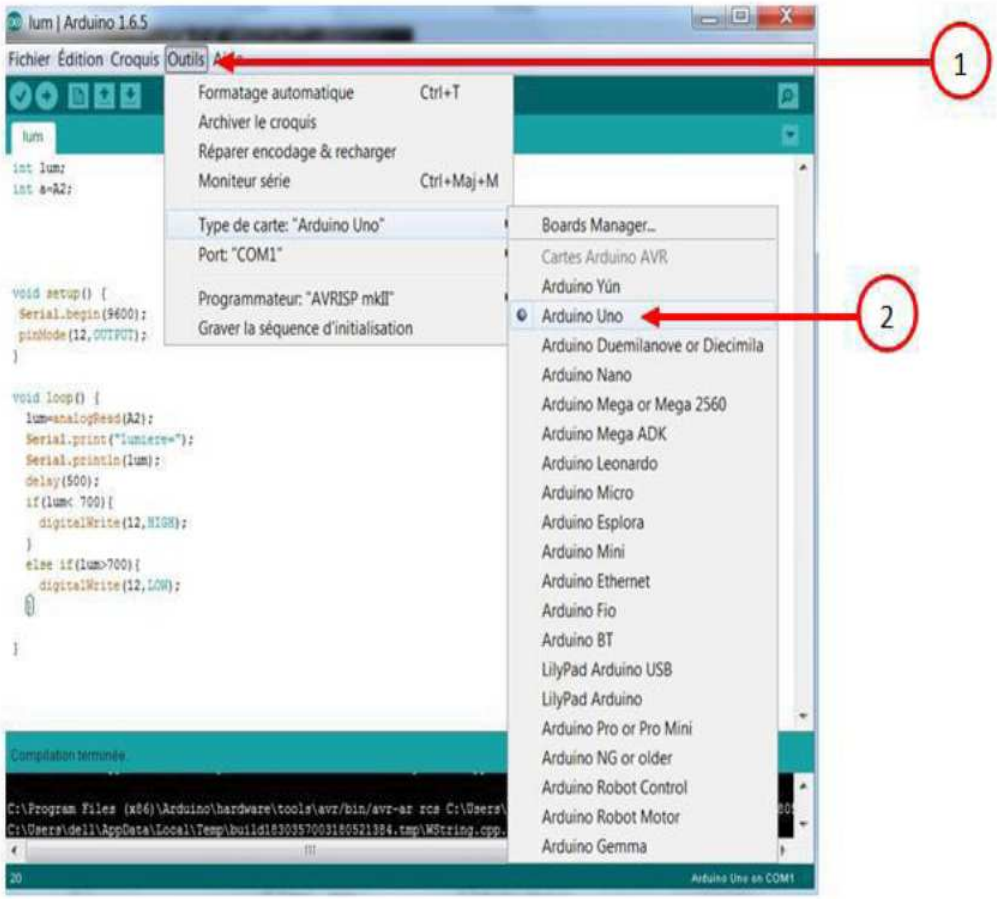


Figure II. 10 Etape 1 : Paramétrage de la carte Arduino.

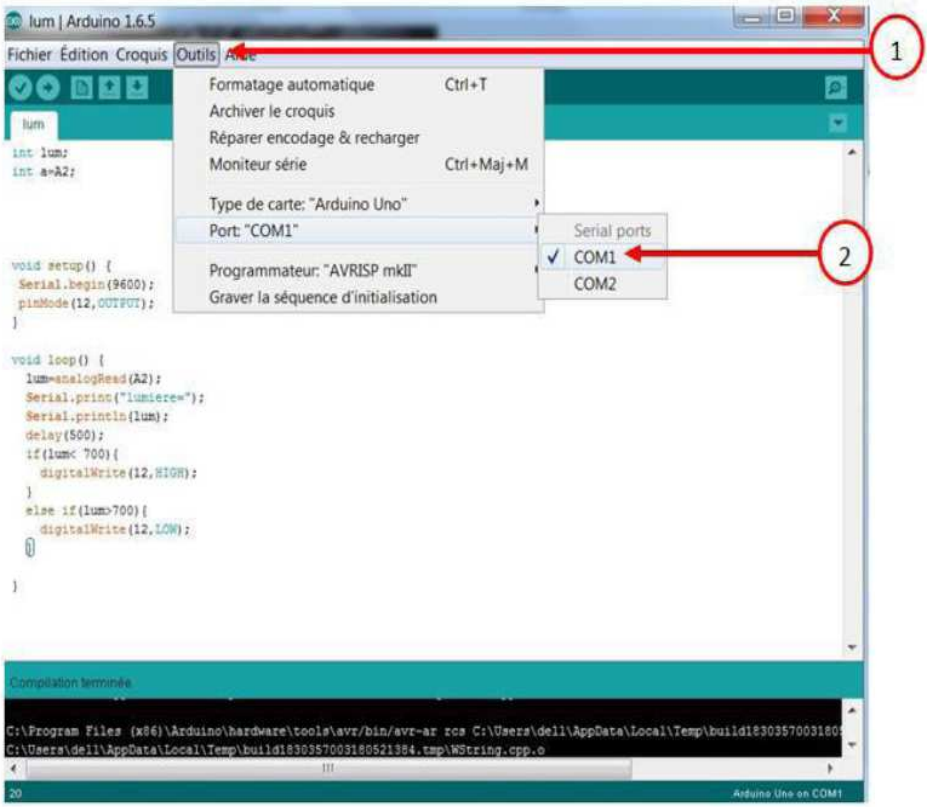


Figure II. 11 Etape 2 : Paramétrage de la carte Arduino.

La troisième étape : vérifier le programme et corrigé les erreurs (voir la figure II.12).

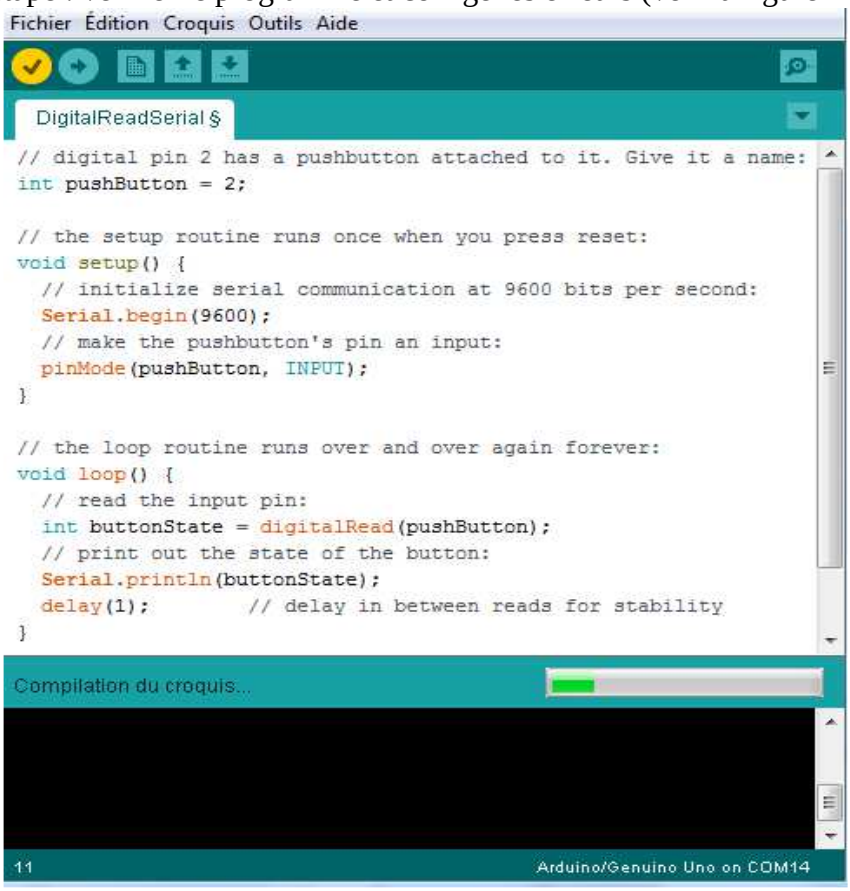


Figure II. 12 Etape 3 : Boutons pour la vérification de programme.

La quatrième étape : chargé (téléverser) le programme dans L'ARDUINO (voir la figure II.13).

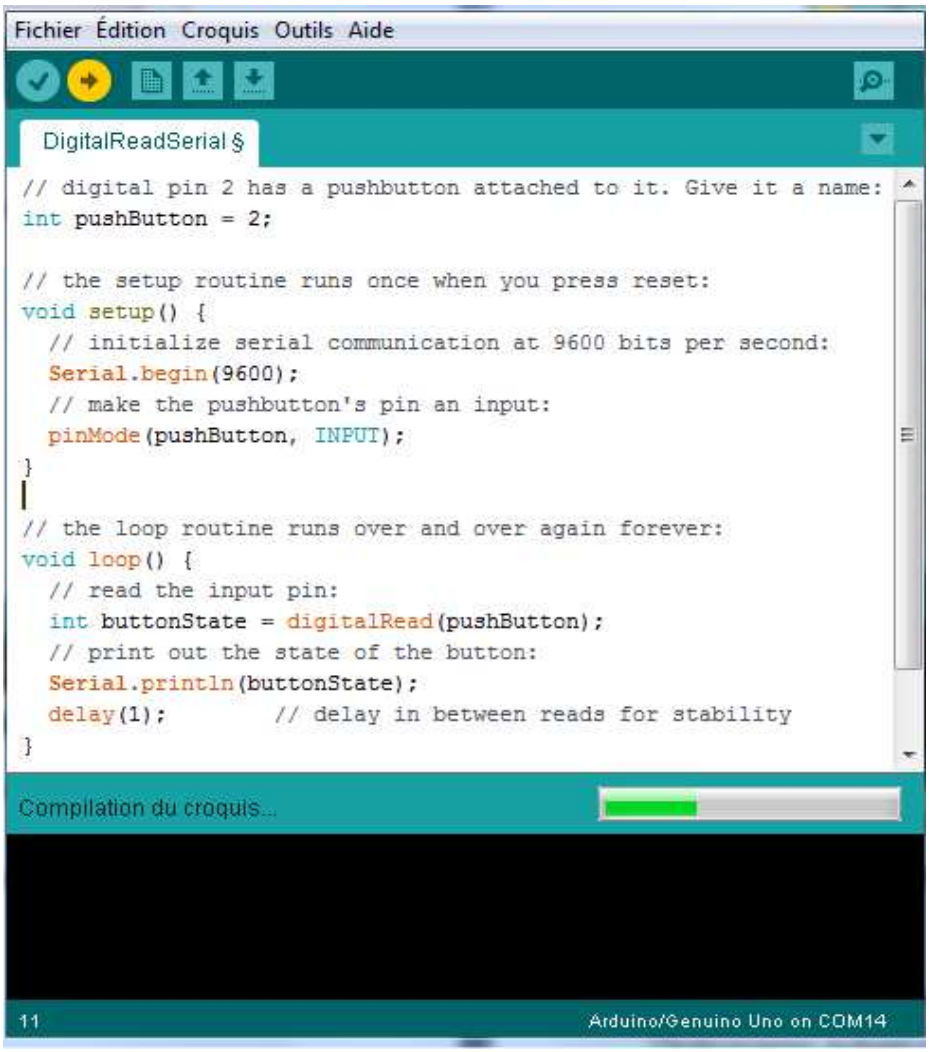


Figure II. 13 Etape 4 : Boutons pour téléverser le programme.

II.3.1.2. Structure d'un programme

Un programme Arduino comporte trois parties :

(1) La partie déclaration des variables (optionnelle).

(2) La partie initialisation et configuration des entrées/sorties : la fonction **setup ()**.

(3) La partie principale qui s'exécute en boucle : la fonction **loop ()**.

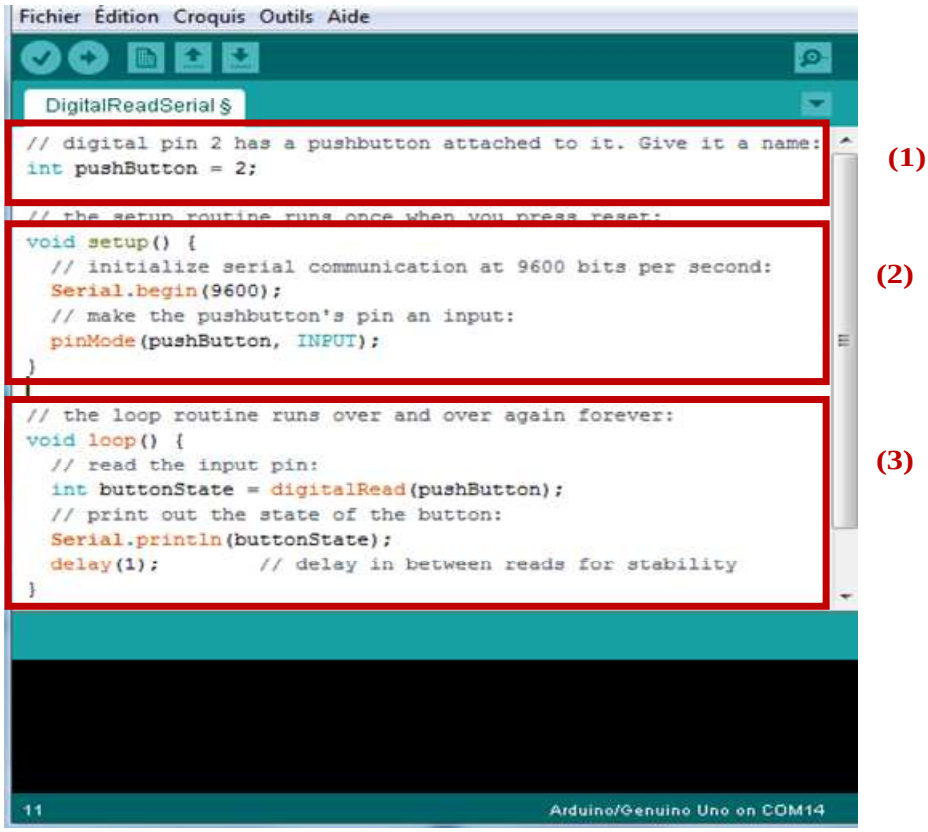


Figure II. 14 Fenêtre principale du programme.

II.4. Les cartes d'extension (Shields)

Les Shields sont des ''extensions'' pour rajouter des fonctions à une carte Arduino, sous forme des cartes conçues pour se connecter directement sur ce dernier. L'avantage du Shield est de minimiser les soudures et les fils, par une simple installation.

Nous avons dans cette partie les shields pour commander le bon fonctionnement de notre système de détection d'objet. Pour notre système, on propose trois types des cartes d'extension :

- **Le Shield de Capteurs :**
  - ✓ Le Shield des capteurs (ultrasons).
  - ✓ Le module NRF24L01 : pour le but de transformer n'importe quelle carte Arduino en un objet communicant par le biais des ondes électromagnétiques.
- **Les cartes d'extension actionneurs :**
  - ✓ Les moteurs à courant continu.
  - ✓ Les Servomoteurs.
- **Le Shield d'affichage :**
  - ✓ L'afficheur LCD.

II.4.1. Le Shield de Capteurs

Le Shield capteur est un terme qui désigne un objet communicant ou plus spécifiquement une carte électronique qui collecte les informations sur l'état physique de l'environnement et les transmets. Il contient quatre unités de base : l'unité de captage, l'unité de traitement, l'unité de transmission, et l'unité de contrôle d'énergie.

Le capteur est le dispositif transformant l'état d'une grandeur physique observée en une grandeur utilisable, telle qu'une tension électrique, une température, une distance. Il est généralement composé de deux sous-unités : le récepteur et le transducteur (convertissant le signal du récepteur en signal électrique). Le capteur fournit des signaux analogiques, basés sur le phénomène observé, au convertisseur Analogique/Numérique. Ce dernier transforme ces signaux en un signal numérique compréhensible par l'unité de traitement. Pour notre cas, la carte Arduino comprend tous ces unités de base sauf l'unité de capture et l'unité de transmission, elles sont représentées par les capteurs (ultrasons) et le module NRF24L01 respectivement, ces deux derniers forment l'extension qu'on appelle le «Shield de Capteurs», et qui s'empile sur les cartes Arduino.

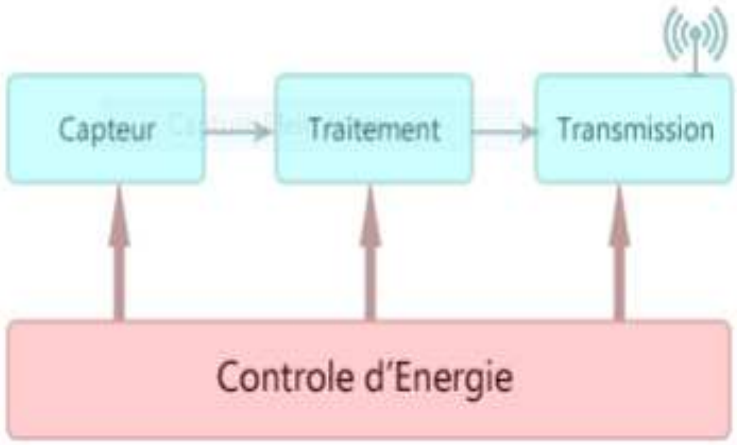


Figure II. 15 Schéma block d'un Shield Capteur.

II.4.1.1. Capteur de distance (ultrason)

L'ultrason est une onde acoustique dont la fréquence est trop élevée pour être audible par l'être humain.

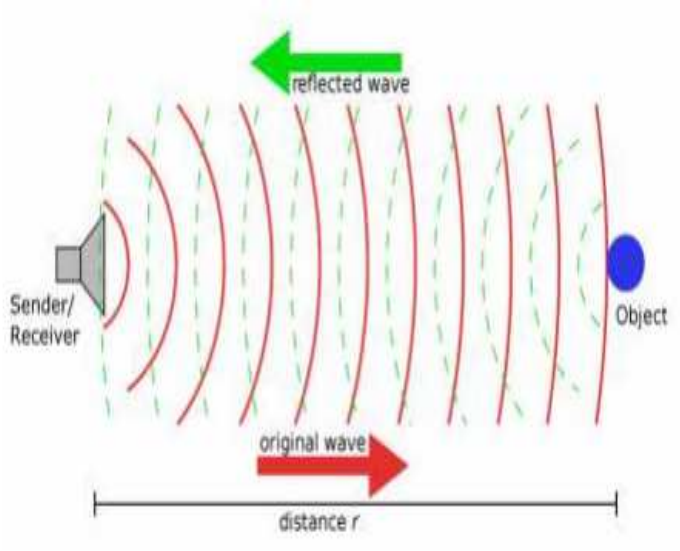


Figure II. 16 Principe du capteur à ultrason.

L'émetteur et le récepteur sont situés dans le même boîtier. L'émetteur envoie un train d'ondes qui va se réfléchir sur l'objet à détecter et ensuite revenir à la source. Le temps mis pour parcourir un aller–retour permet de déterminer la distance de l'objet par rapport à la source. Plus l'objet sera loin plus il faudra long temps pour que le signal revienne.

Figure II. 17 Le capteur de distance à Ultrason HC-SR04.

Dans ce mémoire, on utilise le capteur ultrasons HC-SR04 comme montre la Figure II. 17. Un des deux petits cylindres sur la platine envoie des ultrasons, l'autre cylindre récupère ceux qui reviennent suite à la collision avec un objet. En fonction du temps que l'onde aux ramis pour revenir, nous pourrons savoir la distance qu'elle a parcourue.

II.4.1.2. La Transmission-Réception Radio Fréquence (RF24)

L'élément commun à tous les Shields de notre système est le module radio NRF24L01. L'implémentation de ce module dans un circuit avec un microcontrôleur qui supporte la communication SPI est simple. Il faut juste identifier les pins du bus SPI de la carte Arduino utilisé, et les attacher aux pins SPI respectives du module RF. On peut consulter le tableau suivant pour les différentes cartes Arduino :

Tableau II. 2 Câblage des 8 broches du module nRF avec les différents types d'Arduino.

Module Pin	Description	Uno	Nano	Mega
1	GND	GND	GND	GND
2	VCC	3.3V	3.3V	3.3V
3	CE	D8	D8	D8
4	CSN	D7	D7	D7
5	SCK	D13	D13	D52
6	MOSI	D11	D11	D51
7	MISO	D12	D12	D50
8	IRQ	--	--	--

On remarque que le NRF24L01 marche avec une alimentation VCC de 3.3V, donc il faut le brancher avec le pin 3.3V de la carte Arduino. Le reste des pins tolèrent et marchent avec des tensions de 5V. Avec cette considération bien entendue, on n'aura pas une possibilité de griller le module RF. Mais dans nos essais pour tester une ligne de communication entre deux Arduino équipés des modules radios de même type avec une simple installation, c'est-à-dire que on a utilisé juste les modules directement avec les cartes Arduino sans ajouter d'autre composants électroniques, le résultat est aucun paquet RF n'a été transmis. Pour cela on a ajouté un condensateur de découplage (1uF100uF) entre les bornes d'alimentation du module RF, ce condensateur sert à filtrer les harmoniques et le bruit du pin à 3.3V, et fournis une puissance électrique quand il y'a un pique de courant exercer par le module RF. Les deux figures suivantes représentent deux types de nRF sans antenne et avec antenne.

Figure II. 18 Module nRF24L01.

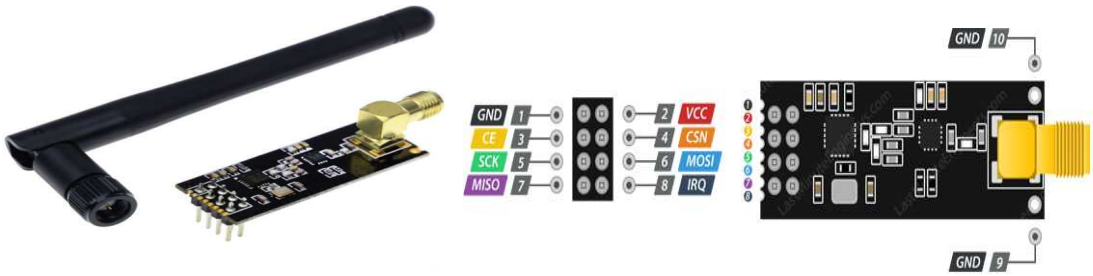


Figure II. 19 Module nRF24L01-PA-LNA.

II.4.2. Les cartes d'extension actionneurs

Nous avons utilisé deux actionneurs dans ce système, le moteur à courant continue et le servomoteur.

II.4.2.1. Moteurs à courant continu

Le moteur à courant continu, comme le dit son nom marche en courant continu et il convertit l'énergie électrique en énergie mécanique. Pour notre système nous avons utilisé ce moteur (Figure II. 20) pour commander le mouvement de robot mobile. Pour commander le mouvement d'un robot utilisant les cartes microcontrôleurs, nous avons besoin d'un pont en H.

Figure II. 20 Moteur à courant continu.

II.4.2.1.1. H-bridge

C'est un circuit électrique qui se compose par quatre transistors (2 PNP et 2 NPN) monté de telle façon que le courant puisse passer soit dans un sens, soit dans l'autre au travers d'un moteur à courant continu. La figure ci-dessous présente les deux sens de rotation.

L'utilisation de transistors pour modifier le sens du courant peut aussi dégrader le montage, en effet :

- Si *t*1 et *t*2 sont ouverts: une courante position traverse le moteur.
- Si *t*3 et *t*4 sont ouverts: un courant négatif traverse le moteur.
- Si plus de 2 transistors sont ouverts: le courant ne circule pas.
- Si *t*1 et *t*2 ou *t*3 et *t*4 sont fermés: le montage court-circuite.

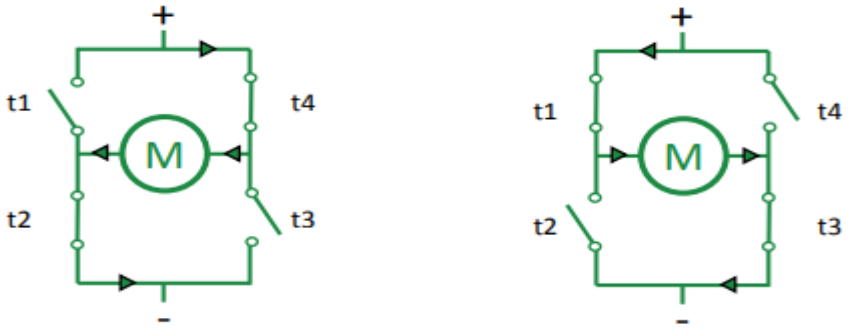


Figure II. 21 Schéma de principe de fonctionnement du pont en H.

Il est possible d'activer la rotation d'un moteur à l'aide d'un relais ou d'un transistor. L'inconvénient de l'option transistor (ou relais) est qu'il n'est possible de facilement contrôler le sens de rotation du moteur. La solution réside dans l'utilisation d'un pont H. composant constitué de plusieurs transistors mais vendu pré assemblé sous forme de circuit intégré. Pour cela on a utilisé dans notre system le circuit intégré **L298N**.

II.4.2.1.1.1. Pont-H L298N

Ce breakout board est un Double Pont-H destiné au contrôle de moteur continu (H-Bridge Motor Driver). Il est basé sur le composant L298N qui est un double Pont-H conçu spécifiquement pour ce cas d'utilisation.

C'est un module extrêmement utile pour le contrôler de robots et ensembles mécanisés. Il peut contrôler deux moteurs à courant continu ou un moteur pas-à-pas 4 fils 2 phases. Il est conçu pour supporter des tensions plus élevées, des courants importants tout en proposant une commande logique TTL (basse tension, courant faibles, idéal donc pour un microcontrôleur). Il peut piloter des charges inductives comme des relais, solénoïdes, moteurs continus et moteurs pas-à-pas. Les deux types de moteurs peuvent être contrôlés aussi bien en vitesse (PWM) qu'en direction. Toutes les sorties en puissance sont déjà protégées par des diodes anti-retour [32]. La figure II. 22 représente les détails techniques du module L298N.

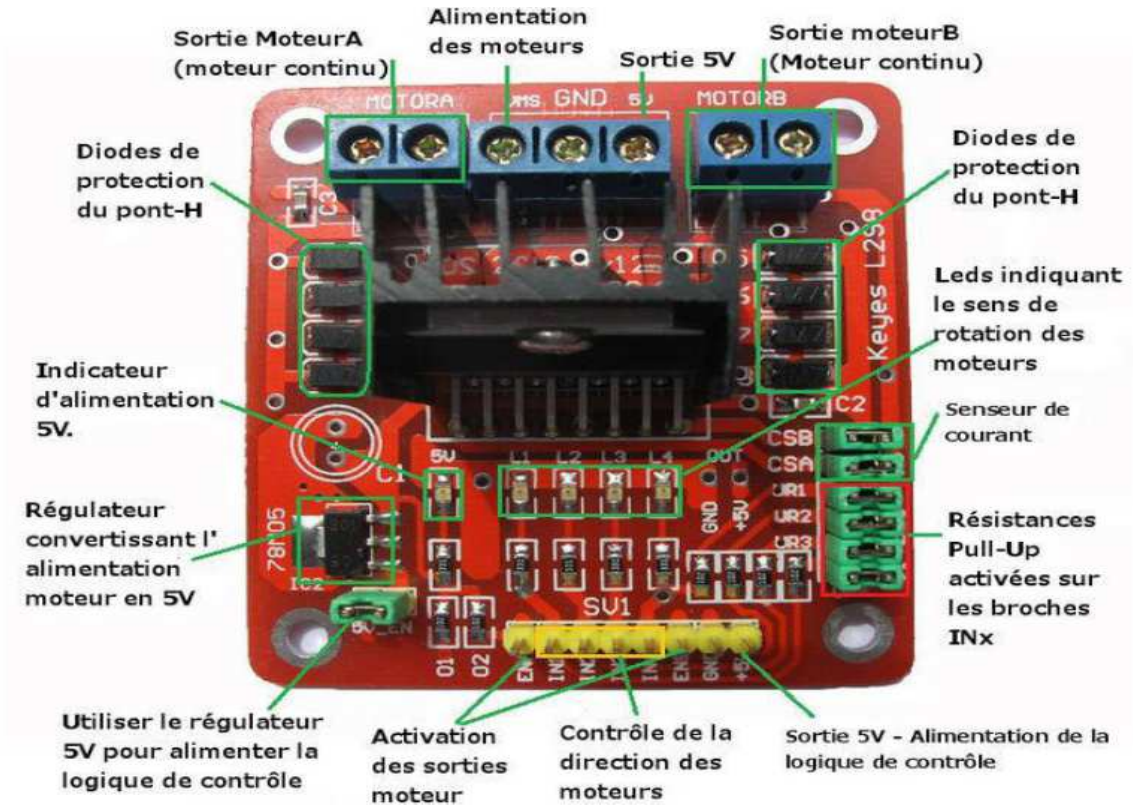


Figure II. 22 Les détails techniques du L298N.

II.4.2.2. Servomoteur

Les servomoteurs sont de petits moteurs avec un réducteur intégré permettant des déplacements angulaires sur un axe de 180°. Le pilotage du servomoteur se fait avec l'Arduino. Il nous permet de nous positionner avec précision sur un angle [33]. La plage de l'angle est de 0 à 180 degrés. Il existe différent types de servomoteur, mais ils fonctionnent de la même manière voici une figure qui illustre quelques types:

Figure II. 23 Quelques types des servomoteurs.

Le servomoteur est composé de plusieurs éléments :

Eléments visibles :

- **Les fils :**
  - Rouge pour l'alimentation positive (4.5v à 6v).
  - Noir ou marron pour la masse (0v).
  - Orange ou jaune ou blanc : l'entrée du signal de commande.
- L'axe de rotation sur lequel est monté un accessoire en plastique ou en métal.
- Le boitier qui le protège

Eléments non visibles :

- Un moteur a courant continu.
- Des engrenages pour fermer un réducteur (en plastique ou en métal).
- Un capteur de position de l'angle d'orientation de l'axe.
- Une carte électronique pour le contrôle de la position de l'axe et le pilotage du moteur a courant continu.

Dans ce travail, nous avons utilisé le servomoteur représenté sur la figure suivante :

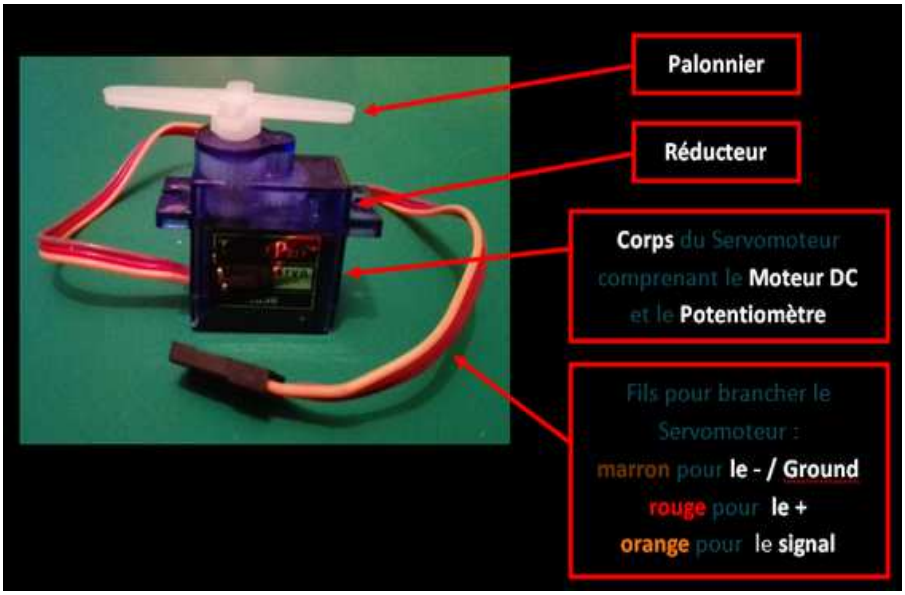


Figure II. 24 Servomoteur.

II.4.2.2.1 Fonctionnement d'un servomoteur

Le servomoteur comprend essentiellement un moteur à courant continu, un système à engrenages, un capteur de position et un circuit de commande. Les moteurs à courant continu sont alimentés par une batterie et fonctionnent à haute vitesse et à couple réduit. L'ensemble de l'engrenage connecté aux moteurs à courant continu abaisse cette vitesse à une vitesse suffisante et à un couple plus élevé. Le capteur de position détecte la position de l'engrenage à partir de sa position définie et transmet les informations au circuit de commande. En conséquence, le circuit de commande décode les signaux du capteur de position et compare la position réelle des moteurs à la position souhaitée et commande en conséquence le sens de rotation du moteur à courant continu pour obtenir la position requise. Le servomoteur nécessite généralement une alimentation en courant continu de 4,8V à 6 V.

Pour calculer le couple, on utilisera la formule suivante :

$$P_{\max} = \frac{C}{d} \qquad \text{(II.1)}$$

Donc :

$$d = \frac{C}{P_{\max}} \qquad \text{(II.2)}$$

Avec :

- $\boldsymbol{P_{max}}$: le poids maximal de l'objet à déplacer en *kg*.
- $\boldsymbol{C}$: le couple du servomoteur exprimé en *kg.cm*.
- $\boldsymbol{d}$: la distance à laquelle est placé l'objet de l'axe, exprimé en *cm*.

II.4.3. Module Afficheur LCD

Les afficheurs à cristaux liquide, autrement appelés LCD (liquide Crystal Display), sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement. Ils consomment relativement peu d'énergie (de 1 à 5*mA*), sont relativement bons marchés et s'utilisent avec beaucoup de facilité. Plusieurs afficheur sont disponibles sur le marché et différent les un des autre, non seulement par leur dimension mais, aussi par leur caractéristique technique, et leur tension de service. Pour notre projet nous avons choisi LCD 16×2 parce qu'il répond à l'affichage des données relevées de notre commande (distance).
La Figure II.25 présente l'afficheur LCD 16×2 et leurs différents pins.

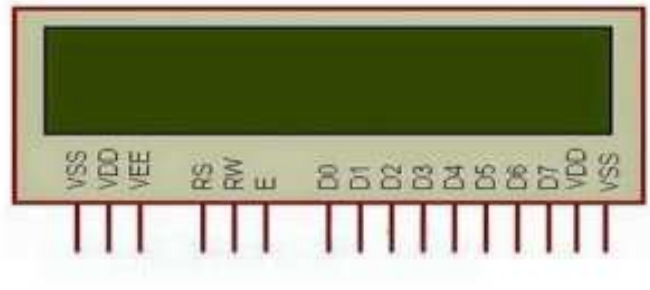


Figure II. 25 Afficheur LCD 16×2.

II.5. Conclusion

Dans ce chapitre, nous avons expliqué les deux parties essentielles de l'Arduino; plus précisément (la partie matérielle et la partie de programmation). Ainsi quelques descriptions théoriques sur le moteur à courant continu, H-bridge, le servomoteur, l'afficheur LCD, et l'ultrason. Nous avons également expliqué les différents carte d'extension utilisé dans notre travail, ainsi les raisons pour lesquelles on l'a choisie les Shields capteurs (capteur de distance, la transmission radiofréquence), les cartes d'extension actionneurs (moteur à courant continu, servomoteur) et le module d'affichage.
Nous sommes intéressés par la méthode de détection des objets selon la couleur avec l'utilisation de la carte Raspberry *Pi* 3 et une camera qui sera le but de notre travail dans le chapitre qui suit.

Chapitre III

Traitement d'images sur Raspberry Pi

III.1. Introduction

La détection et le suivi d'objet est une tâche très importante dans le domaine de la vision par ordinateur. Il consiste à estimer le déplacement d'un objet dans la cène par le biais de primitives visuelles qui le représentent. Dans ce chapitre, nous intéressons qu'aux détections d'objets couleurs. Les images considérées contiennent un seul objet placé sur un fond uniforme et éclairé avec un illuminant de caractéristiques inconnues qui diffère d'une image à l'autre. Elles sont acquises dans les conditions suivantes: les objets peuvent subir une rotation et/ou une translation dans un plan perpendiculaire à l'axe optique de la caméra, les paramètres de réglage de la caméra ne sont pas modifiés entre les acquisitions, les modifications d'éclairage résultent de changements d'illuminant, c'est à dire d'une modification de l'intensité de l'illuminant utilisé.

Le changement de l'espace couleur est une étape importante dans l'étude et la mise en ouvre de traitement et la détection d'objet. Dans la littérature scientifique plusieurs types de changement d'espace couleur ont été proposés, le changement YCbCr, HSI, YIQ, HSV, CMY, ... etc.

Le chapitre présent est divisé en deux parties, dans la première partie nous présentons brièvement la carte Raspberryi *Pi* 3, dans la deuxième partie, nous présentons deux méthode de détection d'objets couleur : seuillage fixe et adaptative. Par la suite on présentera une application de traitement d'image sur le Raspberry *Pi* 3 afin de trouvé le meilleur espace couleur adapté à la détection d'objet couleur.

III.2. Le Raspberry *Pi*

Raspberry est une carte mère d'un mini-ordinateur qui peut être branchée à n'importe quel périphérique (souris, clavier...). Cette carte est fabriquée pour aider à étudier les ordinateurs et pour représenter un moyen d'apprentissage de la programmation informatique en plusieurs langages (python, scratch...). Elle est aussi capable de lire les vidéos à haute définition et même à installer des jeux vidéo. L'intérêt d'utiliser le Raspberry *Pi* est sa capacité d'interaction avec le monde extérieur et d'exécuter plusieurs variantes du système d'exploitation libre (GNU/Linux, Raspbian Debian ...) et des autres logiciels compatibles

III.2.1. Raspberry *Pi* 3

La carte Raspberry Pi 3 est une ordinateur mono-carte pouvant se connecter à un téléviseur, à un clavier et disposant d'une connectivité Wi-Fi et Bluetooth, possède 4 ports USB, un port micro-SD, un connecteur d'E/S 40 broches et un port HDMI. Cette version est physiquement identique à la version Pi 2 la rendant compatible avec les boîtiers et autres accessoires de la

version *Pi* 2. La version 3 est basée sur un processeur ARM Cortex-A53 64 bits quatre cœurs à 1,2 GHz (environ 10x plus rapide que le *Pi* 1 et 50% plus performante que le modèle *Pi* 2) et possède 1 GB de mémoire RAM.

Le Raspberry *Pi* peut effectuer des tâches d'un PC de bureau (feuilles de calcul, traitement de texte, jeux). Il peut également diffuser des vidéos en haute définition grâce à son circuit Broadcom Video Core IV (permet le décodage des flux Blu-ray full HD).

Caractéristiques:

- CPU: Processeur SoC de noyau de Cortex-A53 de quadruple BCM2837 64bit de Broadcom fonctionnant à 1.2GHz
- Mémoire: 1 Go de RAM LPDDR2
- 4 ports USB2.0 avec une sortie jusqu'à 1.2A
- En-tête GPIO étendu à 40 broches
- Sortie vidéo / audio via un connecteur 3,5 mm 4 broches, HDMI, caméra CSI ou LCD brut (DSI)
- Stockage: Support pour cartes micro-SD
- Port Ethernet 10/100 (RJ45)
- WiFi: 2,4 GHz, 802.11n (Broadcom BCM43438)
- Bluetooth 4.1 (Broadcom BCM43438)
- Périphériques de bas niveau:

  27 x GPIO

  UART

  Bus I2C

  Bus SPI avec deux choix de puces

  + 3,3 V

  + 5V

  GND
- Alimentation : 5 Vcc/maxi 2.5 A * via prise micro-USB (* intensité maxi si toutes les fonctions sont utilisées)
- Bus: SPI, I2C, série.

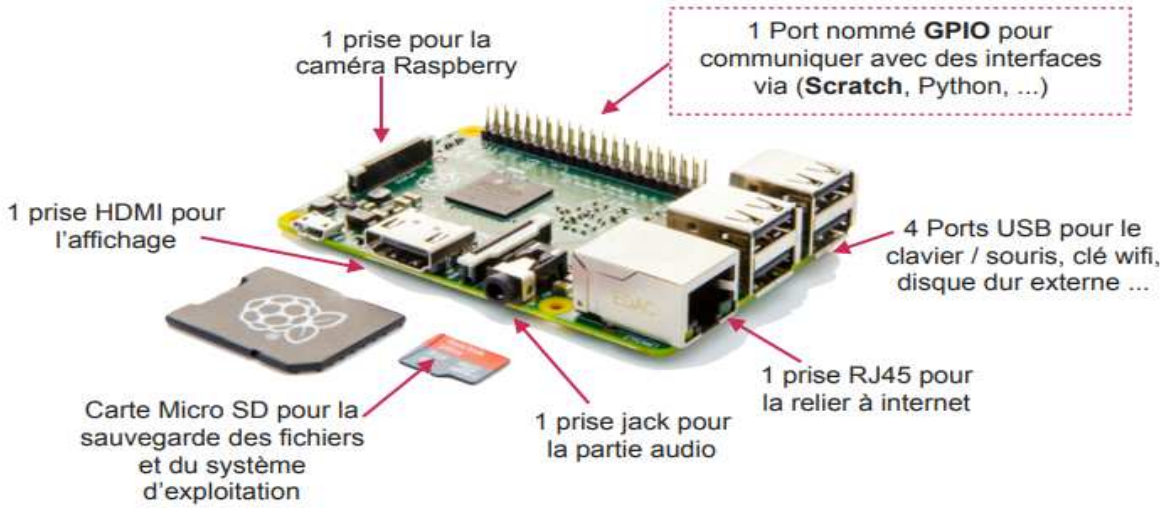


Figure III. 1 Raspberry *Pi* 3.

III.2.2. Les Ports GPIO

Le Raspberry *Pi* possède, en plus des connectiques classiques USB, HDMI, etc… un connecteur GPIO. GPIO signifie en anglais «General Purpose Input Output» et pourrait être traduit en français par entrées/sorties numériques. Ces entrées/sorties permettent d'étendre les fonctionnalités du Raspberry *Pi* en lui donnant la possibilité d'agir sur des LEDs ou des afficheurs LCD par exemple, lire l'état d'u un interrupteur, d'un capteur, etc... Ce connecteur GPIO dispose de différents types de connexion:

- Des broches utilisables en entrée ou sortie numérique tout ou rien.
- Des broches pour une interface I2C.
- Une interface SPI.
- Les broches *Rx* et *Tx* pour la communication UART avec les périphériques séries.
- Des broches pouvant être utilisé en PWM ("Pulse Width Modulation") permettant le contrôle de puissance ou PPM.

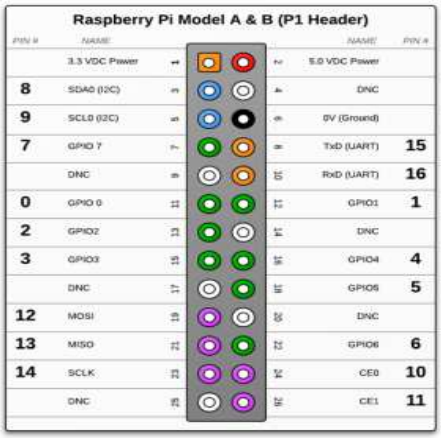


Figure III. 2 Pins GPIO et Leur Fonctions.

III.2.3. Aspects Logiciel

Toutes les applications compatibles avec le système d'exploitation et le processeur ARM, ou utilisant un environnement d'exécution virtuel (Java, émulateurs…) sont susceptibles de fonctionner : Python, MATLAB,… Les principales contraintes portent sur les performances du processeur et la mémoire vive disponible (256 Mo). Ce dernier point a toutefois été corrigé avec l'arrivée des versions embarquant 512 Mo de mémoire vive ce qui est suffisant pour lancer le processus du contrôle et le serveur web sur le système d'exploitation Raspbian.

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. La distribution recommandée est Raspbian. C'est un système d'exploitation libre basé sur Debian optimisé pour le matériel Raspberry *Pi*. Cependant Raspbian fournit plus qu'un simple système d'exploitation : il est livré avec plus de 35.000 paquets, des logiciels précompilés qui nous facilitent le développement soft. Il y a autre système d'exploitation crée par Microsoft spécialement pour le Windows 10 loT qui peut être installé sur notre carte Raspberry *Pi* 3.

Figure III. 3 Logo du système d'exploitation Raspbian et Windows 10 IoT.

III.3. Accès à distance au Raspberry *Pi*

On peut accéder à la carte Raspberry Pi via le protocole SSH et/ou VNC, si on connait l'adresse IP de la carte.

III.3.1. Choix de l'adresse IP du Raspberry *Pi*

Pour trouver l'adresse IP de notre Raspberry *Pi*, nous avons utilisé Advanced IP Scanner qui est un scanner de réseau local utilisé sous Windows. La figure suivante montre l'interface de cet outil et l'information fournie par ce dernier concernant le Raspberry *Pi* :

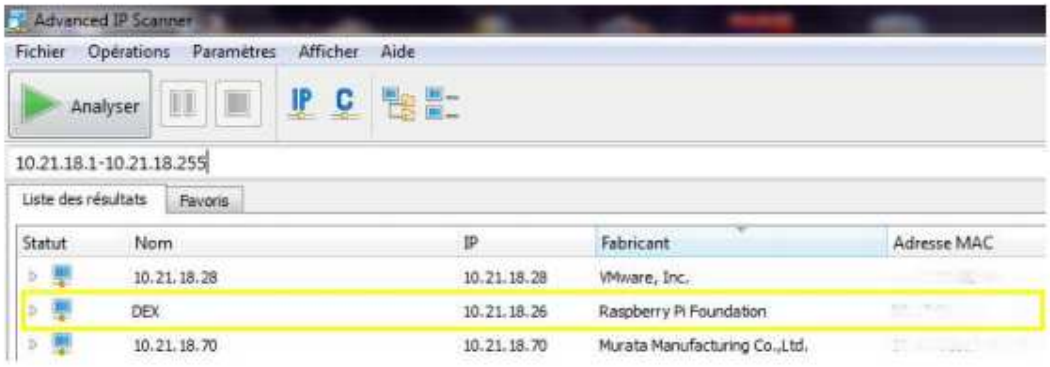


Figure III. 4 Adresse IP de la carte Raspberry *Pi.*

Après que nous avons trouvé l'adresse IP, il faut la fixer afin de nous permettre l'accès à distance au Raspberry *Pi*, pour cela on va suivre les étapes suivantes :

Configuration réseau sous Raspbian For Robots. Par défaut, il contient les valeurs suivantes :

auto lo

iface lo inet loopback

iface eth0 inetdhcp

allow-hotplug wlan0

iface wlan0 inet manual wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf

iface default inet dhcp

La signification de chaque ligne de ce fichier est la suivante :

- auto lo : va démarrer l'interface automatiquement lors de la séquence de boot.
- Iface lo inet loopback : définition de l'interface loopback (interface virtuelle indispensable sui permet de supprimer les paquets erronés)
- iface eth0 inet dhcp : l'interface eth0 (le port RJ45 du Raspberry) sera configurée en envoyant une requête DHCP sur le réseau. Donc IP dynamique.

Les quatre lignes suivantes permettent de configurer le Wi-Fi via le fichier de configuration de wpa_supplicant.

Cette configuration permettre de placer une adresse IP statique pour cela nous allons changer l'interface Ethernet (eth0) en mettant à jour la configuration de l'interface : IP, masque et passerelle comme indique la figure suivante :

```
root@dex:/etc# cd network/
root@dex:/etc/network# ls
if-down.d       if-pre-up.d  interfaces      run
if-post-down.d  if-up.d      interfaces.bkp
root@dex:/etc/network# cat interfaces
auto lo

iface lo inet loopback
iface eth0 inet static
address 10.21.18.26
netmask 255.255.248.0
gateway 10.21.16.100
dns-nameservers 8.8.8.8

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface wlan0 inet dhcp
```

Figure III. 5 Fichier de configuration réseau.

La ligne après gateway (passerelle) sert à définir les serveurs de noms de domaine (DNS) pour qu'on puisse se connecter à une ressource réseau externe (convertir un nom alphanumérique « par exemple www. Google.com» en une adresse réseau numérique « 172. 217.19.164»). Le DNS 8.8.8.8 est un DNS Google

Finalement le redémarrage de Raspberry *Pi*.

Alors à cette étape nous avons donné à notre carte une adresse IP fixe qui va nous servir à la manipulation de la carte à distance via le protocole SSH et ou VNC.

III.3.2. Connexion à distance via SSH

Secure Shell est un protocole de réseau crypté pour initier des sessions Shell textuelles sur des machines distantes de manière sécurisée. Cela permet à un utilisateur d'exécuter des commandes sur l'invite de commande d'une machine sans qu'ils soient physiquement présents à proximité de la machine. SSH a été créé en 1995 pour le principal but est de permettre la prise de contrôle à distance d'une machine à travers une interface en lignes de commande.

Pour faire le contrôle à distance sur le Raspberry *Pi* on doit installer le SSH côté d'un serveur SSH sur notre Raspberry *Pi* et d'un autre d'un client SSH sur notre ordinateur, mais heureusement le serveur SSH est déjà installé et activé par défaut sur la Raspberry *Pi*, ce qui reste seulement d'installer un client SSH sur notre ordinateur ; pour cela on va installer le logiciel Putty sous Windows qui est un client SSH.

Figure III. 6 Logiciel Putty.

Après son installation on entre l'adresse IP de la carte puis on clique sur «Open», et après l'authentification (en entrant le nom d'utilisateur pi et notre mot de passe), on se connectera directement sur l'invité des commandes de notre système d'exploitation Raspbian For robots comme l'indique la figure suivante:

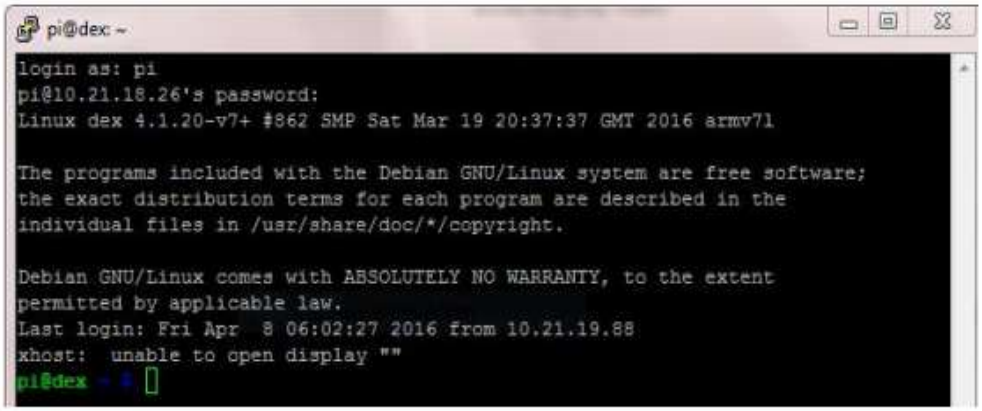


Figure III. 7 Invite de commande via SSH.

III.4. Raspberry Pi basé sur des langages de programmations

La programmation d'une carte Raspberry *Pi* implique souvent de travailler avec des images, de la vidéo, de l'audio et autres données de capteurs.

MATLAB et Simulink permettent aux utilisateurs d'analyser et visualiser rapidement ces données et de programmer leur carte Raspberry *Pi* en conséquence. Les produits prennent en charge deux méthodes de travail [34]:

- Lecture, écriture et analyse des données à partir des capteurs et caméras Raspberry *Pi.*
- Développement d'algorithmes s'exécutant de manière autonome sur la carte Raspberry *Pi.*

III.4.1. Analyse des données à partir la carte Raspberry *Pi*

Le support package MATLAB pour Raspberry *Pi* permet d'écrire des programmes MATLAB qui communiquent avec la carte Raspberry *Pi* et acquièrent des données à partir des ports GPIO de la carte, des caméras et autres appareils connectés. MATLAB étant un langage interprété de haut niveau, il permet de facilement prototyper et affiner des algorithmes pour notre projet basée sur le Raspberry *Pi*. MATLAB inclut des milliers de fonctions mathématiques et de tracés que nous pouvez utiliser dans le cadre de la programmation de Raspberry *Pi*, couvrant des domaines tels que le traitement vidéo et le traitement d'image, l'optimisation, les statistiques et le traitement du signal.

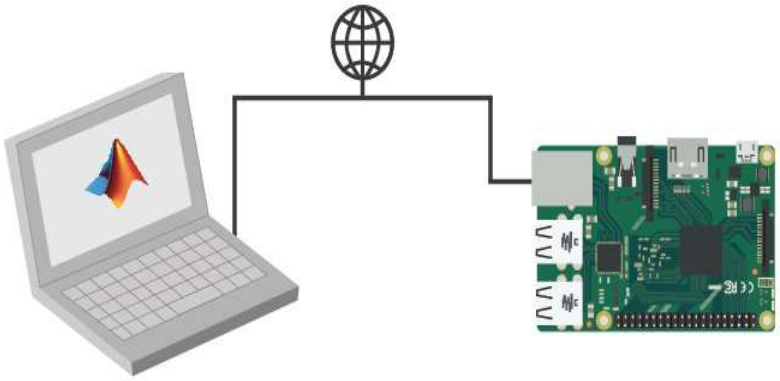


Figure III. 8 Connexion ordinateur et Raspberry *Pi*.

Avec le support package MATLAB pour Raspberry *Pi,* la carte Raspberry *Pi* est connectée à un ordinateur exécutant MATLAB. Le traitement est effectué sur l'ordinateur avec MATLAB.

III.4.2. Programmation Raspberry *Pi*

L'utilisation de langage de programmation MATLAB dans la carte Raspberry *Pi* permet d'effectué les opérations suivantes :

- Analyse des données du capteur Raspberry *Pi* à l'aide de milliers de fonctions pré-intégrées pour le traitement d'image, le traitement du signal, la modélisation mathématique, etc.
- Visualisation rapide des données grâce au vaste choix de tracés MATLAB.

- Utilisation du même logiciel pour programmer d'autres cartes matérielles, notamment Arduino.

III.5. Développer des algorithmes Simulink

Le support package Simulink pour Raspberry Pi permet de développer des algorithmes dans Simulink, un environnement de programmation par blocs pour la modélisation de systèmes dynamiques et le développement d'algorithmes, avec une exécution autonome sur la carte Raspberry *Pi*. Le support package intègre des blocs Simulink pour configurer la carte Raspberry *Pi*, envoyer et recevoir des paquets UDP et lire et écrire des données à partir des capteurs.

Une fois le modèle Simulink créé, nous pouvant le simuler, affiner les paramètres de notre algorithme pour arriver au bon réglage, et télécharger l'algorithme final pour une exécution autonome sur la carte. Utilisez le bloc MATLAB Function pour incorporer du code MATLAB dans un modèle Simulink.

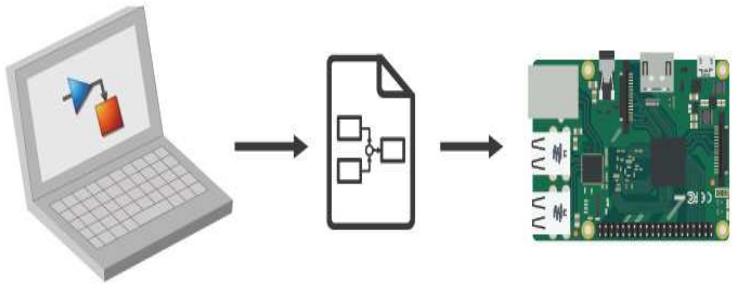


Figure III. 9 L'environnement de programmation pour Raspberry *Pi* par l'intermédiaire du support package Simulink.

Avec le support package Simulink pour Raspberry *Pi*, on peut développer des algorithme dans Simulink et le déployer sur la carte Raspberry *Pi* à l'aide de la génération de code automatique. Le traitement est alors effectué sur la carte Raspberry *Pi*.

Utiliser Simulink dans le cadre de la programmation Raspberry *Pi* permet d'effectuer les opérations suivantes :

- Développement et simulation des algorithmes dans Simulink et utilisation de la génération de code automatique pour exécuter les algorithmes sur la carte.
- Incorporation du traitement du signal, de la conception de contrôleurs, de la logique de contrôle, et d'autres programmes avancés pour les mathématiques et l'ingénierie dans les projets de programmation Raspberry *Pi*.

- ❖ Réglage et optimisation interactive des paramètres durant l'exécution des algorithmes sur la carte Raspberry *Pi.*

III.5.1. Seuillage

La caméra prend des photos à intervalles de temps réguliers (figure III.10). On effectue alors un seuillage, qui permet d'obtenir une image ne contenant que du blanc et du noir (Objet/ arrière-plan). On définit pour cela un seuil et on compare la luminance de chaque pixel à ce seuil. Dans un cas on remplace ce pixel par du blanc, dans l'autre par du noir. L'image binaire obtenue est noir eaux endroits qui n'ont pas d'objet, et blanche où il ya eu un objet. On compare alors le nombre de pixels blancs (ceci peut se faire par une moyenne sur la luminance dans l'image) à un seuil bien choisi et prédéfini. Si ce seuil est dépassé on indique qu'un objet a été détecté par un signal lumineux.

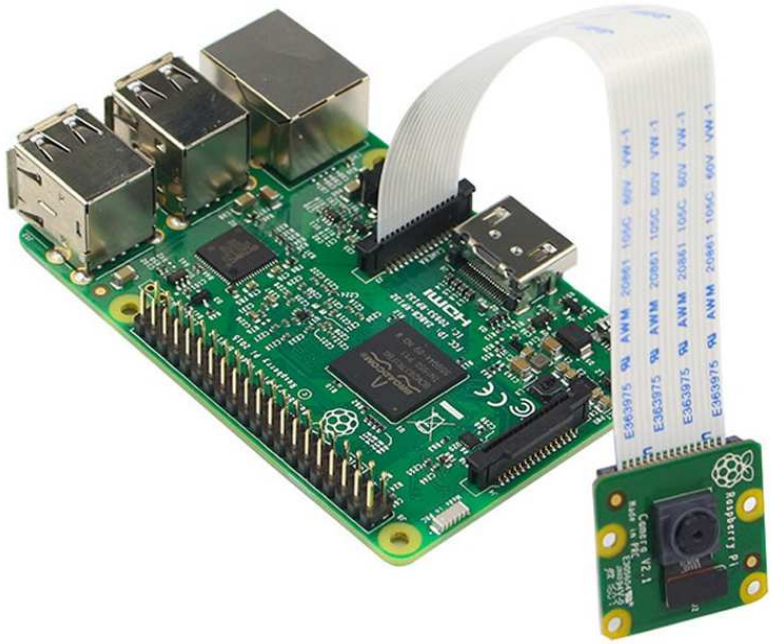


Figure III. 10 Raspberry *Pi* 3 et camera.

Le seuillage est une technique de segmentation simple, non contextuelle et efficace :

- ✓ Seuillage d'intensité.
- ✓ Classification des pixels en deux catégories.
- ✓ Création d'une image binaire (binarisation).
- ✓ Le seuillage peut utiliser un seuil soit fixé soit adaptatif.
- ✓ Diverses techniques ont été imaginées pour définir automatiquement ce seuil mais aucune n'est complètement robuste.

Les deux méthodes qui seront étudiées par la suite (fixe, et adaptative) permettent de faire une séparation Objet et arrière-plan d'image (classe 1 et classe 0). Pour chaque méthode en utilise les différents changements d'espace couleur (RGB, YCbCr, YIQ, HSI, HSV).

Pour démontrer l'efficacité des méthodes de seuillage dans le domaine de détection d'objets, nous avons besoin d'un critère de précision. Dans notre cas on mesure le taux de classification (ou taux de réussite) *Acc* dans chaque plan de l'image originale (ou après l'application de changement d'espace couleur) par l'expression suivante :

$$Acc(\%) = 100 \times \frac{1}{C} \sum_{C=1}^{2} \frac{B_C}{N_C} \qquad \text{(III.1)}$$

L'erreur de classification est donnée par l'expression suivante :

$$Err(\%) = 100 - Acc \qquad \text{(III.2)}$$

C : nombre de classes, B_c : Pixels bien classés (B_1 : classe 1, B_2 : classe 0), et N_c: nombre de pixels des classes cibles.

Dans la littérature scientifique, il n'y a pas un espace couleur qui soit meilleure qu'une autre. Tout dépend de l'application utilisée. Pour cela nous avons utilisé cinq différents types d'espace couleur sur les images de test, ensuite une méthode de seuillage fixe pour détecter la présence ou l'absence des objets couleur (reconnaissance d'objet), l'image de test est représenté dans la figure III.11 (Objets : rouge, vert, bleu, jaune). Les quatre objets/ arrière-plan sont représentées dans la figure III.12.

Figure III. 11 Image de test avec quatre objets couleur (rouge, vert, bleu, jaune).

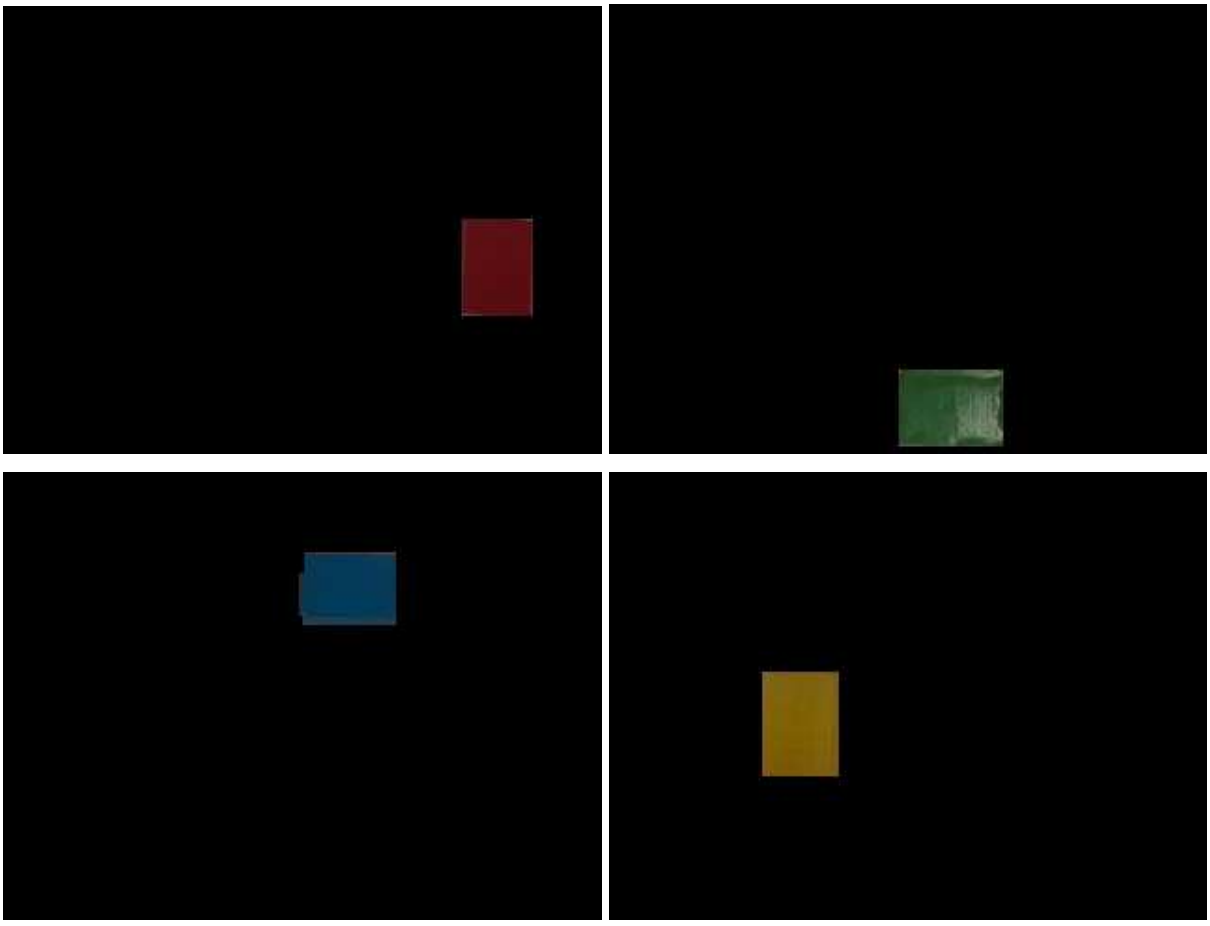


Figure III. 12 Les objets couleurs: rouge, vert, bleu, et jaune.

III.5.1.1. Seuillage fixe

La méthode consiste à prédéfinir un seuil d'après l'histogramme de l'image qui approche le mieux l'image binarisée de l'originale. Si les coefficients obtenus est inférieurs ou égaux à TH, ces coefficients mis à 0 (arrière-plan), sinon 1 (Objet). On varie le seuil TH de 10 jusqu'à 250.

Pour l'image de test, on a obtenu un taux de classification (Acc) minimum et maximum de l'ordre de 0 et 100% pour les différents types d'espace couleur, mais aucune discrimination ou séparation entre les couleurs (corrélation entre les objets couleurs) pour les cinq espaces couleur (figures III.13 à III.17).

Les résultats visuel est très mauvais donc elle est rarement utilisée. Les figures III.18 et III.19 représentes deux exemples de seuillage fixe (TH=100) dans les espaces couleurs RGB et HSV.

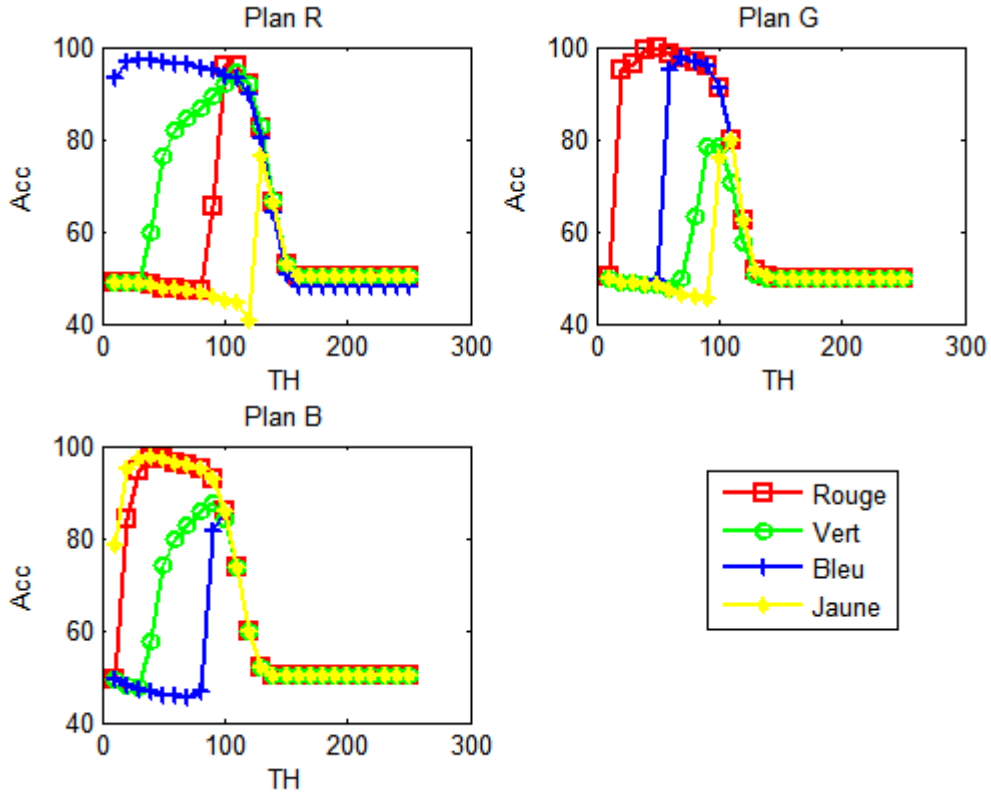


Figure III. 13 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-TH pour l'espace couleur RGB de l'image de test.

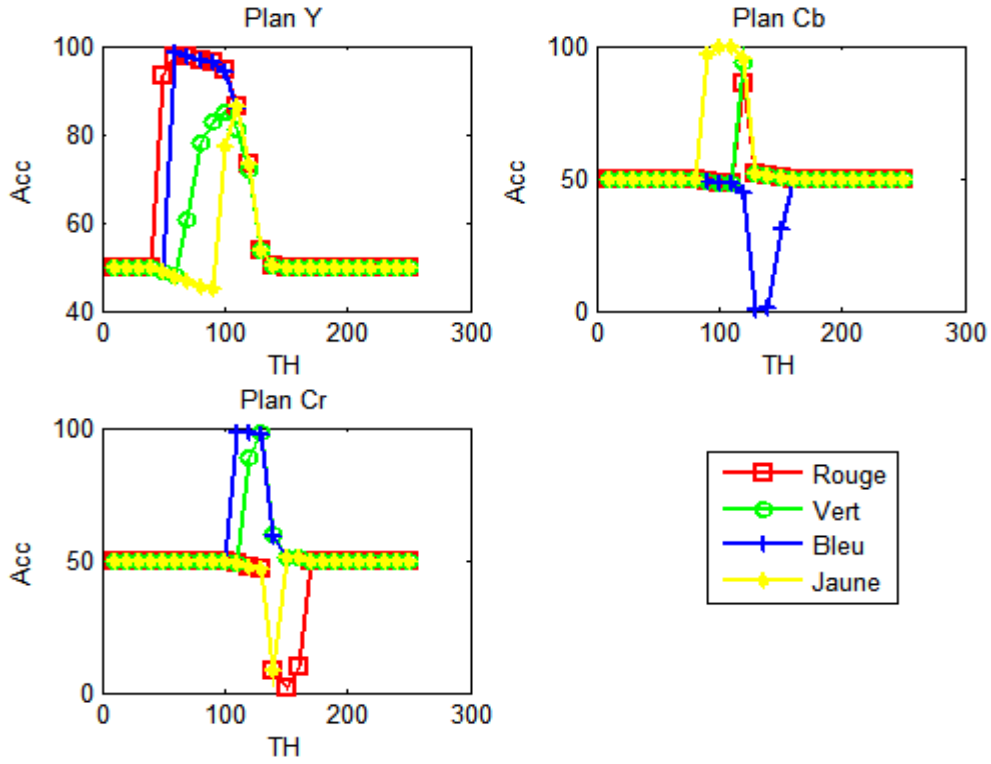


Figure III. 14 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-TH pour l'espace couleur YCbCr de l'image de test.

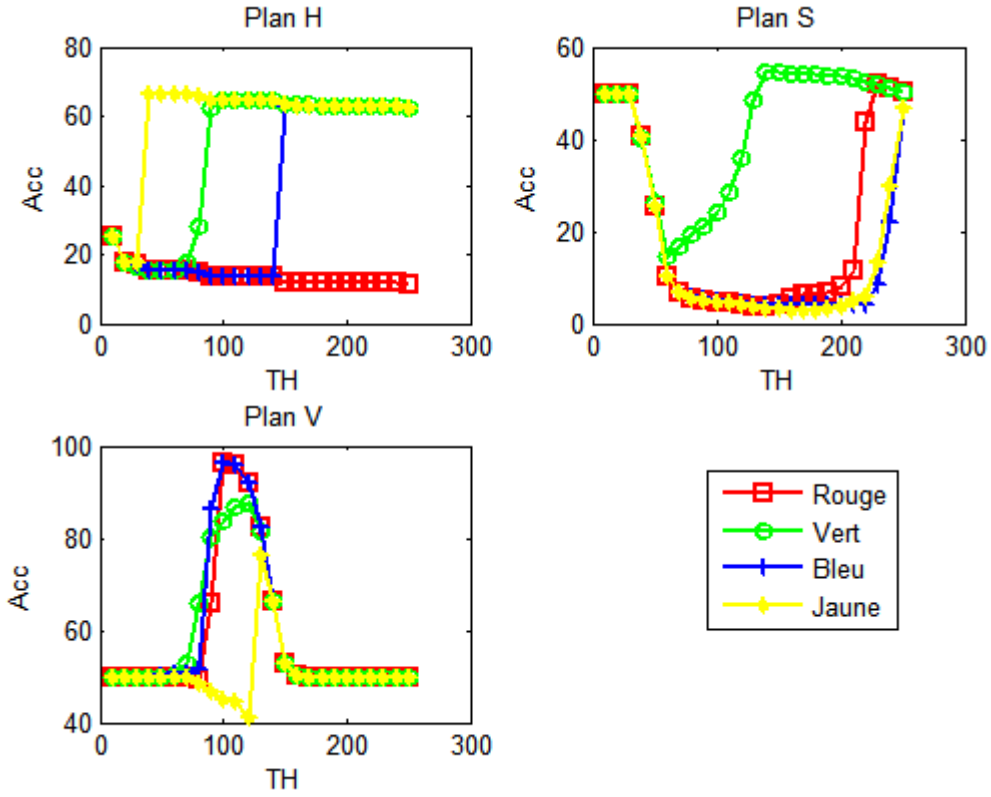


Figure III. 15 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-TH pour l'espace couleur HSV de l'image de test.

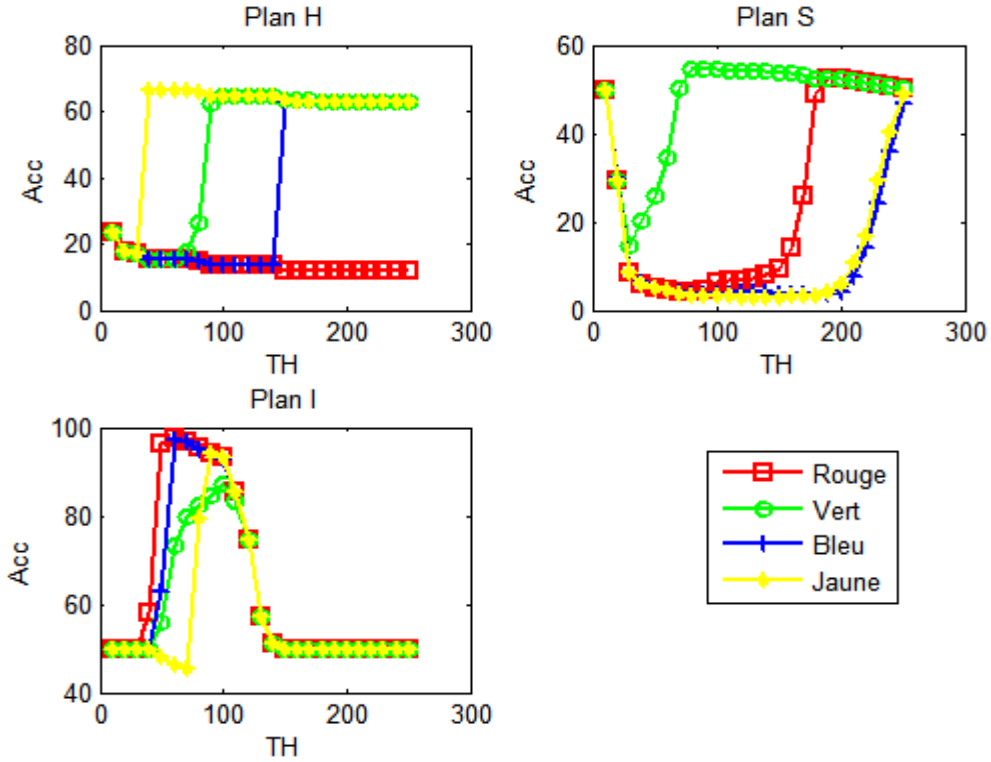


Figure III. 16 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-TH pour l'espace couleur HSI de l'image de test.

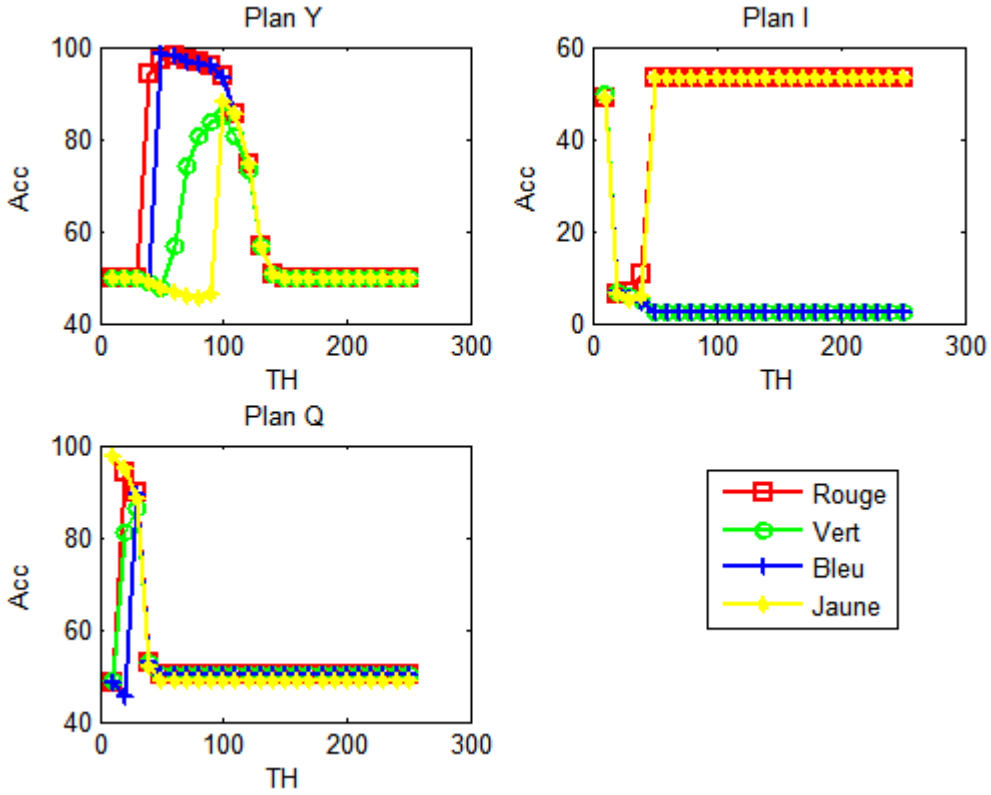


Figure III. 17 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-TH pour l'espace couleur YIQ de l'image de test.

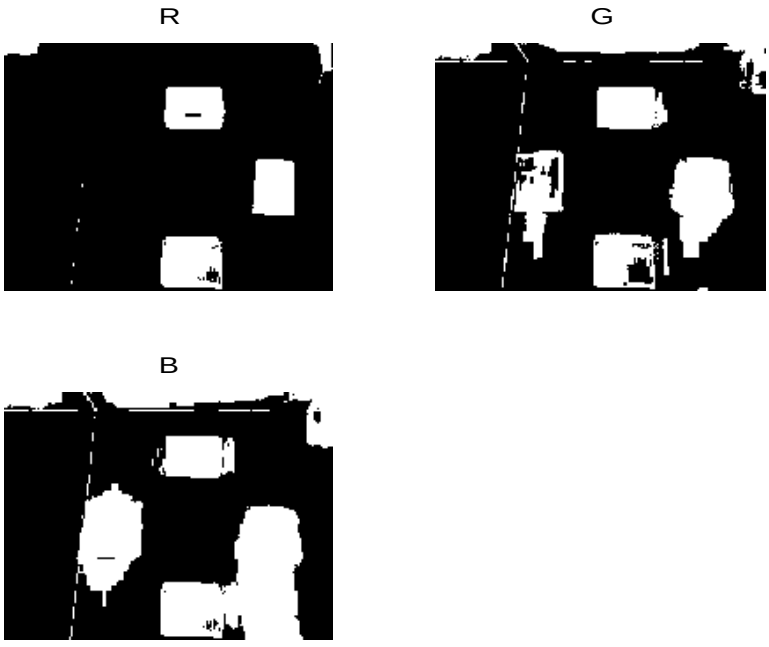


Figure III. 18 Résultats visuel de détection d'objets couleur (rouge, vert, bleu, jaune) dans l'espace couleur RGB de l'image de test pour un seuil fixe TH=100.

Figure III. 19 Résultats visuel de détection d'objets couleur (rouge, vert, bleu, jaune) dans l'espace couleur HSV de l'image de test pour un seuil fixe TH=100.

III.5.1.2. Seuillage adaptative

Les figures III.20 à III.24 montrent l'évolution du taux de classification pour les objets couleurs (Rouge, Vert, Bleu, Jaune) en fonction des paramètres THmin, THmax appliqués sur les cinq espaces couleur (RGB, YCbCr, HSV, HSI, YIQ) sur l'image de test, respectivement, la Figure III.20 montre que aucun discrimination des intervalles de variation des seuils entre les objets couleurs pour les trois plans de couleurs (R, G, B), les variations des seuils pour la détection de l'objet rouge dans les plans R, G et B sont THmin=[70, 10, 10] et THmax=[100, 50, 40], par contre dans l'espace couleur HSV plusieurs objets sont détectés (l'objet rouge se trouve dans le plan S, et les objets couleurs vert, bleu et jaune sont apparus dans le plan H).

Les résultats pour la détection d'objet de l'image de test avec les espaces couleurs YCbCr, HSV, HSI et YIQ sont présentés dans les figures III. 21, 22, 23 et 24.

Les tableaux suivants (III.1, III.2, III.3, III.4 et III.5), illustres les meilleurs seuils obtenus pour la détection d'objets couleurs dans les différentes espace couleurs (RGB, YCbCr, HSV, HSI, YIQ). On peut également noter que le fait d'appliquer un seuil adaptative à amélioré les performances de détection d'objets sur les différentes type d'espace couleurs.

Tableau III. 1 Les meilleurs taux de classification pour la détection d'objets dans l'espace couleur RGB.

Espace	***R***			***G***			***B***		
Objets	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**
Rouge	70	100	98.79	10	50	99.43	10	40	97.89
Vert	30	110	95.66	60	100	81.04	30	90	89.94
Bleu	10	40	52.49	40	70	98.79	40	100	90.34
Jaune	120	130	86.16	80	110	84.03	10	40	65.39

Tableau III. 2 Les meilleurs taux de classification pour la détection d'objets dans l'espace couleur YCbCr.

Espace	***Y***			***Cb***			***Cr***		
Objets	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**
Rouge	10	70	97.78	110	120	87.59	150	170	98.10
Vert	60	100	86.87	110	120	96.12	110	130	99.14
Bleu	50	60	99.52	130	160	99.47	10	120	98.17
Jaune	90	110	91.11	10	100	99.53	140	150	94.29

Tableau III. 3 Les meilleurs taux de classification pour la détection d'objets dans l'espace couleur HSV.

Espace	***H***			***S***			***V***		
Objets	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**
Rouge	10	10	50	140	230	98.10	80	100	96.80
Vert	50	100	99.60	60	140	90.30	10	120	87.48
Bleu	130	150	99.54	190	250	92.89	10	100	96.16
Jaune	20	40	98.74	170	250	93.86	120	130	86.04

Tableau III. 4 Les meilleurs taux de classification pour la détection d'objets dans l'espace couleur HSI.

Espace	***H***			***S***			***I***		
Objets	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**
Rouge	10	10	50	70	190	98.23	10	60	97.51
Vert	50	100	99.60	30	90	89.45	40	100	87.52
Bleu	130	150	99.45	180	250	93.49	40	60	97.71
Jaune	20	40	98.74	130	250	96.10	70	90	98.52

Tableau III. 5 Les meilleurs taux de classification pour la détection d'objets dans l'espace couleur YIQ.

Espace	***Y***			***I***			***Q***		
Objets	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**	**THmin**	**THmax**	**Acc**
Rouge	10	60	98.02	20	50	97.10	10	20	95.52
Vert	50	100	86.99	10	10	50	10	30	86.40
Bleu	40	50	99.52	10	10	50	20	30	94.04
Jaune	80	100	92.55	30	50	98.09	10	10	50

La méthode de seuillage adaptative montre toujours le bon comportement de taux de classification ce qui procède à de bons résultats précis. En comparaison avec les résultats obtenus par la méthode seuillage fixe, cette section a démontré l'efficacité et la fiabilité de cette méthode.

Nous avons appliqués cinq types de changement d'espace couleur, la question est laquelle des techniques à choisir parmi toutes les approches proposées ? Nous proposons une solution possible pour résoudre ce problème qui est la fusion des résultats fournis par un ensemble de différentes méthodes de changement d'espace couleur.

Le tableau III.6 illustre les meilleurs résultats obtenues pour les différents espace couleur (RGB, YCbCr, HSV, HSI, YIQ) ce qui nous permet de remarquer que dans le cas de l'objet rouge le meilleur espace couleur et plan est celui de l'application de RGB sur l'image brute, le plan est G. Pour les deux objets couleur vert et bleu, le changement d'espace couleur le plus adapté est l'espace HSV avec le plan H. Pour la quatrième couleur le meilleur espace couleur est YCbCr, le plan est Cb.

La figure III.25 représente les meilleures combinaisons des espaces couleurs et seuillage adaptative. Les résultats obtenus montrent que par la combinaison d'un ensemble de techniques de changement d'espace couleur, est un moyen efficace pour améliorer non seulement la robustesse du processus de détection d'objet, mais aussi la précision finale de taux de classification.

Tableau III. 6 Les meilleurs résultats obtenues de la détection d'objets pour les différents espaces couleurs RGB, YCbCr, HSV, HSI et YIQ.

Objets couleurs	**Espace couleur**	**Plan**	**THmin**	**THmax**	**Acc**
Rouge	RGB	G	10	50	99.43
Vert	HSV	H	50	100	99.60
Bleu	HSV	H	130	150	99.54
Jaune	YCbCr	Cb	10	100	99.53

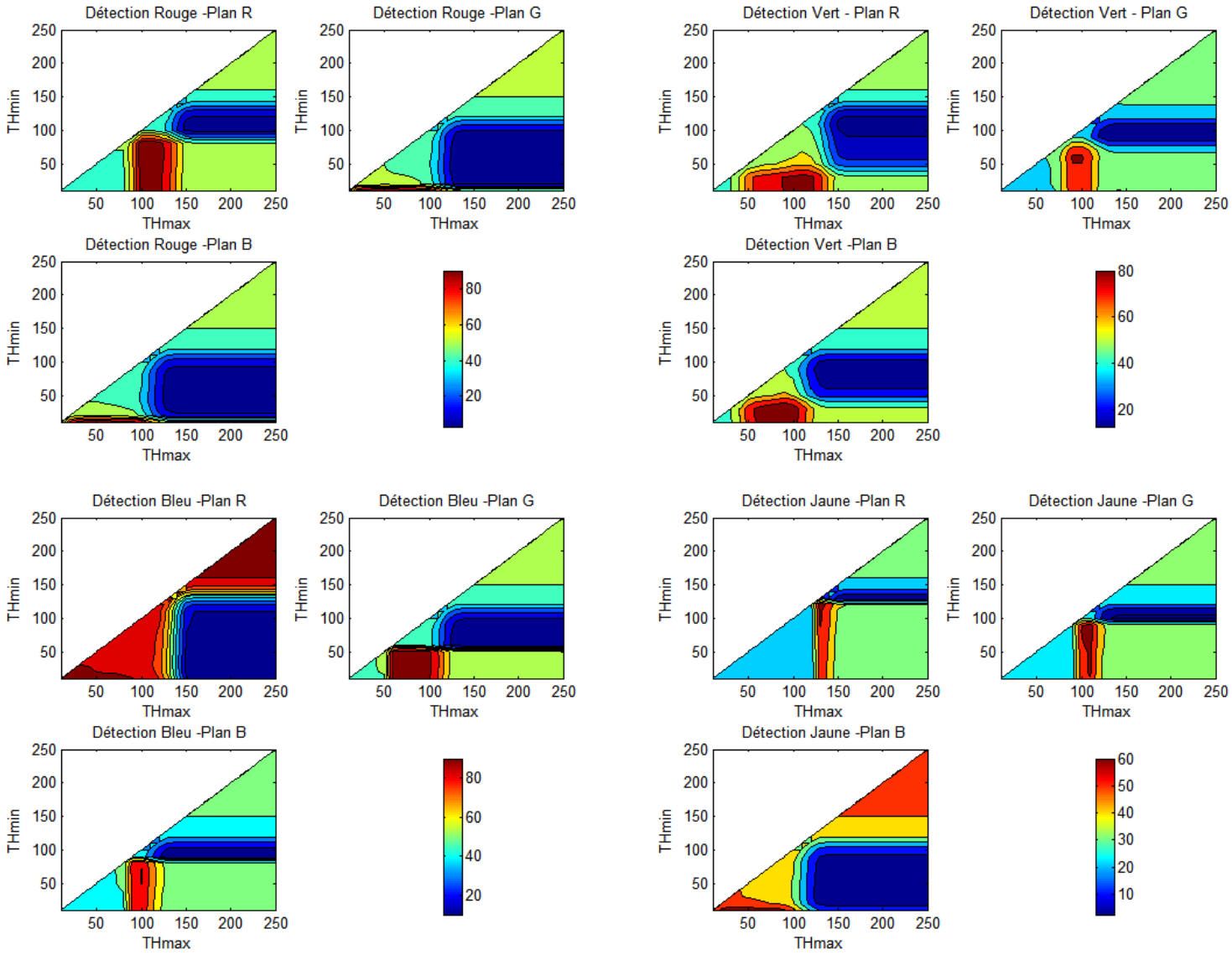


Figure III. 20 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-THmin-THmax pour l'espace couleur RGB de l'image de test.

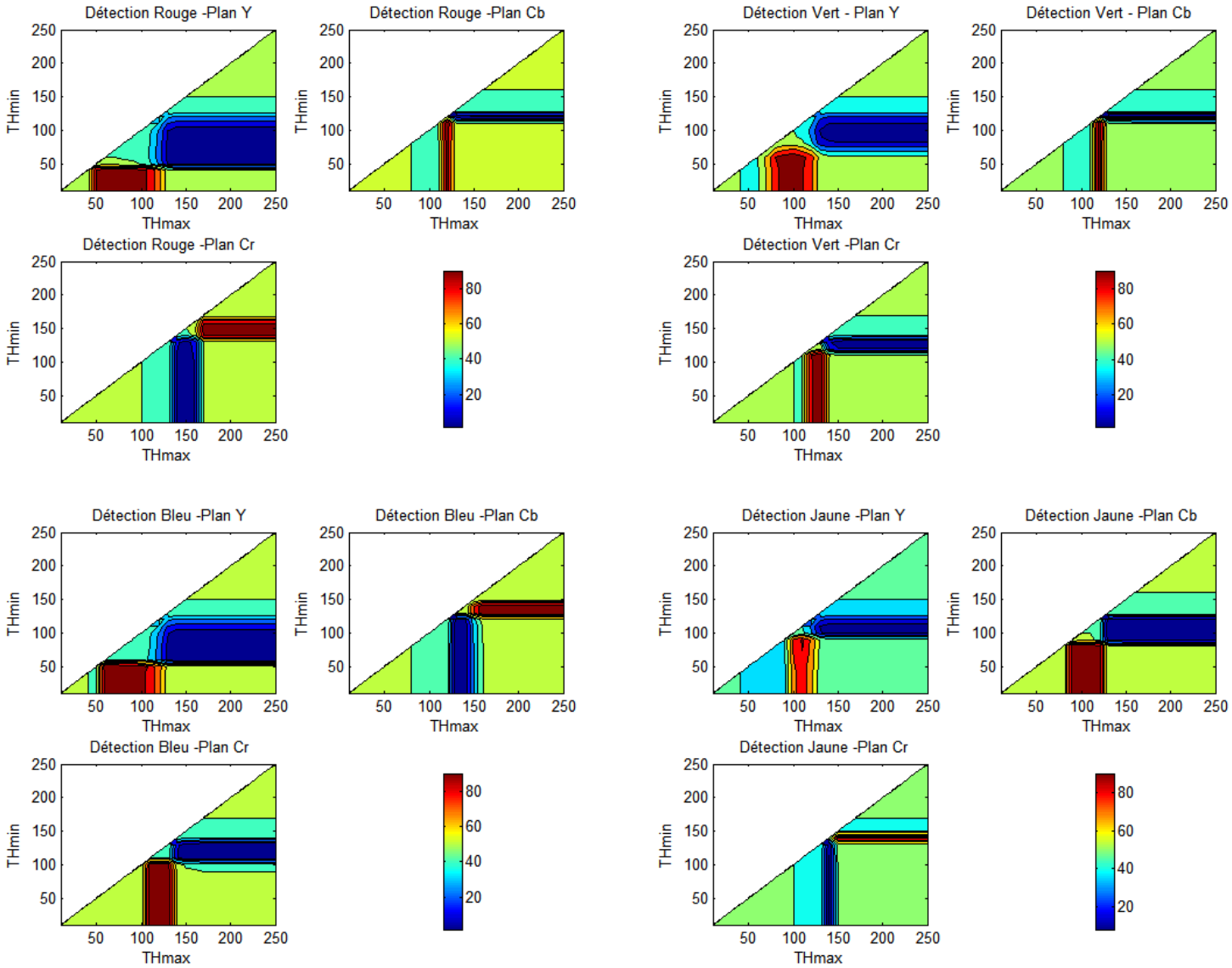


Figure III. 21 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-THmin-THmax pour l'espace couleur YCbCr de l'image de test.

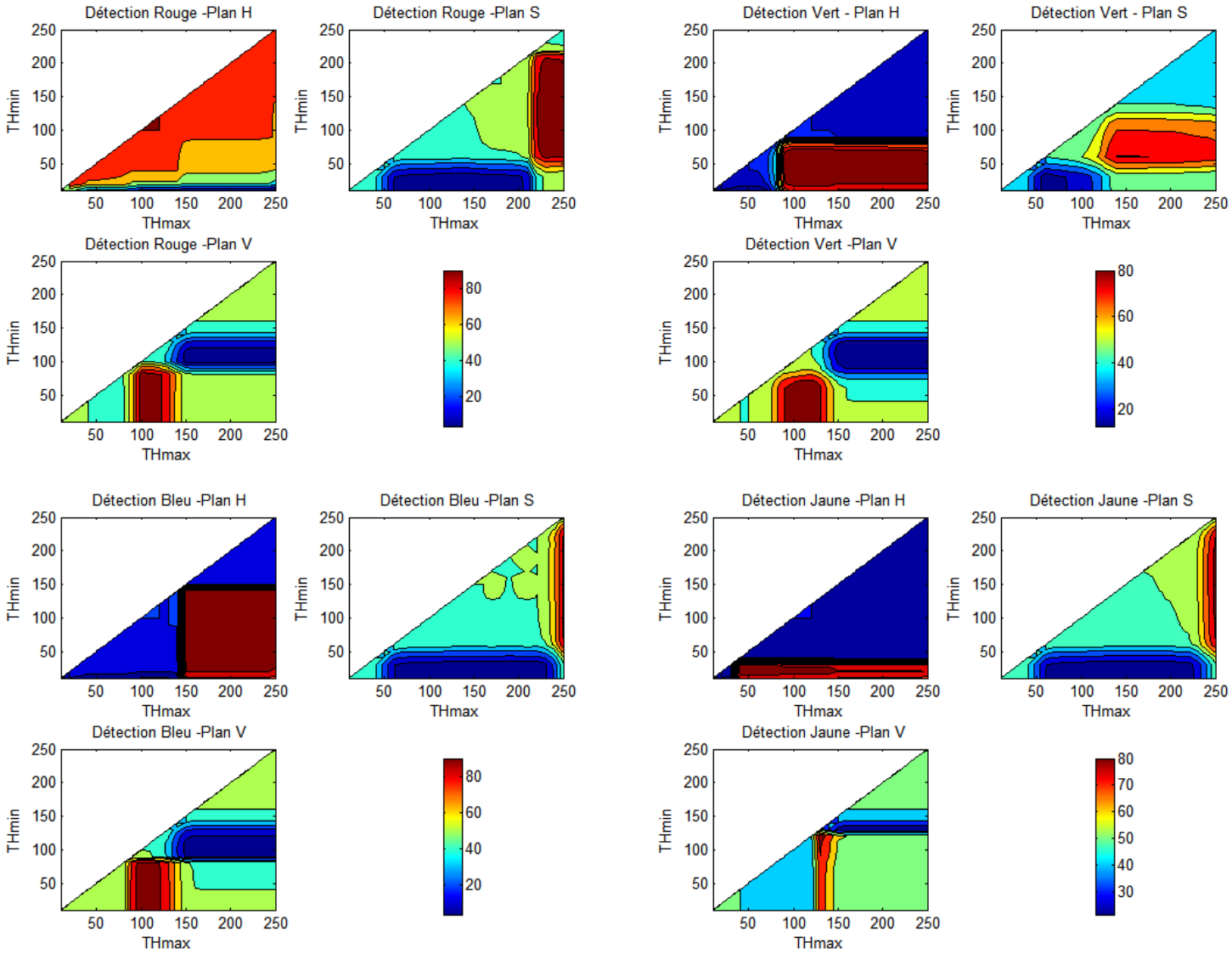


Figure III. 22 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-THmin-THmax pour l'espace couleur HSV de l'image de test.

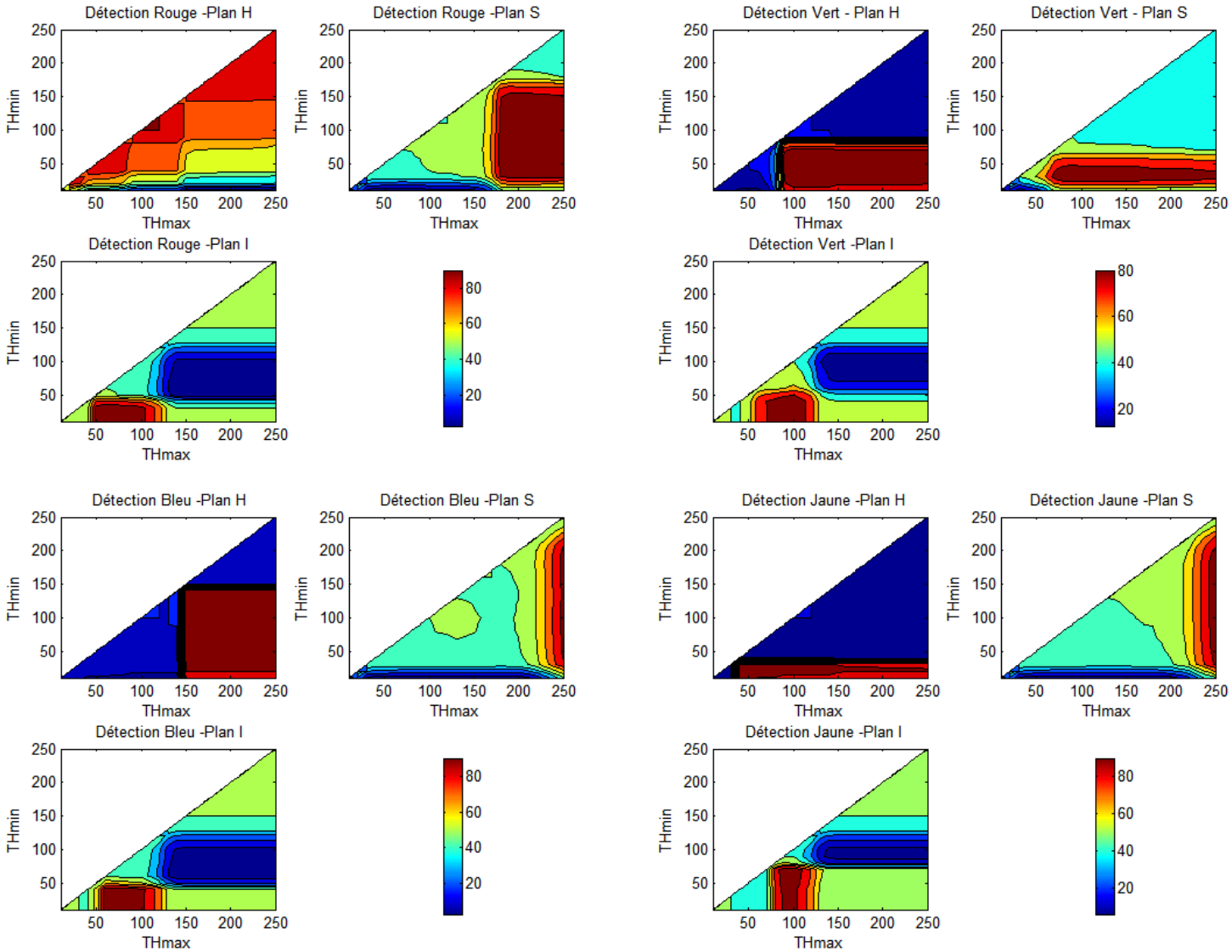


Figure III. 23 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-THmin-THmax pour l'espace couleur HSI de l'image de test.

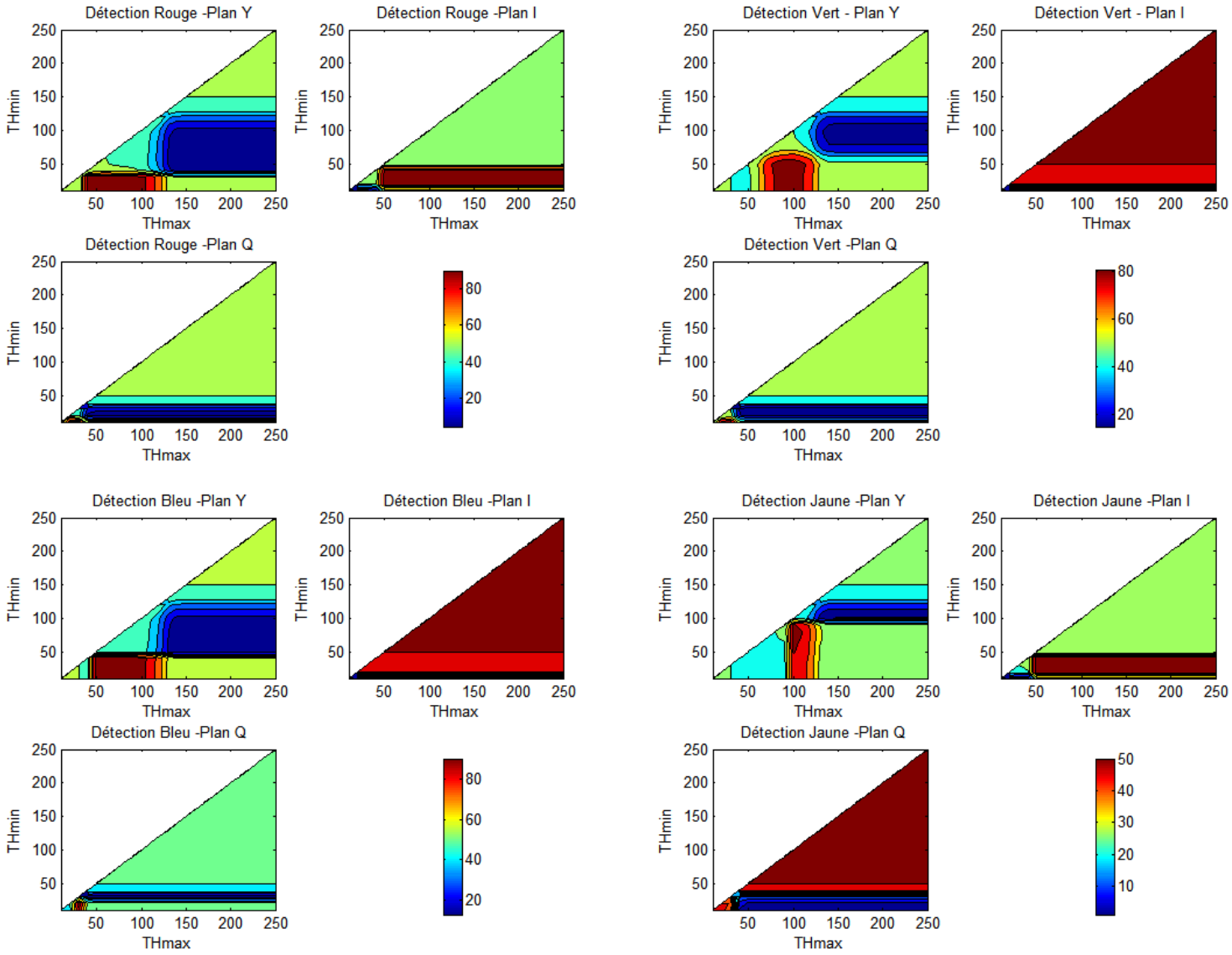


Figure III. 24 Résultats de détection d'objets couleur (rouge, vert, bleu, jaune) en termes de Acc-THmin-THmax pour l'espace couleur YIQ de l'image de test.

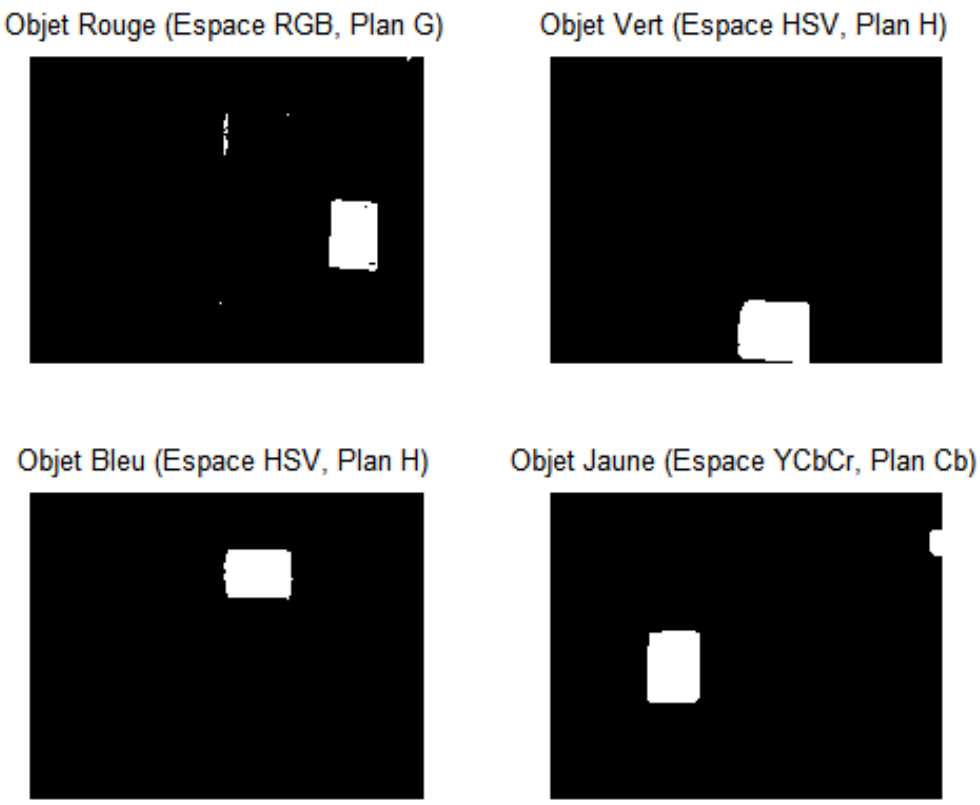


Figure III. 25 Détection d'objets couleurs Rouge, Vert, Bleu et Jaune de l'image de test.

III.6. Conclusion

Dans ce chapitre dédié à la présentation du Raspberry *Pi*, nous avons présenté les divers de la carte Raspberry *Pi* 3, et nous avons expliqué les portes GPIO et l'aspect logiciel de l'adresse IP du Raspberry *Pi* 3, pour utiliser ce dernier à distance avec une application via SSH.

Ensuite, une application de la détection d'objets sur le capteur d'image de Raspberry *Pi* 3. La détection d'objets couleurs est basée sur deux concept différents le seuillage (fixe, adaptative) et le changement d'espace couleurs. Les meilleures combinaisons entre le changement d'espace couleur et le seuillage sont examinés dans le chapitre IV.

Pour tester les performances de notre stratégie, dans le chapitre IV nous utiliseront un robot mobile qui fonctionne en mode autonome grâce à notre algorithme de détection.

Chapitre IV

Réalisation pratique

IV. 1. Introduction

Ce mémoire consiste à l'étude et la réalisation d'un suiveur visuel d'objets par un capteur d'image. Pour atteindre l'objectif, plusieurs stratégies de classification sont possibles (seuillage, détection de couleurs). Comme application, nous comptons utiliser un robot mobile qui fonctionne en mode autonome. La plateforme utilisée est constituée par le Raspberry *Pi* 3, Arduino Mega, Arduino Nano et une caméra. Le robot est muni d'une communication Wi-Fi pour accéder au réseau local et une caméra pour transmettre une vidéo en temps réel à un ordinateur qu'on peut accéder à distance, et ainsi nous avons utilisés un autre type de communication sans fil à base des modules radiofréquence.

Dans les sections suivantes, nous allons essayer de présenter une description détaillée de la réalisation de notre système.

IV. 2. Schéma globale

Dans ce mémoire, le contrôle du robot mobile est effectué grâce à deux méthodes : la fusion des espaces couleurs et seuillage adaptative, détaillé dans le chapitre III. L'image d'environnement est capturée par une caméra qui est interfacée avec le Raspberry *Pi* 3. L'image d'environnement apparaît à l'écran pour que l'utilisateur puisse traiter et visualiser. Les informations du Raspberry *Pi* sont envoyées à l'unité émettrice de l'Arduino Nano et sont transmises sans fil à l'unité réceptrice de l'Arduino Mega. La communication sans fil se fait via le module émetteur-récepteur nRF24L01, qui est connecté aux unités d'émission et de réception des cartes Arduino Nano et Mega. Les informations reçues par Arduino Mega (distance et la couleur d'objet) sont visualisés dans l'afficheur LCD 16×2.

L'ensemble de notre système proposé est divisé en trois parties, à savoir une unité de traitement (commande centrale), unité de contrôle (Raspberry *Pi*, Arduino Mega, Arduino Nano) et une unité de transmission (Wi-Fi, Radiofréquence).

Le schéma synoptique suivant va nous permettre de mieux comprendre le fonctionnement global du système étudié (Figure IV.1):

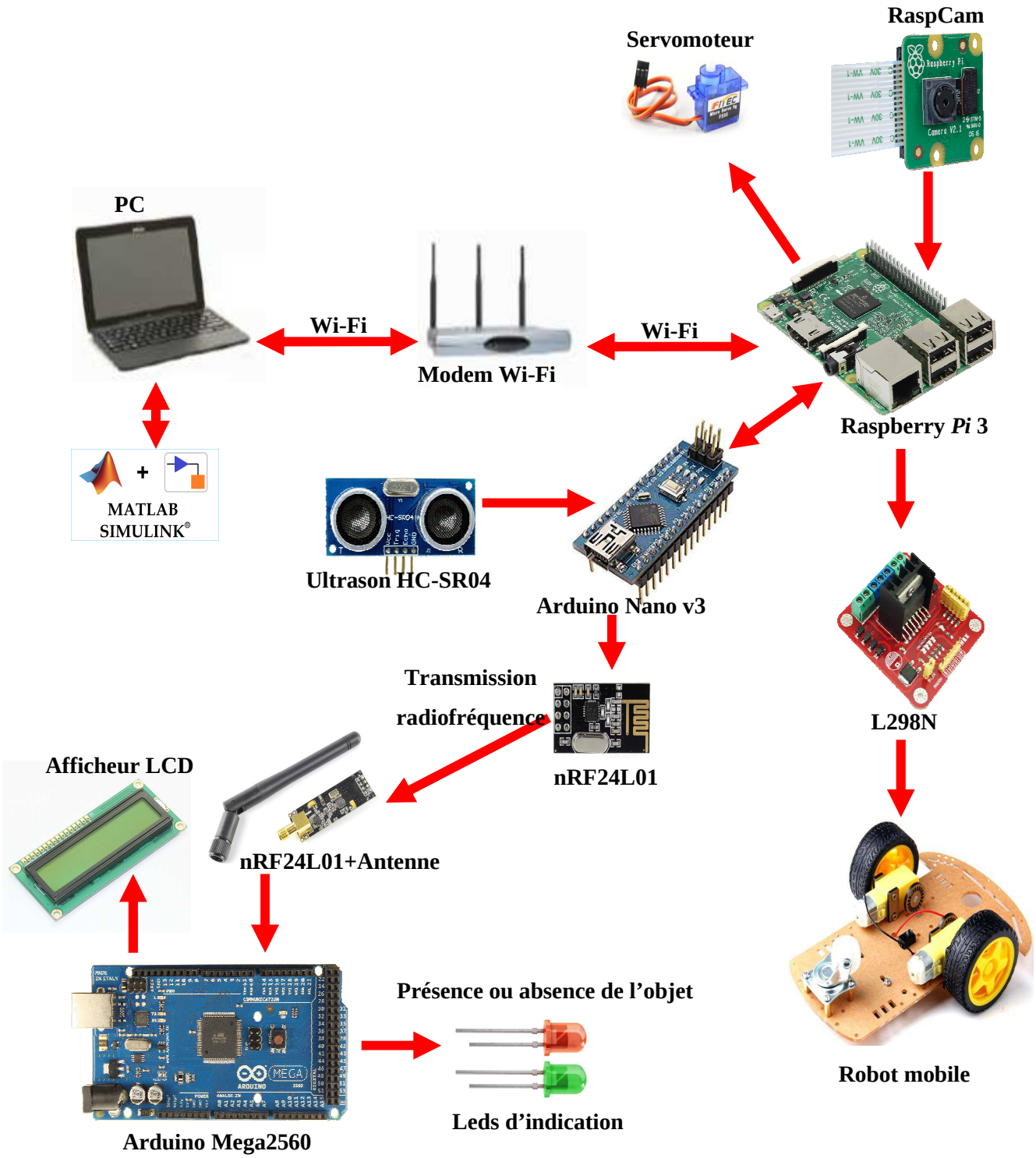


Figure IV. 1 Schéma globale du système de détection d'objets.

IV.2.1. Unité de traitement

Un ordinateur peut être utilisé pour répondre à toutes les exigences informatiques très lourdes. Cet ordinateur collecte les informations d'une suite de capteurs (par exemple : caméras). Les informations recueillies sont utilisées à plusieurs fins. Cela inclut l'adaptation objective, l'optimisation, la transmission/diffusion d'informations et la génération de commandes pouvant être émises au système. Dans ce mémoire, nous utilisons simplement un ordinateur portable et le logiciel MATLAB-SIMULINK.

IV.2.2. Unité de contrôle

Pour prendre des décisions, une série de capteurs doit être disponible tels que : ultrasons, radiofréquence. Cette suite fournit des informations sur l'état de notre environnement, qui pouvant être utilisées par l'unité de traitement, et ensuite contrôlé notre système. Dans le cadre de ce mémoire, la détection globale est obtenue en envoyant une vidéo en temps réel du robot à l'ordinateur portable.

IV.2.3. Unité de communication

Dans notre cas, nous avons utilisée deux types de communication sans fil : Wi-Fi et radiofréquence (nRF24).

IV. 3. Description du kit de robot

La plate-forme utilisée est très simple, composée de deux moteurs à courant continu (CC), deux roues et une roue folle, servomoteur qui joue le rôle des actionneurs, et toutes les pièces nécessaires pour assembler le châssis (Figure IV.2). Il comprend également un boîtier où peuvent se loger trois piles. Le tableau IV.1 liste certaines caractéristiques mécaniques du robot mobile. Dans la section suivante, le principe de fonctionnement est détaillé et expliqué.

Tableau IV. 1 Caractéristique du robot mobile.

N°	Désignation	Caractéristiques	Quantité
1	Châssis	Dimension 21x15 cm	1
2	Moteur avec réducteur	Tension: 3V ~ 12V DC - Vitesse sans charge: 170 TRS/MIN (3V) - Réduction : 1:48 - Courant Max 250mA	2
3	Roue	-Diamètre 6.5cm - Hauteur 2.5cm	2
4	Encodeur de Vitesse	Diamètre 2.5 cm	2
5	Forme – T pour fixation Moteurs	/	4
6	Boite de Batterie(Pile)	04 Piles type AA rechargeable	4
7	Interrupteur	/	1
8	Kit vis+écrou	/	1

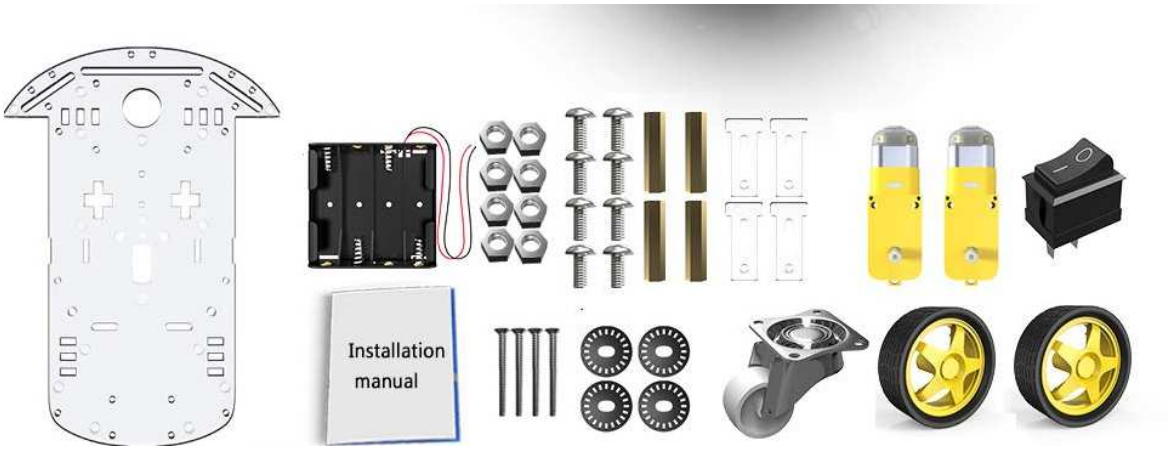
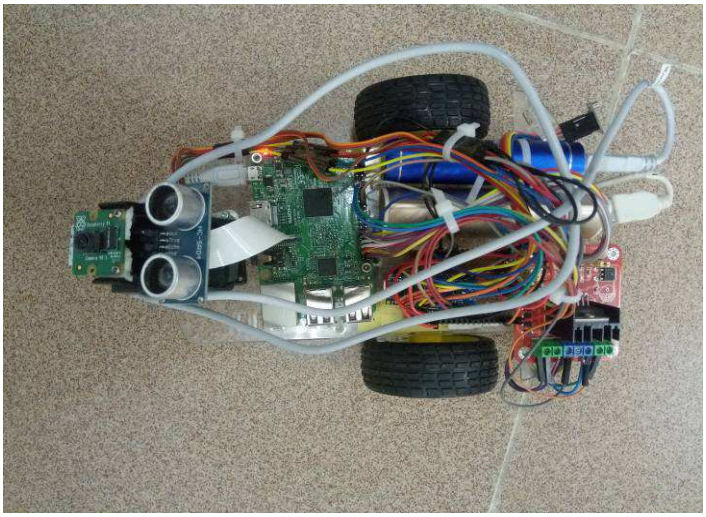
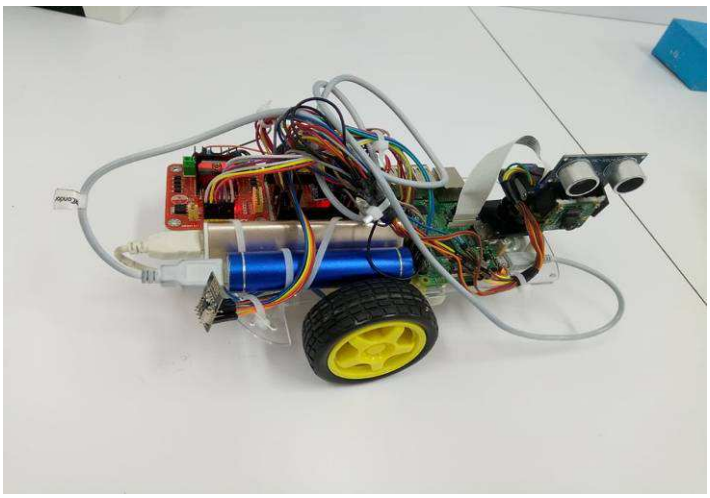


Figure IV. 2 Le robot avant et après l'assemblage.

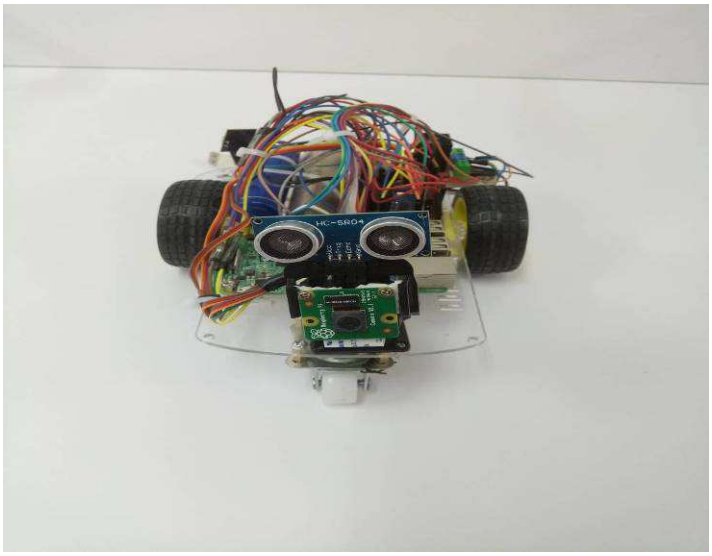
La figure suivante illustre le robot mobile après l'assemblage des différentes composants (moteur CC, camera, Arduino Nano, Ultrason, L298N , Raspberry *Pi* 3) :



(a)



(b)



(c)

Figure IV. 3 Robot mobile après l'assemblage, (a) Face de haut, (b) Face de l'avant droit, (c) Face en avant.

IV. 4. Fonctionnement du système

La réalisation matérielle est faite en premier lieu, chaque module de notre montage est réalisé et testé séparément, les montages sont d'abord construits sur des "breadboard" ou cartes de montage expérimental. Après les avoir expérimentés et adoptés séparément, nous les avons regroupés et réalisés sur trois circuits (capteur HC-SR04, transmission radiofréquence, Raspberry Pi 3…). Ces derniers sont faits en utilisant le logiciel de dessin des schémas électroniques ‘‘***fritzing***’’.

IV.4.1. Capteurs HC-SR04

Nous avons placé un capteur ultrasons HC-SR04, Il est sur le coté avant du robot qui permet de calculer les distance «gauche-avant-droite» car il est placé sur un servomoteur.

Les capteurs ultrasons fonctionnent en mesurant le temps de retour d'une onde sonore émise par le capteur. La vitesse du son étant à peu près stable, on en déduit la distance à l'obstacle.

Pour cette partie, nous avons assemblé la carte Arduino Nano et le capteur sonar à Ultrasons HC-SR04, comme montré dans les figures IV.4 et IV.5. La carte Arduino Nano est programmée dans le logiciel Arduino IDE.

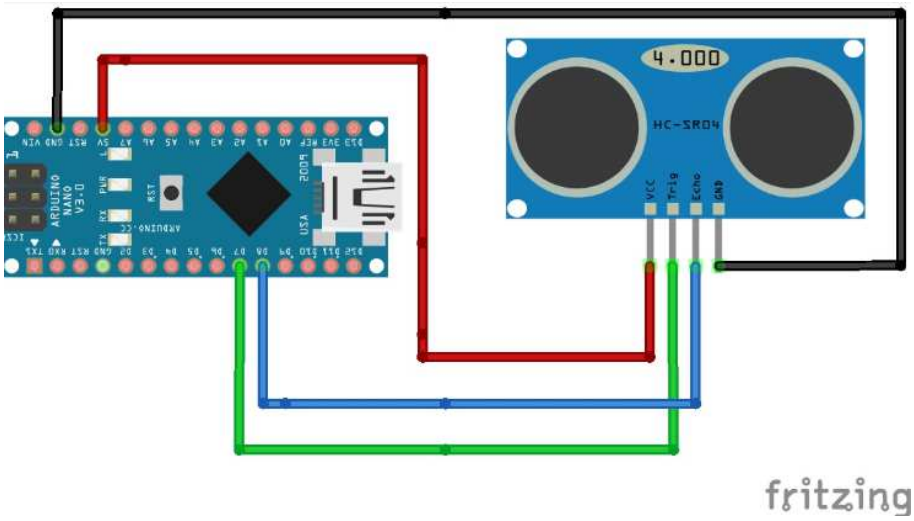


Figure IV. 4 Schéma du capteur ultrason avec la carte Arduino Nano.

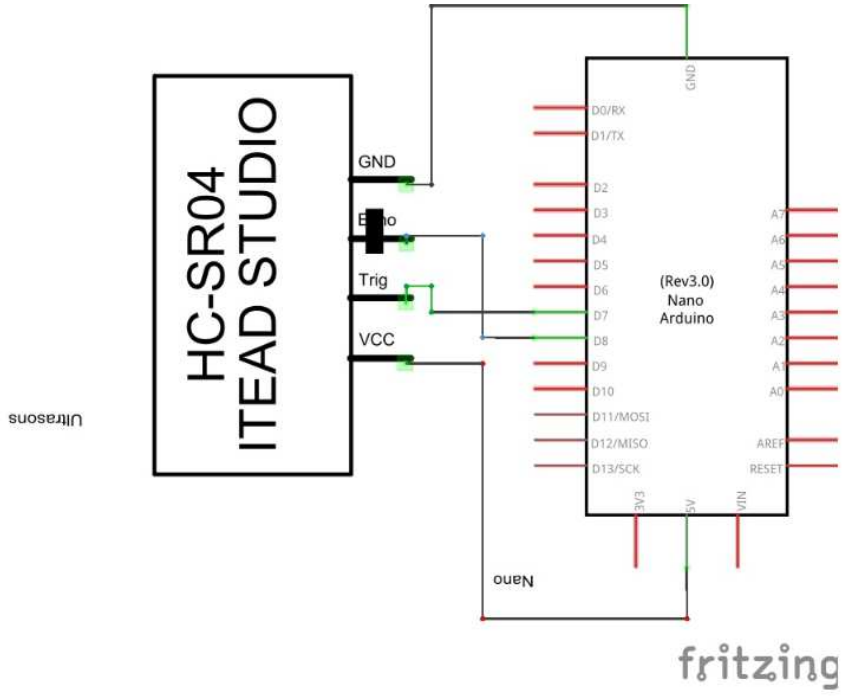


Figure IV. 5 Circuit électrique du capteur ultrason avec la carte Arduino Nano.

IV.4.2. Transmission radiofréquence

IV.4.2.1. Emetteur nRF24L01

Dans le cas d'échange d'information entre l'unité de communication (émetteur, récepteur) et prise de décision (Figure IV.6 à IV.7), le module émetteur-récepteur NRF24L01 est utilisé. Il utilise la bande 2.4GHz et peut fonctionner avec des vitesses de transmission allant de 250Kbps à 2Mbps. S'il est utilisé dans un espace ouvert et avec un débit inférieur, sa portée peut atteindre 100 mètres.

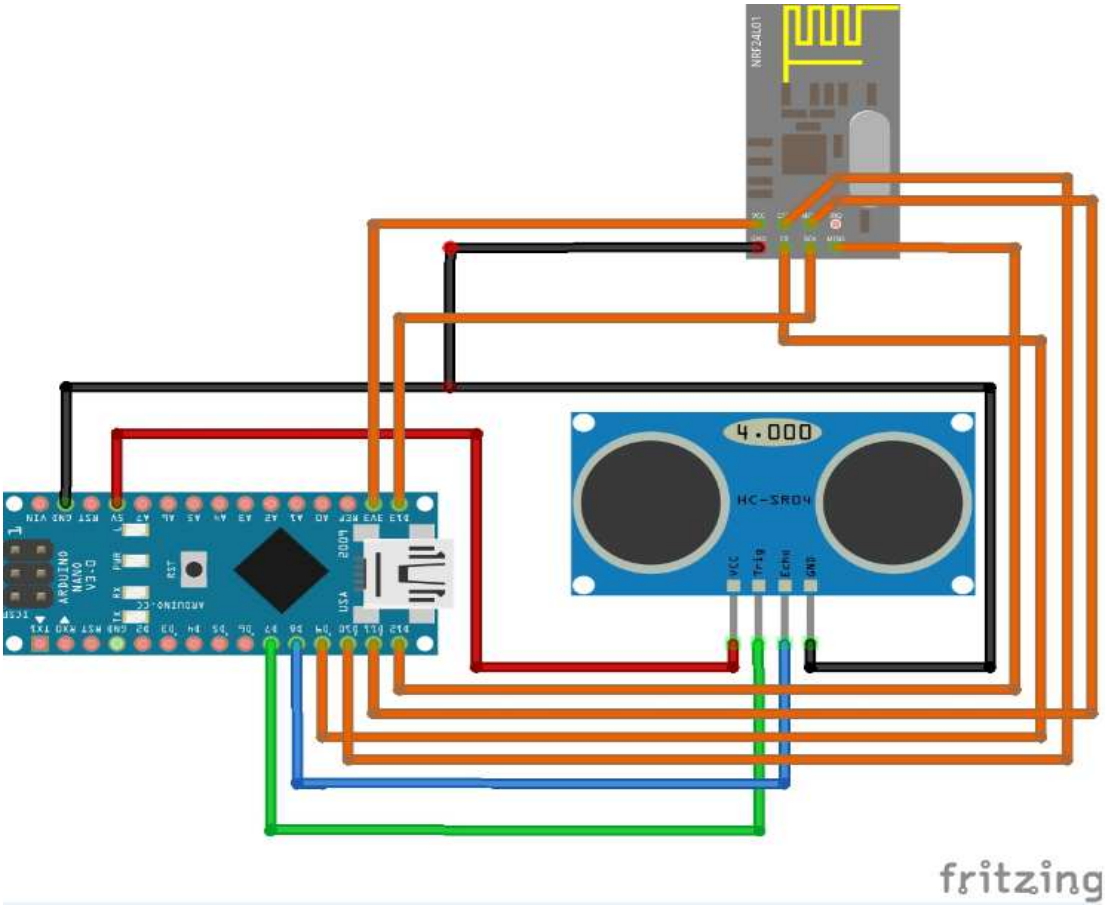


Figure IV. 6 Branchement du capteur ultrason et radiofréquence nRF24l01 avec la carte Arduino Nano.

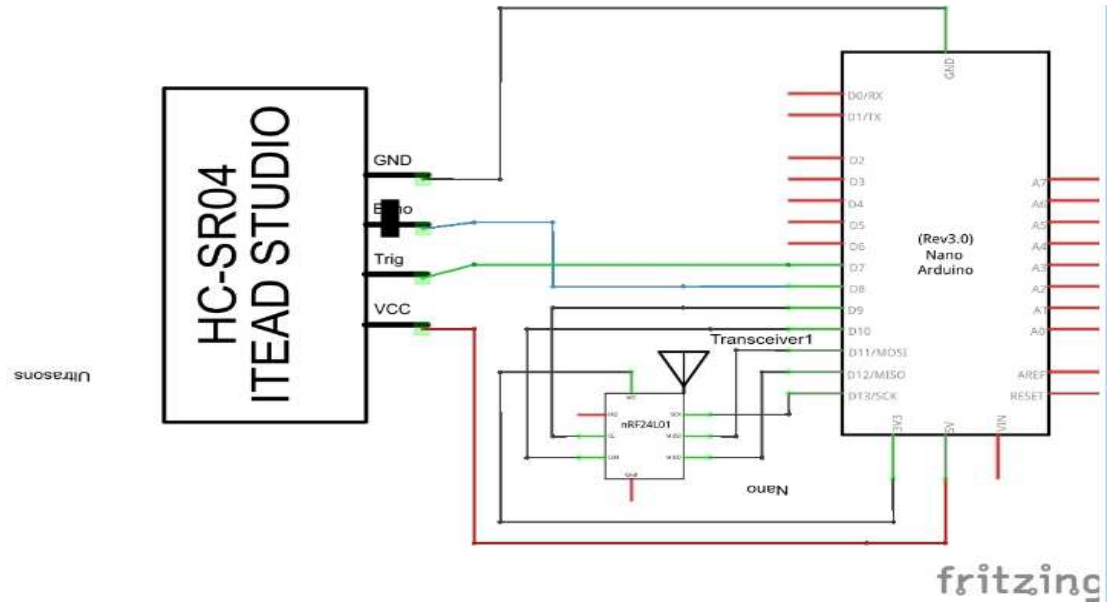


Figure IV. 7 Circuit électrique du capteur ultrason et radiofréquence nRF24l01 avec la carte Arduino Nano.

IV.4.2.2. Récepteur nRF24L01

Pour visualisés les informations reçues par l'émetteur nRF24L01 on utilise les composants : Arduino Mega et l'afficheur LCD 16×2, comme montré dans les figures IV.8 et IV.9.

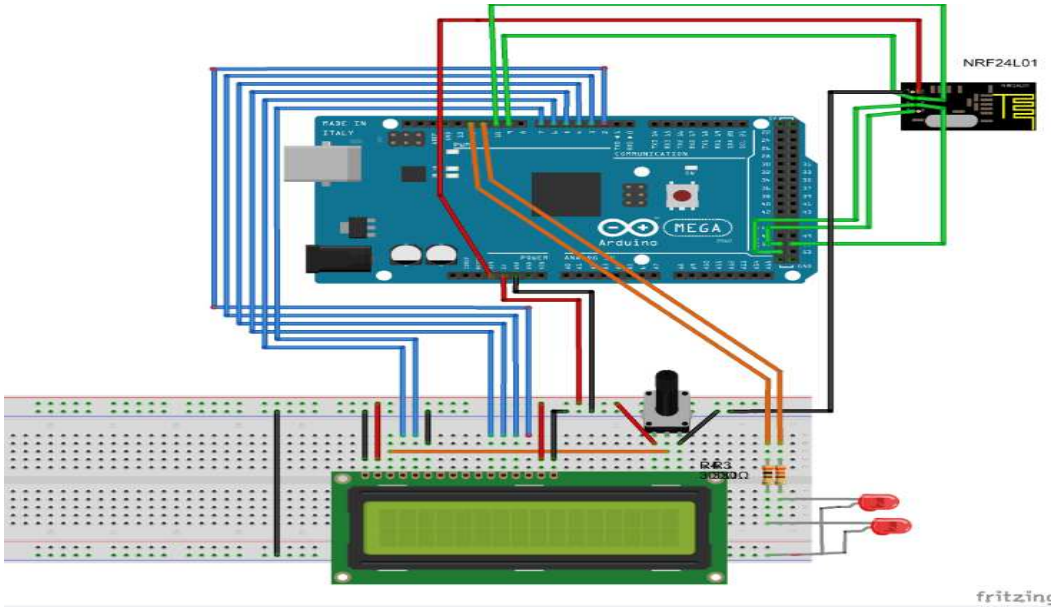


Figure IV. 8 Branchement de récepteur radiofréquence nRF24l01 avec la carte Arduino Nano et l'afficheur LCD 16×2.

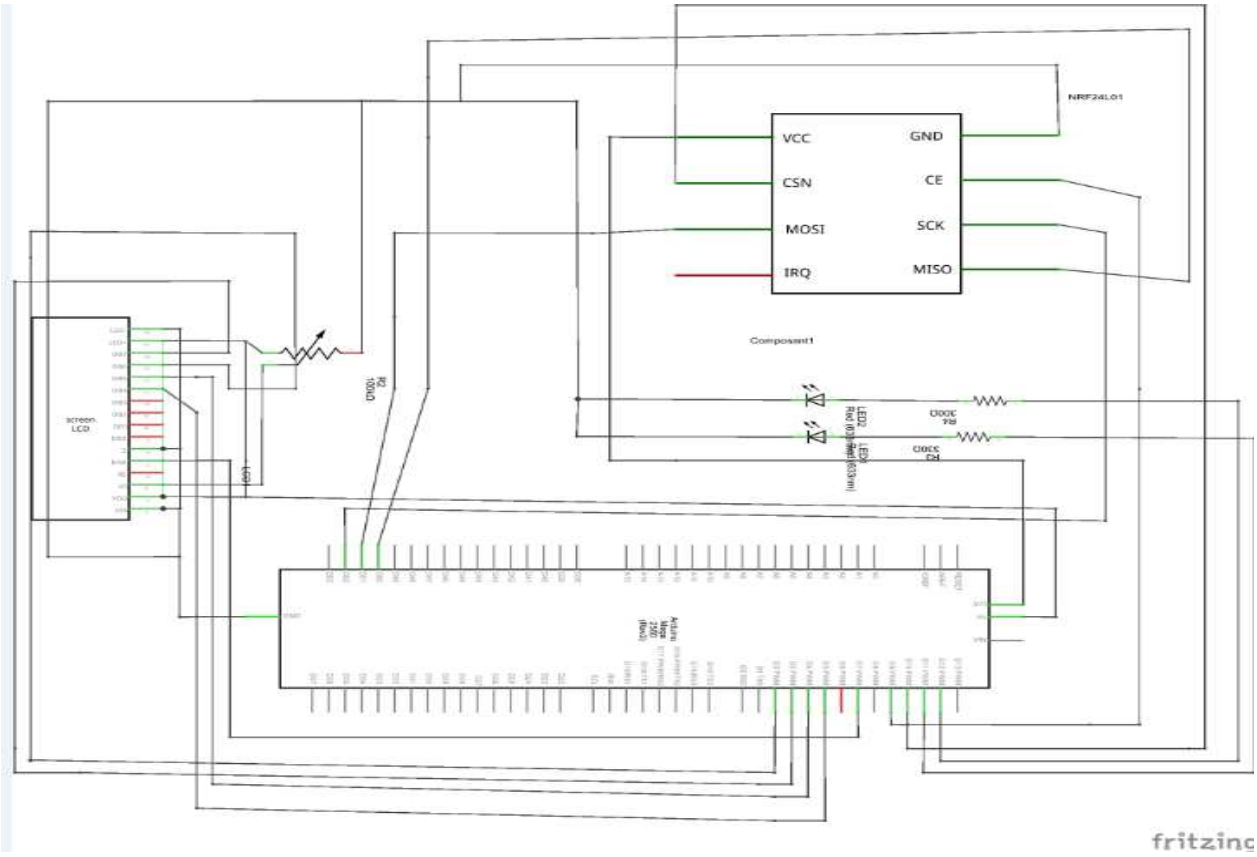


Figure IV. 9 Circuit électrique de récepteur radiofréquence nRF24l01 avec la carte Arduino Nano et l'afficheur LCD 16×2.

Nous avons placé deux Antennes nRF24L01, le premier est sur le coté arrière du robot (voir Figure IV.10) qui permet et du envoyer les données (Distance captées par l'ultrason), le deuxième est fixe dans un boitier avec un afficheur LCD pour afficher les données reçus par la première antenne, et deux leds d'indication de la présence ou l'absence des objets couleurs, pour comme représenté dans la Figure IV.11. Dans ce cas, la carte Arduino Mega est programmée dans le logiciel Arduino IDE.

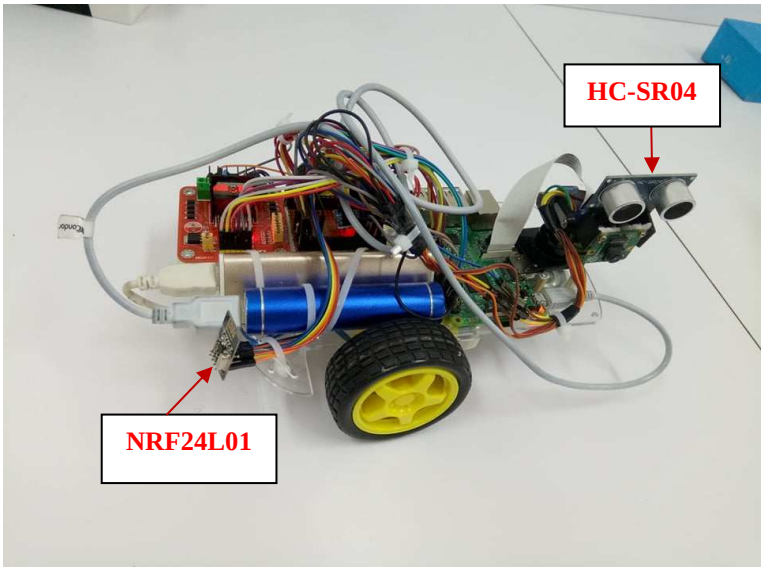
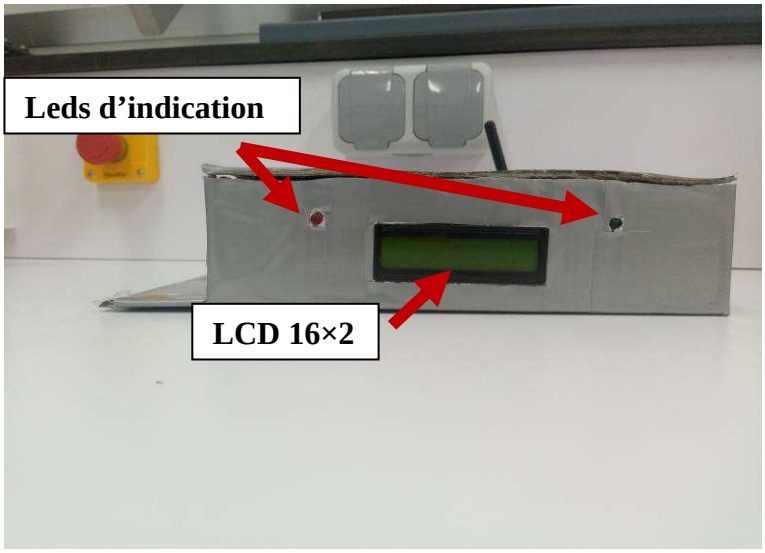
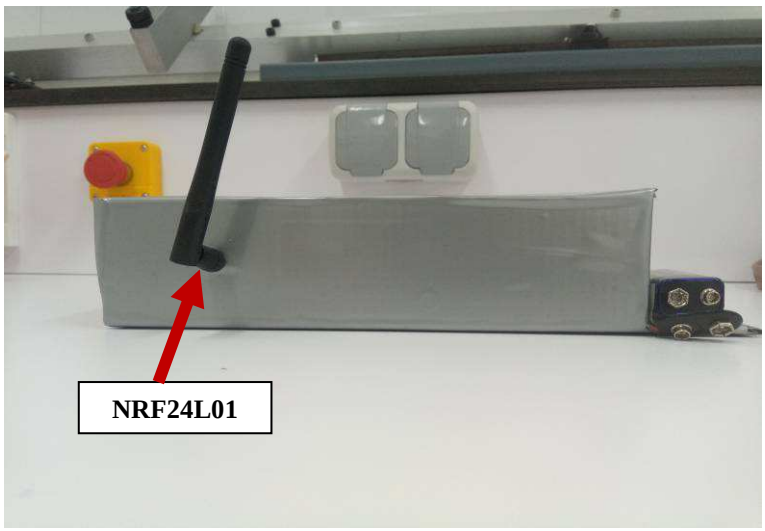


Figure IV. 10 Emetteur radiofréquence nRF24L01.



(a)



(b)

Figure IV. 11 Récepteur radiofréquence nRF24L01, (a) indication de présence ou absence de l'objet : Leds et afficheur LCD, (b) module nRF24L01.

IV.4.3. Raspberry *Pi* 3

IV.4.3.1. Connexion à distance

La technologie WiFi qui équipe aujourd'hui tous les nouveaux smart phones, et est utilisée principalement par les réseaux locaux sans fil. Le Wi-Fi permet l'accès à Internet avec des vitesses de transfert atteignant facilement des dizaines de méga bits par seconde.

Pour pouvoir se connecter à distance à notre Raspberry, il faut le connecter à un réseau local et faire appel au protocole SSH [35]. Le serveur SSH est activé par défaut sur le Raspberry *Pi*. Sous Linux, le client étant intégré dans la plupart des distributions. Par contre, sous Windows, nous devons installer «Putty» qui est un client SSH. Ensuite, il suffit d'entrer l'adresse IP du Raspberry dans Putty pour pouvoir se connecter.

IV.4.3.1.1. Réservation d'adresse IP

Un serveur DHCP distribue une adresse IP aux clients selon la disponibilité des adresses de la plage définie. Cependant, il est possible de faire en sorte qu'un hôte ait toujours la même adresse IP attribuée, c'est ce que l'on appelle de la réservation d'adresse. On doit donc indiquer dans le serveur DHCP (Dynamic Host Configuration Protocol) que telle adresse IP est réservée à telle adresse MAC.

Ou bien on alloue une adresse IP statique à notre Raspberry pour une liaison filaire par la configuration suivante :

-On ouvre le fichier de configuration des cartes réseaux:

sudo nano /etc/network/interfaces

-Puis indiquons à notre carte réseau que nous souhaitons passer en static:

```
auto eth0
iface eth0 inet static
address 192.168.1.111
netmask 255.255.255.0
gateway 192.168.1.1
```

IV.4.3.2. Système de Détection d'image avec Camera Raspberry

Dans le chapitre précédant, nous avons trouvé les meilleures combinaisons de l'espace couleur et les seuils pour la détection de chaque objet couleur utilisant un seuillage adaptative. Le programme Matlab s'occupe de détecter et reconnaître l'objet couleur (rouge, vert, bleu, jaune). Par la suite, on sélectionne le centre de ce dernier puis d'envoyer une notification à l'unité de contrôle. Enfin utiliser une simple commande des moteurs à courant continu pour suivre l'objet (commande avant, arrière, arrêt).

Et la résolution du Camera Raspberry sur Matlab est trouvée par la fonction :

```
wcam = webcam(mypi)
wcam =
  webcam with properties:
                    Name: '/dev/video0'
              Resolution: '320x240'
    AvailableResolutions: '320x240' '640x480'
```

Et le nombre des frames prends est varié sous cette fonction :

```
for ii = 1:10 (10 est le nombre des frames)
    img = snapshot(wcam)
    imagesc(img)
    drawnow
end
```

IV.4.3.3. Montage du Raspberry Pi 3 avec le contrôleur des moteurs (L298N)

Afin que le robot mobile soit capable de se déplacer et de suivre les objets, il est nécessaire d'utiliser les moteurs. Dans notre travail, nous avons utilisé un robot avec deux roues, et le types des moteurs est à courant continu. L'utilisation de deux moteurs a été causé par le coût n'est cher par rapport un servomoteur, il n'y a pas beaucoup de vibrations, faible consommation d'énergie et plus rapide par rapport un moteur pas à pas. Pour contrôler les moteurs, nous avons choisi le circuit L298N puisqu'il est disponible, facile à réaliser et ça coûte pas cher, d'une part et d'une autre part un seul circuit c'est suffisant pour commander

les deux moteurs. Maintenant le robot mobile est réagir selon les informations qu'il reçoit de la carte Raspberry Pi 3 à partir l'unité de traitement. Le schéma du circuit de commande est représenté dans la figure suivante :

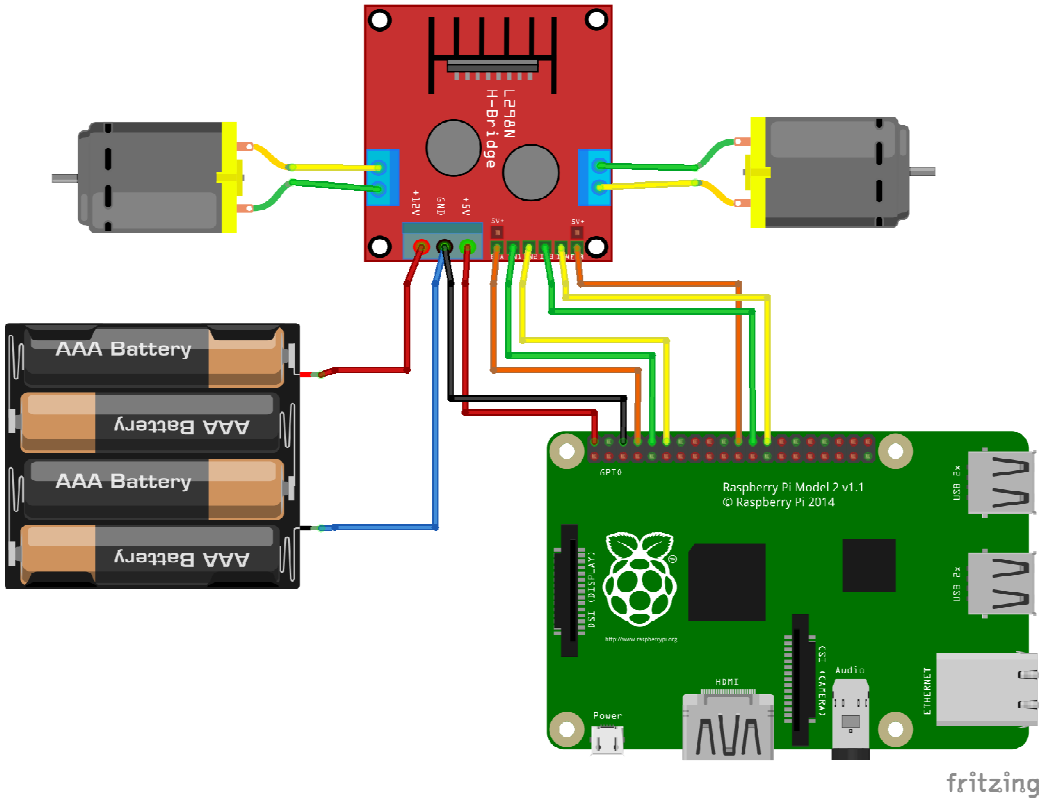


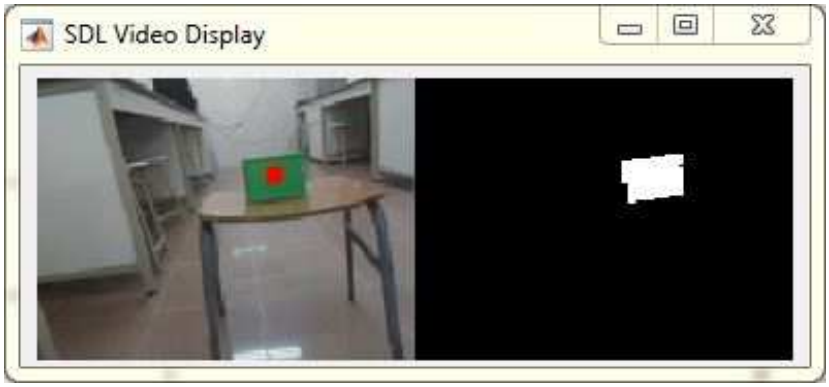
Figure IV. 12 Branchement du montage Raspberry Pi 3 avec le contrôleur L298N.

IV.4.3.4 Détection d'objet

Afin de démontrer la méthode décrite précédemment dans le chapitre III (détection de couleur), un modèle SIMULINK a été développé pour montrer chaque étape du processus.

Une image a été prise avec la RaspCam et utilisée pour démontrer les capacités de la caméra ainsi que le modèle développé. La figure IV.13 montre la détection des différents objets couleurs utilisé dans notre travail.

(a)



(b)

(c)

(d)

Figure IV. 13 Détection des objets couleurs : Rouge, Vert, Bleu, et Jaune.

IV. 5. L'organigramme du programme principal

Le programme global du système de détection et de suivi l'objet couleur est décrit dans l'organigramme suivant (Figure IV.14):

Le principe de fonctionnement du programme est le suivant:

- Assurer la communication sans fil (Wi-Fi) entre l'ordinateur et le robot mobile.
- Après on donne des commandes au robot par MATLAB-SIMULINK pour détecter une couleur précisée.

- Si le système fonctionne bien et détecte la couleur alors le robot doit suivre l'objet colorisé et faire plusieurs mouvement :
  - Si la distance D>30cm le robot conduit vert l'objet.
  - Si $15 < D < 30$ cm; le robot s'arrête.
  - Si D < 15cm le robot conduit le contraire de l'objet (en arrière).
- On répète les mêmes étapes précédentes pour chaque couleur.

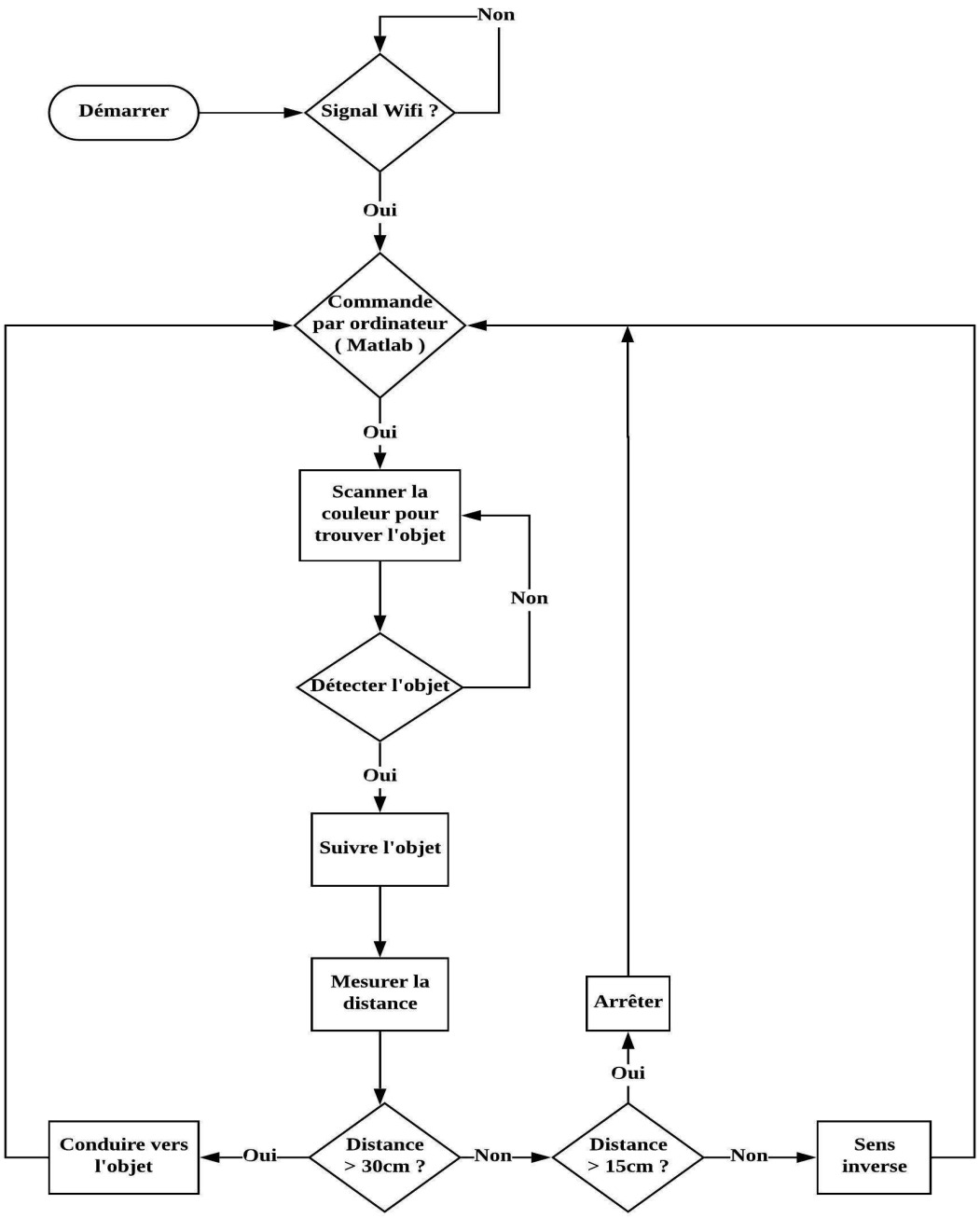
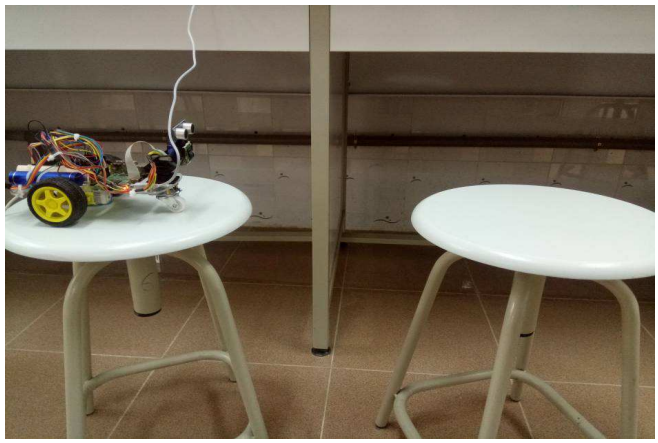


Figure IV. 14 Organigramme générale du système de détection d'objets implémenté sur les deux cartes de programmation Arduino et Raspberry *Pi*.

IV. 6. Teste du robot

Dans le chapitre III, nous avons présentée une approche basée sur seuillage adaptative pour la détection d'objets couleurs. Cette approche repose sur deux phases; une phase d'optimisation hors ligne qui permet de donner une classification de couleurs selon les couleurs des cibles; et une phase en ligne qui permet de détecter quatre objets couleurs différentes : rouge, vert, bleu, jaune.

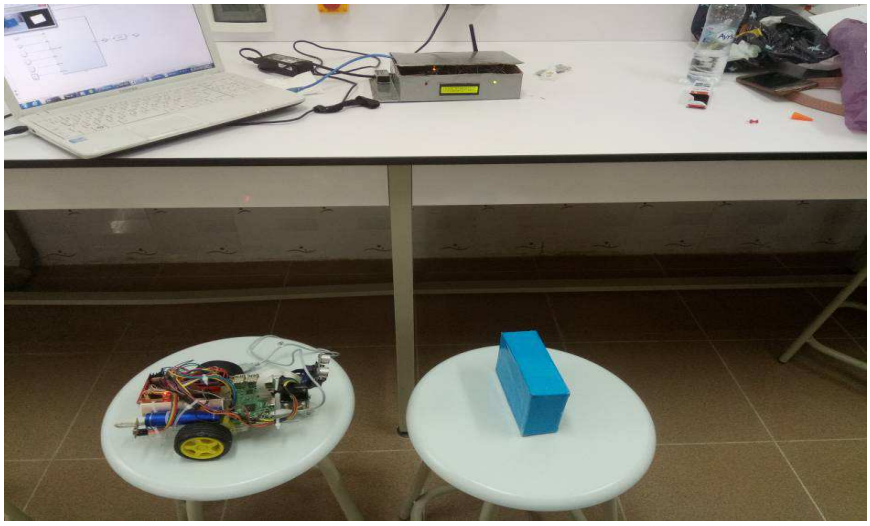
Les Figures suivantes montrent les étapes de traitement de l'image pour la détection d'objet couleur. La figure IV.15 montre dans le cas d'une absence d'un objet couleur, le suivi de l'objet n'a pas affecté par le robot mobile. Dans le cas de présence d'un objet couleur (exemple objet bleu) comme représenté dans la figure IV.16, notre système détecte l'objet par l'unité de traitement, ce dernier transmis des informations sur l'objet concernant la couleur R, V, B, J (suivi d'objets B c-à-dire le robot suivre l'objet couleur bleu, R : objet rouge, V : l'objet vert, J : l'objet jaune) et la distance par rapport le robot mobile par l'intermédiaire des antennes radiofréquences (unité de transmission).



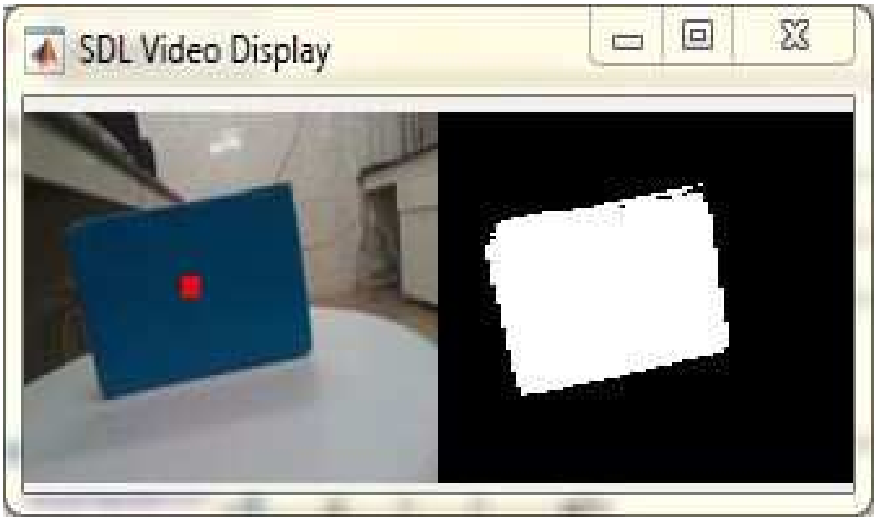
(a)

(b)

Figure IV. 15 Absence des objets couleurs.



(a)

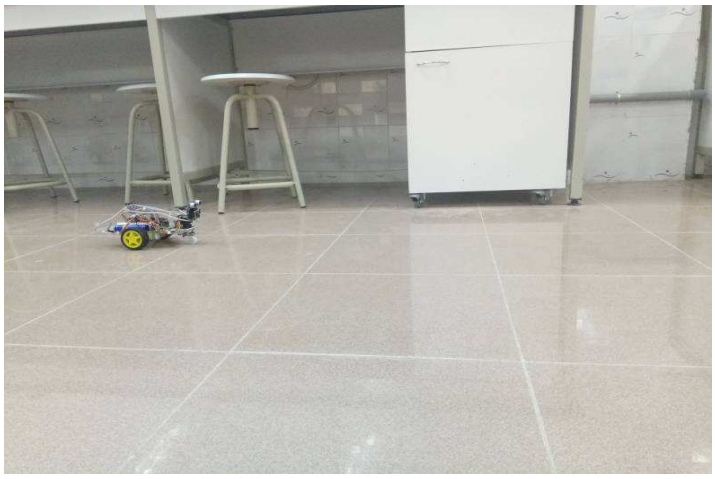


(b)

(c)

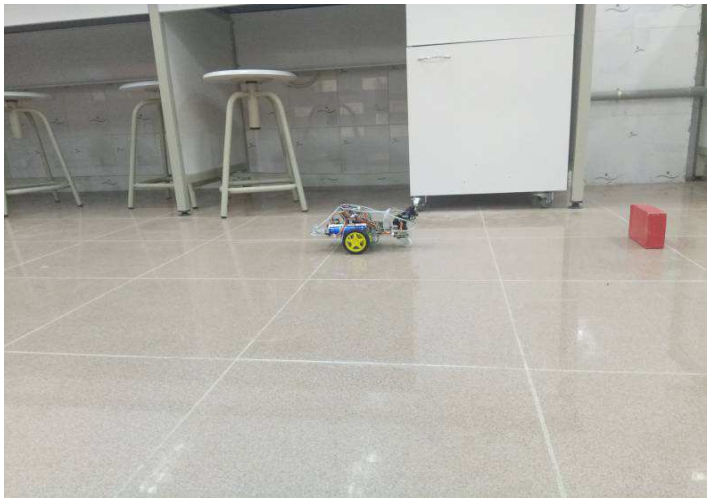
Figure IV. 16 Exemple d'une détection d'objet bleu, la distance entre le robot et l'objet est 27.98cm.

La figure suivante représente l'étape finale de notre test de suivi des différents objets couleurs, pour suivi l'objet une simple commande est appliquée sur les moteur à courant continu par le Raspberry *Pi* 3 : avant, arrière, arrêt, comme montré dans l'organigramme générale (Figure IV.14).

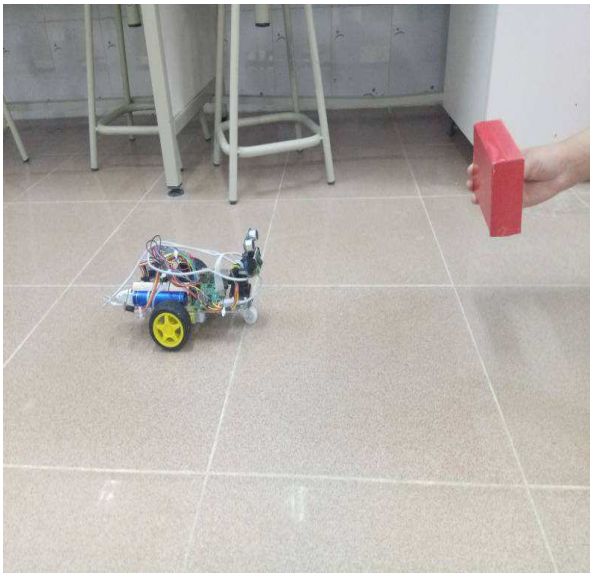


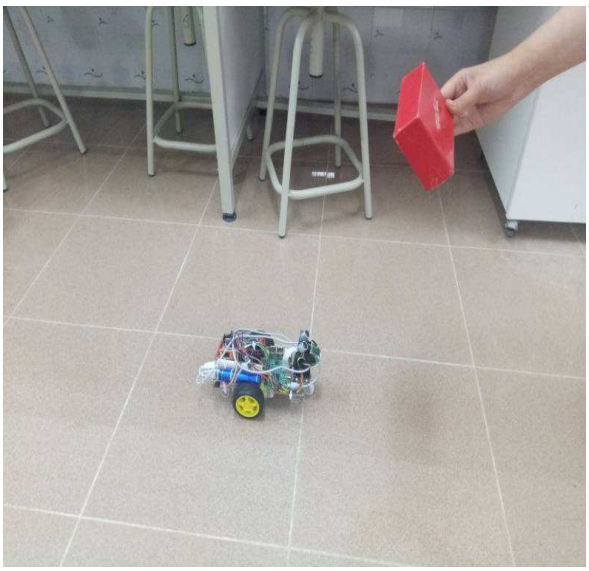
(a)

(b)



(c)





(d)

Figure IV. 17 Test du robot mobile, (a) aucun objet détecté, (b) Le démarrage du robot, (c) Le robot se dirige vers l'objet, (d) Orientation de la caméra vers l'objet par le servomoteur.

IV. 7. Conclusion

Dans ce chapitre nous avons présenté les différentes étapes qui nous ont permis de réaliser ce robot, ces composants utilisés et leurs tâches, on a présenté aussi l'efficacité et les performances de notre algorithme de détection d'objets couleurs implémenté sur les deux cartes Arduino et Raspberry *Pi*.

Conclusion générale

Conclusion générale

L'objectif de notre travail est la réalisation d'un suiveur d'objet couleur, on à utilisé plusieurs types des capteurs (ultrasons, capteur d'image Raspberry Cam), et deux type de communication sans fil, à savoir la communication radiofréquences et la communication Wi-Fi, ainsi deux type des cartes : Arduino et Raspberry *Pi* 3.

Pour mieux expliquer la détection d'objet, on a défini tout d'abord dans le premier chapitre le traitement d'image, ces techniques, changement d'espace couleur. Dans le deuxième chapitre, une représentation concernant le traitement d'image sur la carte Raspberry *Pi* 3 et une étude sur le choix de l'espace couleur les plus adapté à la détection des objets couleurs.

La couleur, de par son pouvoir discriminant, est un des attributs les plus utilisés en détection d'objets. Notre travail, se concentrent autour l'identification de l'espace couleur le plus discriminant. Nous avons montré que l'utilisation de l'espace RGB n'a y a pas l'espace couleur idéal pour la détection des objets. Du fait qu'il y a une forte corrélation entre les différents plans dans l'espace RGB, nous effectuons, dans une première étape, un changement de l'espace originale vers un autre espace en moins corrélé. Evidement, cette transformation permet une séparation de l'information relative, tels que HSV, HSI, YCbCr, YIQ. Ensuite, nous avons appliqué deux méthodes de seuillage fixe et adaptative. En général, les résultats obtenus montrent que par la combinaison d'un ensemble de stratégies de changement d'espace couleurs, il est possible de capturer les meilleures particularités pour atteindre un détecteur d'objet robuste. De cette façon, il était possible d'exploiter au mieux les particularités des différentes espaces couleurs pour améliorer la robustesse et éventuellement, la précision du processus de détection.

Pour suivre les objets de manière cohérente et pour éviter les échecs de détection à court terme causés par les occultations de l'objet ou par des changements brusques de luminosité, nous avons proposé une méthode basée sur deux phases : la phase de calibrage pour trouver l'espace couleur, le plan ou se trouve l'objet et les paramètres des seuils, cette dernière est une phase hors ligne, et une autre phase en temps réel qui permet de détecter quatre objets couleurs différentes : rouge, vert, bleu, et jaune, avec l'utilisation des paramètres trouvé dans la phase de calibrage.

Afin de validé notre travail, nous avons utilisé comme application un robot mobile avec un contrôle de mouvement plus simple (avant, arrête, arrière), notre robot exécute des commandes MATLAB-SIMULINK reçues par l'unité de traitement. Pour réaliser ce travail, on a passé par différentes étapes:

- ❖ On a utilisé un capteur ultrason HC-SR04 pour calculer la distance entre le robot et l'obstacle et de transmettre sans fil avec l'utilisation d'une radiofréquence nRF24L01. Le robot réalisé se déplace par 2 roues et une roue folle.
- ❖ L'ensemble de systèmes de perception et de déplacement est commandé par un microcontrôleur qui doit en utilisant les informations actuelles, décider l'action à prendre. Pour notre cas; on a utilisé le microcontrôleur célèbre ATmega328p, dont ses caractéristiques particulières nous ont aidé à faciliter les taches surtout en ce qui concerne sa programmation.
- ❖ Transmettre une vidéo en temps réel à l'ordinateur.

Ce travail nous a permis de traiter des problèmes d'ordre pratique et de vérifier des connaissances théoriques acquises toute le long de notre formation. Grâce au travail continu, on a peu atteindre notre but et satisfaire le cahier de charge, mais cela ne veut pas dire qu'il est complet, nous proposons que le travail réalisé soit la base de toute une série d'améliorations que nous n'avons pas eu la chance de les faire par manque de temps et de matériel. Le nombre d'améliorations que peuvent être ajoutes sont :

- ➢ Utilisé plus de capteurs ou un plateau de capteurs rotatif pour couvrir l'environnement pour choisi meilleur trajectoire.
- ➢ L'utilisation des moteurs pas à pas pour avoir une grande précision dans le mouvement du robot.
- ➢ Utilisation la radiofréquence en hautes fréquences pour assurer la connexion sans fil Wifi et diminuer les pertes des informations.
- ➢ L'étude approfondie de la phase de détection d'objets :
  - o Utilisation d'autre type de seuillage plus robuste.
  - o Les différentes étapes de seuillage fixe ou adaptative peuvent certainement être améliorées davantage, dans le but d'accélérer la vitesse d'exécution et/ou d'augmenter l'efficacité de notre méthode de détection, par exemple en effectuant l'optimisation des seuils (*THmin* et *THmax*), l'espace couleur et plan d'image. Potentiellement les algorithmes génétiques explorent tous l'espace des seuils en même temps, ce qui donne des meilleurs résultats de détection d'objets.
  - o Utilisation d'autre méthode de changement d'espace couleur.
- ➢ Etude approfondie sur le problème de généralisation, dans le cas si la scène ou l'objet change:

- Utilisation des techniques d'intelligence artificielle pour la classification des objets (réseau de neurones : perceptron multi-couche MLP ou la fonction radiale de base RBF, Support vector machine SVM).

- Dans ce travail, nous avons utilisé une simple stratégie de commande des moteurs à courant continu. Pour construire un robot plus performant, on utilise les techniques de commandes plus avancé tels que : filtre de Kalman linéaire ou non linéaire, commande neuro-floue.

Bibliographie

Bibliographie

[1] H. Ghazouani, Suivi d'objets basé forme et couleur pour la navigation robotique en temps réel, (2014).

[2] F. Hachemi, La détection et suivi des objets en mouvement dans une scène vidéo en utilisant la bibliothèque OpenCV, Master en Informatique, Université Abou Bakr Belkaid– Tlemcen Faculté des Sciences Département d'Informatique, (2016).

[3] B. Rémi, N. François, La compression JPEG, Rapport de Projet, Ecole nationale supérieure d'ingénieurs de Caen et centre de recherche (ENSICAEN), Université de Caen Basse-Normandie, (Année 2005-2006).

[4] F. Douak, Reconstruction des images compressées en utilisant les réseaux de neurones artificiels et la DCT, Thèse de magister en micro ondes pour télécommunications, Université de Batna, (2008).

[5] C. Bencheriet, A. Boualleg, H. Tebbikh, Segmentation de la Couleur de Peau par Seuillage Selon Différents Espaces de Couleur, JIG'2007 - 3èmes Journées Internationales sur l'Informatique Graphique, Université 8 Mai 45 de Guelma, (2007).

[6] M. Beladgham, F. Derraz, M. Khélif, D. Chouguer, Application de l'algorithme EZW pour la compression d'image médicale, Symposium International : Qualité et Maintenance au Service de l'Entreprise, Université Abou Bekr Belkaid –Tlemcen, Algérie, (2004).

[7] J.C. Russ, The image processing –Handbook-, Third Edition. CRC Press, CRC Press LLC, (1998).

[8] C.D. Bibhas, C. Bhabatosh, Color image compression based on block truncation coding using pattern fitting principle, ELSEVIER, 9 (2007) 2408-2417.

[9] T. Acharya, A.K. Ray, Image processing: principles and applications, John Wiley & Sons, (2005).

[10] B. Marir, M. M. Kalla, F. Douak, A. Daamouche, A Modular Support Vector Machine for Active Learning of Urban Remote Sensing Images Classification in Algeria, Journal of the Indian Society of Remote Sensing, 46 (2018) 515-529.

[11] F. Douak, R. Benzid, N. Benoudjit, Color image compression algorithm based on the DCT transform combined to an adaptive block scanning, AEU-International Journal of Electronics and Communications, 65 (2011) 16-26.

[12] A. Manzanera, TERI : Traitement et reconnaissance d'image, Cours Traitement et reconnaissance d'image, Master, Université Pierre et Marie CURIE, Paris 2006.

[13] R.S. Gonzalez, P. Wintz, Digital image processing, (1977).

[14] D. Zeroual, Implémentation d'un environnement parallèle pour la compression d'images al'aide des fractales, Thèse de magister en informatique, Université de Batna, 2006.

[15] Y. Hayat, Détection de contours et suivi d'objet dans une séquence d'images par les réseaux de neurones impulsionnels, Mémoire de Magister, Spécialité : Informatique, Universite mohamed boudiaf' des sciences et de la technologie d'oran, (2010).

[16] A. Cziho, Quantification vectorielle et compression d'image. Application à l'imagerie médicale, Thèse doctorat de l'université de Rennes 1, (5 mai 1999).

[17] A.B. Atitallah, Etude et Implantation d'Algorithmes de Compression D'images dans un Environnement Mixte Matériel et Logiciel, Thèse doctorat de l'université bordeaux 1, école doctorale des sciences physiques et de l'ingénieur, (11 Juillet 2007).

[18] D. Lamraoui, H. Slimani, Filtrage des images par différentes approches, Mémoire de Master, Option : Signal et Communications, Université M'hamed Bougara Boumerdes, (2017).

[19] J.C. Russ, The image processing –Handbook-, Third Edition, CRC Press LLC, 1998.

[20] A. Boucetta, Etude de l'effet des Transformées de Décorrélation en Compression des Images Couleurs RGB, Mémoire de Magister en informatique, option Ingénierie des systèmes informatique, Université de Batna 2, (2010).

[21] V. Risson, Application de la Morphologie Mathématique à l'Analyse des Conditions d'Éclairage des Images Couleur, Thèse de doctorat, Ecole des Mines de Paris, (17 Décembre 2001).

[22] A. Boucetta, K.E. Melkemi, DWT based-approach for color image compression using genetic algorithm, InInternational Conference on Image and Signal Processing. Springer, Berlin, Heidelberg, (2012) 476-484.

[23] C. Landry, Correction interactive de couleur par association, Mémoire M. Sc.: maître ès sciences en informatique, faculté des études supérieures, Université de Montréal, (Février 1999).

[24] A. Anwander, Segmentation d'images couleurs par opération gradient vectoriel multiéchelle et contour actif : application à la quantification des phases minéralogiques du clinker de ciment, Thèse de doctorat, Institut national sciences appliquées de Lyon, (Décembre 2001).

[25] W.B. Horng, J.W. Peng, C.Y. Chen, A new image-based real-time flame detection method using color analysis, In Proceedings. IEEE Networking, Sensing and Control, (2005) 100-105.

[26] S.K. Singh, D.S. Chauhan, M. Vatsa, R. Singh, A robust skin color based face detection algorithm, Tamkang Journal of Science and Engineering, 6 (2003) 227-234.

[27] B. Rémi, N. François, La compression JPEG, Rapport de Projet, Ecole nationale supérieure d'ingénieurs de Caen et centre de recherche (ENSICAEN), Université de Caen Basse-Normandie, (2005).

[28] C.D. Bibhas, C. Bhabatosh, Color image compression based on block truncation coding using pattern fitting principle, Pattern Recognition, 40 (2007) 2408-2417.

[29] K. Abdelbasset, Etude et réalisation d'une carte de contrôle par Arduino via le système Androïde, Mémoire Master, Université Ouargla, (2015).

[30] J. Boxall, Arduino workshop: a hands-on introduction with 65 projects, (2013).

[31] N. Nafa, B. Arabi, Conception et réalisation d'un système de sécurité commande à distance, mémoire de Master, UMBB,Boumerdes, (2016).

[32] B. Ahmed, M. Marjorie, Etude de la commande et simulation des circuits d'un pendule inversé, Mémoire de Master en Génie Electrique, Université M'hamed Bougara Boumerdes, (2017).

[33] F. Benaissa, R. Yahyaoui, Etude et réalisation d'une main robotique, Mémoire Master, Université Mouloud Mammeri de Tizi-ouzou, (2018).

[34] https://fr.mathworks.com/discovery/raspberry-pi-programming-matlab-simulink.html.

[35] A. Bouharaoua, B. Hacene, I. Mohammed, Automatisation d'une maison intelligente via une application Android, MASTER en Télécommunications, Université de Tlemcen, (2017).