République Algérienne Démocratique Et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Dr Moulay Tahar Saïda
Faculté des sciences et technologie
Département d'électronique

Mémoire de Fin d'Etudes

Pour l'obtention d'un diplôme de
Master En électronique
Option : instrumentation

Titre:

Commande gestuelle d'un véhicule électrique

Présenté par : **GOREINE Lakhdar**
GUENDOUZ Noureddine

Membres du jury :

Dr. DINE Khaled	**Président (Université de Saïda)**
Dr. MOSTEFAI Lotfi	**Examinateur (Université de Saïda)**
Dr. DAHANI Ameur	**Encadreur (Université de Saïda)**
Mr. BOUHAMEDI Merzoug	**Co-Encadreur (Université de Saïda)**

Année universitaire 2019/2020

REMERCIEMENTS

Avant tout nous remercions dieu qui nous a éclairé notre route et qui nous a donnés la fois et le courage de persister et de continuer en dépit de n'importe quel obstacle.

Nous tonnons à remercier nos encadreurs Dr. Dahani Ameur et Mr. Bouhamedi Merzoug pour leurs conseils et patience pour aboutir à ce travail.

Nous remercions tous les enseignants du département d'ELECTRONIQUE de l'université du Dr Moulay Tahar de Saïda

Nous remercions Monsieur Bouhamedi Merzoug pour sa Disponibilité et ses, conseils concernant la méthodologie du mémoire. Nous remercions également tous les membres du jury.

Dédicace

Je dédie ce travail à mes chers parents qui ont soif de succès et qui m'ont encouragé

Je dédie ce travail à tous mes frères et tous mes amis ***"geundouz noureddine et hichame khemsi, sehoul noureddine ;hachim amina et amin alioane"***

Je dédie ce travail à prof ***chichi noureddine*** *et son professeur* ***bezza ikram***

Goreine lakhdar

Dédicace

Je dédie ce modeste travail à:
Ma chère Mère,
Mon cher Père,
Mes chères Sœurs,
Mes chers frères,
Ceux qui m'aiment,
Ceux que j'aime

Geundouz noureddine

Résumé :

Ce projet consiste à contrôler les différents mouvements (marche avant et marche arrière, tourné à gauche et à droite, évitement d'obstacles...) à l'aide des gestes manuels. Un gant équipé d'accéléromètres transmet via le module wifi au microcontrôleur Arduino installé au niveau de la voiture les mouvements effectués par la main porteuse du gant**.**

لخص

تكون هذا المشروع من التحكم في الحركات المختلفة (إلى الأمام والخلف ، والانعطاف إلى اليسار واليمين ، وتجنب العقبات ...) باستخدام المثبت في Arduino إلى متحكم wifi قفاز مزود بمقاييس تسارع ينقل الحركات التي تقوم بها اليد التي تحمل القفاز عبر وحدة .الإيماءات اليدوية السيارة.

Abstract:

This project consists of controlling the different movements (forward and reverse, left and right turn, obstacle avoidance ...) using manual gestures. A glove equipped with accelerometers transmits the movements made by the hand carrying the glove via the wifi module to the Arduino microcontroller installed in the car.

Sommaire

Chapitre II : Fonctionnement et programmes

Chapitre III : Modules et cartes de la commande gestuelle

Chapitre IV : Tests et commande gestuelle

Liste des tableaux

Liste des figures

Introduction générale

Le développement de la science et de la technologie est vaste dans les différents domaines et disciplines, le monde de l'électronique et de la robotique font parties de notre monde.

La robotique est devenue une discipline extrêmement populaire. Alliant un grand intérêt pédagogique et industriel, elle demande beaucoup de créativité et de connaissances pluridisciplinaires. Un robot est constitué en général d'un système mécanique articulé et d'une partie électronique permettant sa commande. Il peut aussi intégrer une variété de capteurs, selon les tâches à réaliser. La programmation du système de commande exige des connaissances en informatique, automatique, en traitement temps-réel et même en des domaines avancés tels que la vision artificielle, l'intelligence artificielle, etc.

Le sujet abordé dans ce mémoire de fin d'étude concerne la réalisation d'une commande gestuelle d'un véhicule électrique basée sur une méthode de programmation contrôlée par le microcontrôleur arduino BLE_UNO.

Ce travail est divisé en quatre chapitres, dans le premier chapitre nous avons décrit les différents compartiments du véhicule. Le deuxième chapitre portait sur la programmation et le fonctionnement des différents composants du véhicule. Dans le troisième chapitre nous avons parlé des pièces du véhicule et de la reconnaissance contrôlée arduino ble_uno. Dans le dernier chapitre nous avons réalisé le contrôle du véhicule avec des dispositifs sans fil utilisant la technologie d'accès wifi.

Chapitre I :

Les différents compartiments du véhicule

I.1 Introduction

Les cartes Arduino sont conçues pour réaliser des prototypes et des maquettes de cartes électroniques pour l'informatique embarquée. Ces cartes permettent un accès simple et peu couteux à l'informatique embarquée. De plus, elles sont entièrement libres de droit, autant sur l'aspect du code source (Open Source) que sur l'aspect matériel (Open Hardware). Ainsi, il est possible de refaire sa propre carte Arduino dans le but de l'améliorer ou d'enlever des fonctionnalités inutiles au projet.

Le langage Arduino se distingue des langages utilisés dans l'industrie de l'informatique embarquée de par sa simplicité. En effet, beaucoup de librairies et de fonctionnalités de base occultent certains aspects de la programmation de logiciel embarquée afin de gagner en simplicité. Cela en fait un langage parfait pour réaliser des prototypes ou des petites applications dans le cadre de hobby.

Le micro-controlleur Arduino permet de :

- ✓ Contrôler les appareils domestiques.
- ✓ Fabriquer votre propre robot.
- ✓ Faire un jeu de lumières.
- ✓ Communiquer avec l'ordinateur.
- ✓ Télécommander un appareil mobile (modélisme) Etc.…

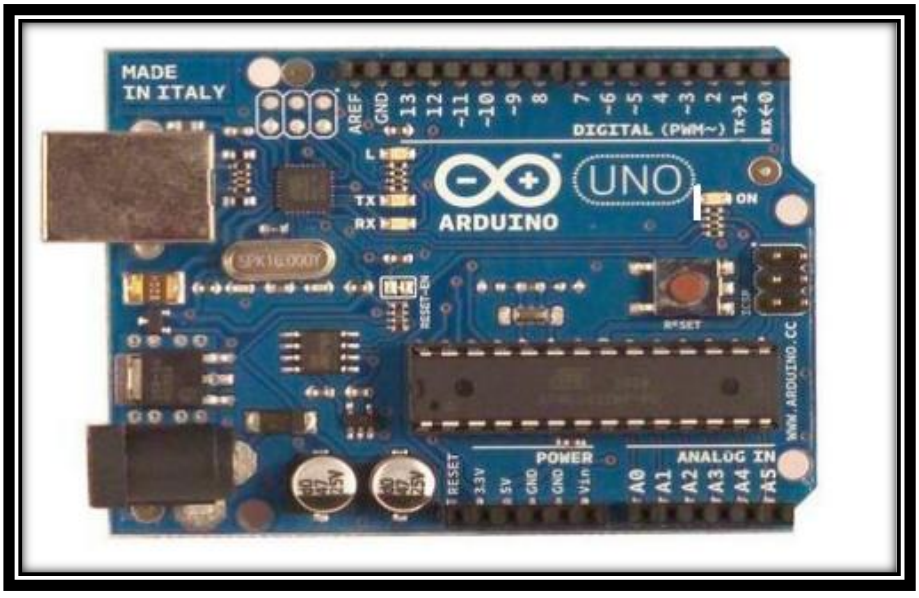


Figure I.1 : La carte Arduino UNO

I.2. Le Module Arduino

I.2.1. Présentation Générale

Arduino est une gamme de circuits électroniques open source basés pour la plupart sur un microcontrôleur du fabricant Atmel. Ces circuits intègrent les composants nécessaires pour permettre une utilisation rapide et simple du microcontrôleur. Cette simplification vise à rendre accessibles à tous la création et la programmation d'objets ou dispositifs interactifs.

Ces objets peuvent contenir toutes sortes de capteurs, d'indicateurs lumineux ou d'interrupteurs que l'on souhaite faire intervenir. Entre autres, les cartes Arduino sont équipées de connecteurs standardisés pour brancher des modules compatibles appelés shields. Ces derniers sont des circuits d'une taille plus ou moins semblable à celle de l'Arduino et qui viennent s'empiler sur ces connecteurs. Ils proposent des extensions matérielles qui permettent d'ajouter des fonctionnalités originales à son projet. En plus de ces connecteurs, les cartes possèdent toutes une connexion USB permettant de programmer facilement le microcontrôleur qu'elles embarquent.

I.2.2. Description Technique

Comme énoncé précédemment, il existe beaucoup de cartes Arduino différentes, mais elles possèdent toutes des éléments en commun.

I.2.2.1. Alimentation

Le microcontrôleur présent en général sur les cartes Arduino est alimenté par une tension de 5V.

En fonction du modèle de la carte, cette tension peut être fournie soit par une des prises d'alimentation présentes sur la carte, soit par la prise USB utilisée pour la connecter à un ordinateur.

La valeur de la tension à fournir sue une des prises d'alimentation doit être comprise entre 7 et 12V mais cette tension n'a pas besoin d'être stabilisée en raison de la présence d'un régulateur de tension sur la carte. Raccorder la branche positive de son alimentation à cette broche, comme on le ferait pour une pile, et la branche négative sur une broche GND.

I.2.2.2. Horloge

L'horloge détermine la fréquence ou la rapidité à laquelle les instructions seront exécutées par le microcontrôleur. Cette vitesse peut varier d'un microcontrôleur à l'autre et est exprimée en hertz(Hz).

Si les PC et Mac peuvent atteindre des fréquences de plusieurs gigahertz, la fréquence est en revanche beaucoup plus faible pour les microcontrôleurs, avec des fréquences de quelques mégahertz seulement. Cette fréquence est déterminée par un oscillateur à quartz ou résonateur céramique.

I.2.2.3. Reset

La reset est une fonction physique permettant au microcontrôleur, comme son nom l'indique, de réinitialiser son état.

Un microcontrôleur exécute en effet les instructions contenues dans sa mémoire de manière cyclique et infinie. Il n'est ainsi pas rare, notamment lors de la conception de son circuit et infinie. Il n'est ainsi pas rare, notamment lors de la conception de son circuit, que le programme soit soumis à une erreur bloquant le bon déroulement du programme.

I.2.2.4. Entrées/sorties

Les entrées et sortie sont les moyens que possède le microcontrôleur pour communiquer avec le monde extérieur. Typiquement, on appelle entrées / sorties les pattes métalliques qui donnent cette forme familière aux puces électroniques. Ces pattes reçoivent ou émettent des signaux logiques qui peuvent alors être interprètes par le microcontrôleur ou d'autres circuits. On peut ranger ces broches par catégorie suivant leur fonctionnalité :

- Les broches digitales : ces broches fournissent des données digitales sous forme de signaux logiques. Elles ne peuvent donc contenir que deux valeurs, un 0 logique correspondant à une absence de tension et une logique correspondant à une tension de 5V.

- Les broches analogiques : ces broches ne fonctionnent qu'en entrée, c'est-à-dire qu'elles ne servent qu'à la lecture de données. Elles acceptant des tensions comprises entre 0 et 5V. Ces tensions sont ensuite utilisées par un convertisseur analogique/numérique qui s'occupe de traduire ces valeurs physiques en données numériques sur 10 bits, soit des données comprises entre 0 et 1023.

- Les broches d'alimentation : ces broches servent à alimenter des sheilds ou circuits externes. Elles permettent également d'alimenter la carte Arduino sans passer par la prise jack présente sur certains modèles.

- Les broches de communication : il existe trois ports de communication sur la carte Arduino : le port série, le port I^2C et le port SPI. Ces ports ne disposent pas de broches dédiées à leur fonctionnement. Ils partagent en revanche leur fonctionnalité avec d'autres broches.

I.2.2.5. Mémoire

Les microcontrôleurs ATmega dont sont équipées la plupart des cartes Arduino disposent de trois types de mémoires : la mémoire flash, la SRAM (staticRandom Access Memory ou mémoire vive statique) et le EEPROM (Electrically-Erasable Programmable Read- Only Memory ou mémoire morte effaçable électriquement et reprogrammable).

I.2.2.6. Le matériel

Il s'agit d'une carte électronique basée autour d'un microcontrôleur Atmega du fabricant Atmel, dont le prix est relativement bas pour l'étendue possible des applications.

Figure I.2 : Carte Arduino.

I.2.3. Le logiciel

Le logiciel permet de programmer la carte Arduino, Il offre une multitude de Fonctionnalités. Le langage Arduino est inspiré de plusieurs langages. Le langage impose une structure particulière typique de l'informatique embarquée.

Le programme est lu par le microcontrôleur de haut vers le bas. Une variable doit être déclarée avant d'être utilisée par une fonction. La structure minimale est constituée :

• En tête : déclaration des variables, des constantes, indication de l'utilisation de Bibliothèques

•Un setup (= initialisation) cette partie n'est lue qu'une seule fois, elle comprend les fonctions devant être réalisées au démarrage (utilisation des broches en entrées ou en sortie, mise en marche du midi, du port série de l'I2C etc...)

•Un loup (boucle) : cette partie est lue en boucle ! C'est ici que les fonctions sont réalisées. En plus de cette structure minimale, on peut ajouter :

• Des « sous-programmes » ou « routines » qui peuvent être appelées à tout moment dans la boucle, très pratiqué pour réaliser des morceaux de codes répétitifs.

• Des « callbacks », ce sont des fonctions qui sont rappelées automatiquement depuis une bibliothèque.

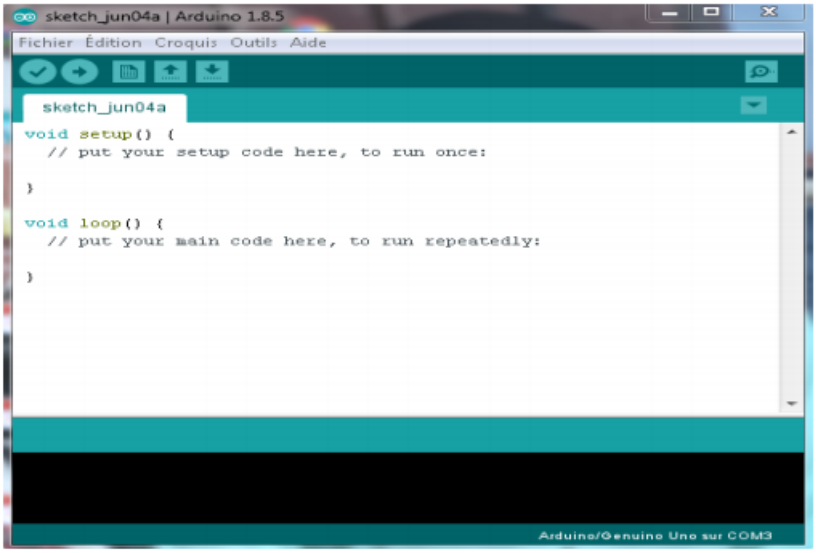


Figure I.3 : Photo d'un logiciel Arduino

I.2.4. L'interface

L'interface du logiciel Arduino se présente de la façon suivante :

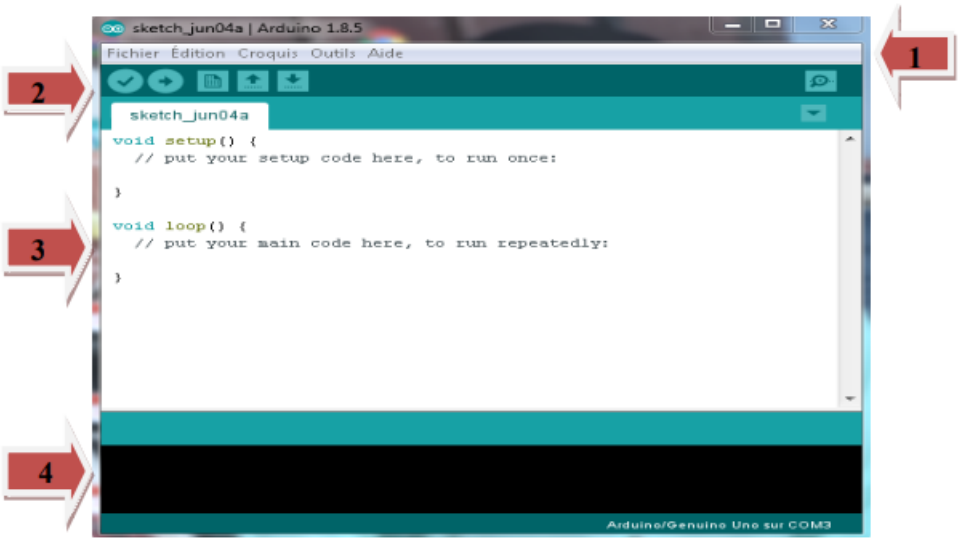


Figure I.4 : Interface de logiciel Arduino

1. Options de configuration du logiciel
2. Boutons pout la programmation des cartes

3. Programme à créer
4. Débogueur (affichage des erreurs de programmation) Les boutons :

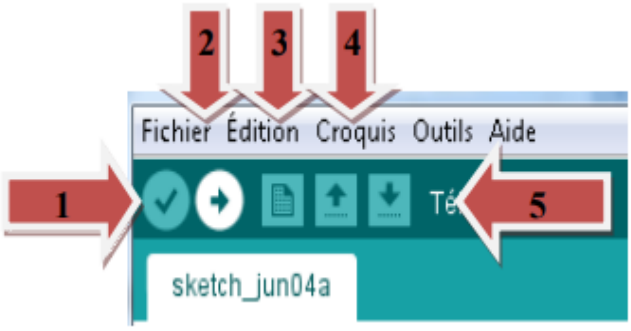


Figure I.5 : Les boutons de logiciel Arduino.

1. Permet de vérifier le programme, il actionne un module qui cherche les erreurs dans le Programme
2. Compiler et envoyer le programme vers la carte
3. Créer un nouveau fichier
4. Charger un programme existant
5. Sauvegarder le programme en cours.

I.2.5. Arduino BLE-UNO

Figure I.6 : Arduino BLE-UNO

Le BLE-UNO comme carte de commande principale, qui dispose de 14 entrées/sorties numériques broches (dont 6 peuvent être utilisées comme sortie PWM), 6 entrées analogiques et un résonateur en céramique 16 MHz, 1 USB connexion, 1 prise de courant, 1 tête ICSP et 1 bouton de réinitialisation.

Il contient tout ce qui prend en charge le microcontrôleur. Il vous suffit de le connecter à un ordinateur via un câble USB ou de démarrer avec un adaptateur AC-DC ou batterie.

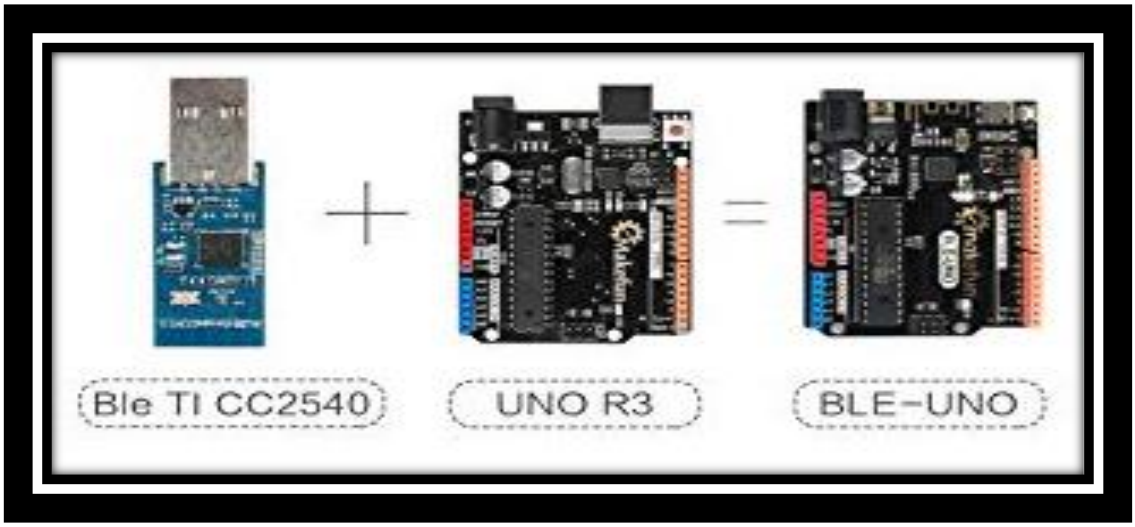


Figure I.7 : BLE-UNO : Composition

Spécifications techniques:

- Tension de travail: 5V
- Puce BLE: TI CC2540
- Canal de travail BLE: 2,4 G
- Distance de transmission Bluetooth: 50 m de distance ouverte
- Interface: Micro-USB
- Tension d'entrée: alimentation USB ou entrée 7V ~ 12V DC externe
- Tension de sortie: sortie 5 V CC et sortie 3,3 V CC et entrée d'alimentation externe
- Microprocesseur: ATmega328 (la fiche technique de la puce est dans la documentation)
- chargeur de démarrage: Arduino Uno
- Fréquence d'horloge: 16 MHz

- Prise en charge du protocole d'interface USB et de l'alimentation (sans alimentation externe)
- Prise en charge de la fonction de téléchargement de l'ISP
- Port d'E / S numérique: 14 (4 ports de sortie PWM)
- Port d'entrée analogique: 6
- Port d'E / S de courant CC: 40mA
- Port de courant continu 3,3 V: 50 mA
- Mémoire flash: 32 Ko (ATmega328) (0,5 Ko pour le chargeur de démarrage)
- SRAM: 2 Ko (ATmega328)
- EEPROM: 1 Ko (ATmega328)
- Taille: 75x55x15mm.

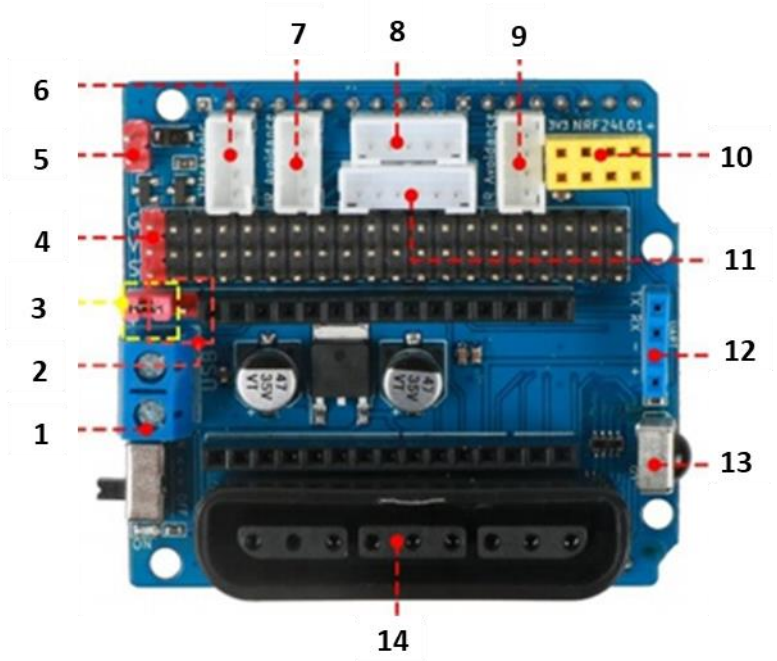


Figure I.8 : Child

Les légendes :

1. Interface d'affichage de tension
2. Broche de sélection de commande de moteur à DC
8. Interface de suivi infrarouge à trois voies
9. Interface d'évitement d'obstacles infrarouge droit et de recherché de lumière

3. Broche de sélection de signal ultrasonique

4. Interface servo

5. Interface d'extinction d'incendie pour petit moteur de ventilateur

6. Interface de module à ultrasons RGB

7. Interface d'évitement d'obstacles infrarouge gauche et de recherche de lumière

10. Interface+ NRF24L01

11. Interface d'entraînement de moteur L298N

12. Interface UART (module WIFI / module Bluetooth)

13. Récepteur de télécommande infrarouge

14. Interface PS2

I.2.6. Assemblage du véhicule

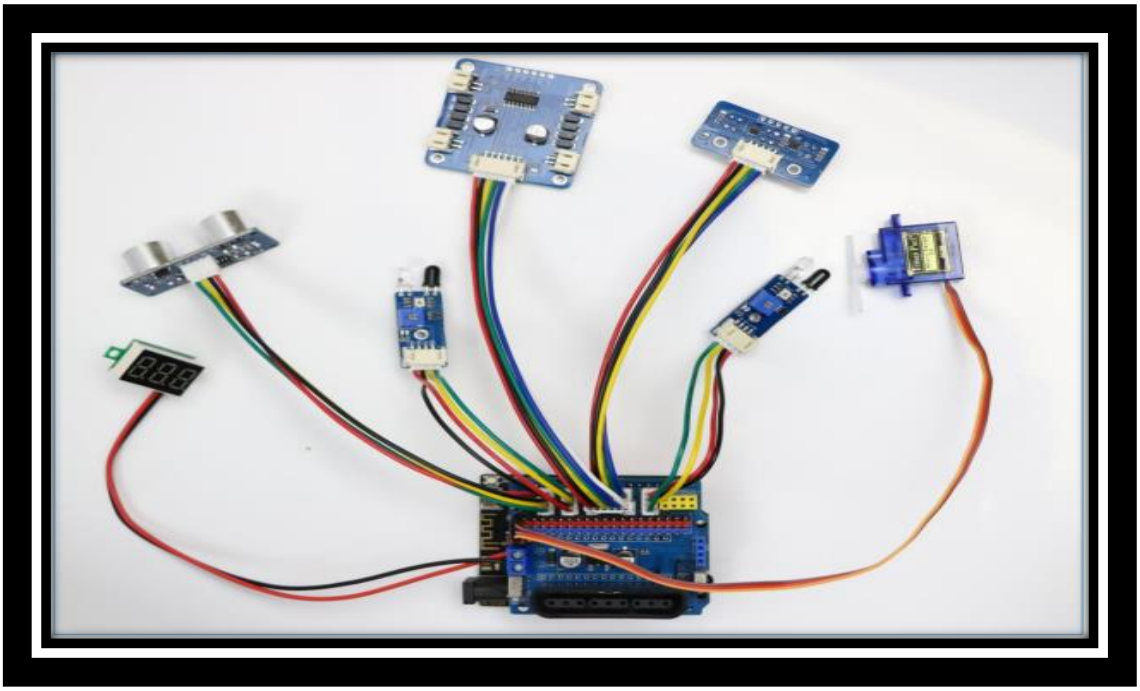


Figure I.9 : Schéma de principe de la connexion globale

I.2.7. Principe d'entraînement du moteur

Dans cet étude, nous avons choisi quatre motoréducteurs à courant continu comme puissance, nous avons choisi MX1616L comme puce de commande du moteur MX1616L utilise une conception de structure de circuit en pont H, la technologie de tube de puissance de haute fiabilité est particulièrement adaptée pour la bobine d'entraînement du moteur du circuit MOSFET de puissance à intégration interne du circuit de charge inductive, la plage de tension de travail couvre

2-10 V, 27 C, VDD = 6,5 V, deux canaux en même temps, conditions de travail ,Le courant de sortie continu maximum à 1 canal est de 1,2 A, le courant de sortie de crête maximum est de 2,5 A ;le courant de sortie continu maximal de 2 canaux atteint 1,2 A et le courant de sortie de crête maximal atteint 2,5 A.

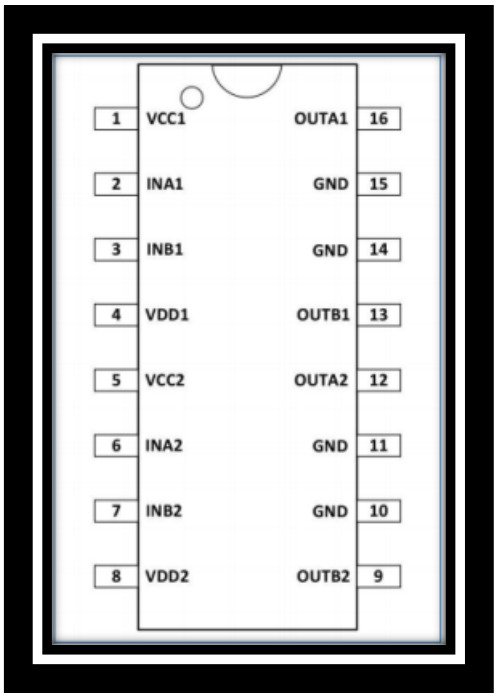


Figure I.10 : Disposition des broches à puce

Le MX1616L peut entraîner quatre moteurs. 2, 3, 6 et 7 sont connectés au niveau de contrôle d'entrée entre OUTA1 OUTB1 et OUTA2 OUB2 respectivement. La rotation positive et négative et la rotation d'arrêt du moteur sont contrôlées par les caractéristiques suivantes:

- Tube d'interrupteur d'alimentation MOSFET à faible résistance interne
- Diode de continuité intégrée en interne
- Petit courant d'entrée
- Contrôlez deux moteurs à courant continu
- Peut contrôler un moteur pas à pas séparément
- Peut réaliser une rotation positive et négative

Tableau I.1 : Tableau des fonctions logiques, X représente 1 ou 2.

Inx	INx	OUTx	OUTx	Une fonction
L	L	Z	Z	Etre prêt
H	L	H	L	Avant
L	H	L	H	Inversion
H	H	L	L	Stop

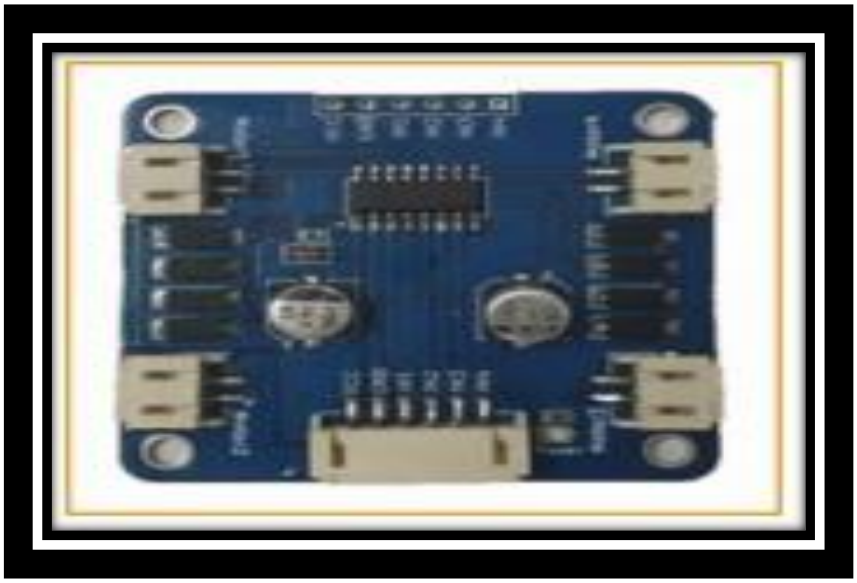


Figure I.11 : Carte physique du module

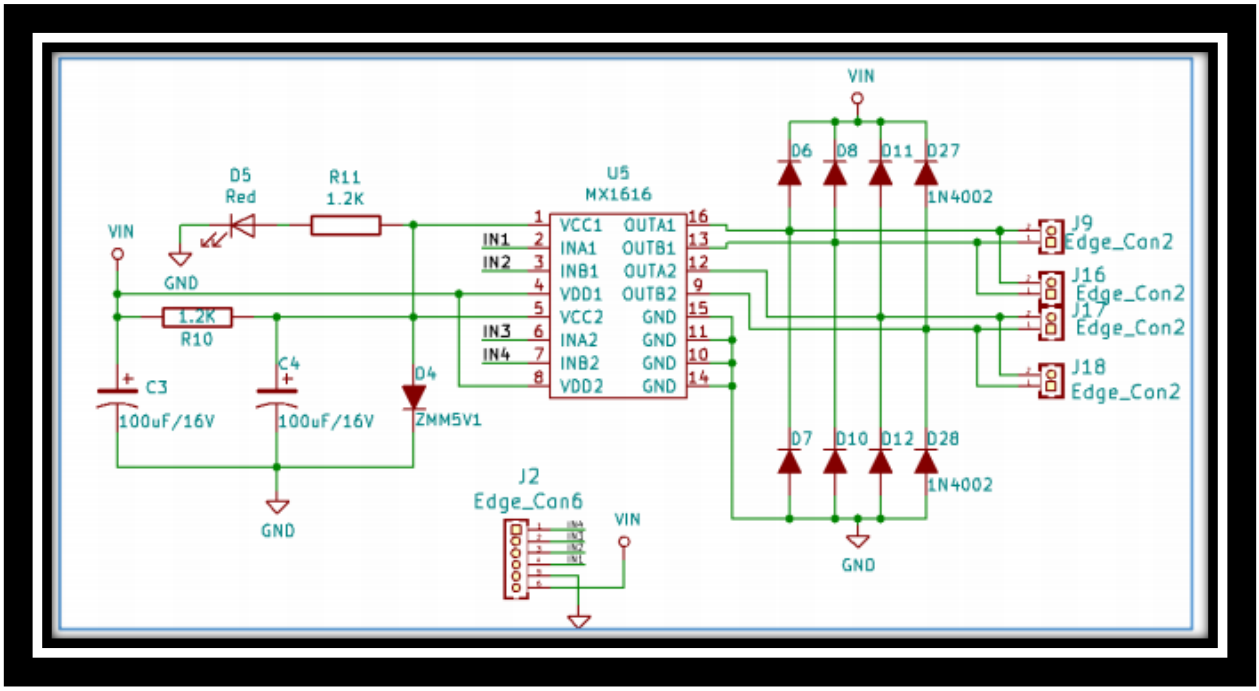


Figure I.12 : Diagramme schématique de l'entraînement du moteur

I.2.8. Connexion du fil de la carte de commande du moteur

En se référant aux données de la puce, nous saurons que l'Arduino UNO a 6 broches PWM, à savoir numérique interfaces 3, 5, 6, 9, 10, 11, et nous sélectionnons 5, 6, 9, 10 comme contrôle moteur IO, la connexion est indiquée dans Tableau I.2

La méthode de câblage de la carte d'extension L298N et Arduino est la suivante:

Tableau I.2 : La méthode de câblage de la carte d'extension L298N et Arduino

MX1616L	BLE- UNO
VCC	VIN
GND	GND
IN1	6
IN2	10
IN3	5
IN4	9

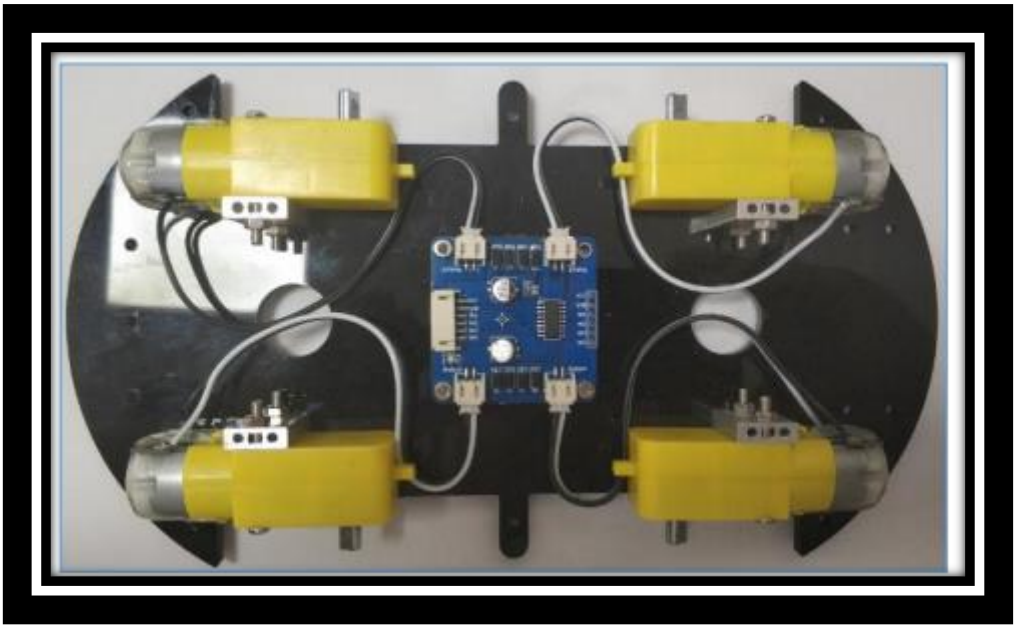


Figure I.15 : Schéma de connexion de la carte pilote MX1616L et de la carte d'extension Arduino

Dessin réel de la connexion entre la carte d'entraînement du moteur et la carte d'extension Arduino

I.3. Principe de l'évitement des obstacles infrarouges

Le module infrarouge d'évitement d'obstacles et de recherche de lumière à une paire d'émission et de réception infrarouge tube et le tube émetteur émet des rayons infrarouges d'une certaine fréquence. Lorsque la direction de détection rencontre un obstacle (surface réfléchissante), la lumière infrarouge est réfléchie et reçue par le récepteur tube, et passe à travers le comparateur. Une fois le circuit traité, le voyant vert sera allumé, et l'interface de sortie de signal produira un signal numérique (un signal de bas niveau).

La plage efficace de détection est de 2 à 30 cm et la tension de travail est de 3,3 V à 5 V.

La distance de détection du capteur peut être ajustée par le potentiomètre. En raison de l'utilisation des rayons infrarouges, la capacité anti-interférence est très forte et la précision de mesure est élevée lorsque la distance est modérée. De plus, le module est facile à assembler et facile à utiliser, et peut être largement utilisé dans de nombreuses situations telles que l'évitement d'obstacles de robot, obstacle chariot d'évitement, comptage de pipelines et suivi de lignes en noir et blanc.

I.3.1. Principe de recherche de lumière

La fonction de recherche optique du module d'évitement d'obstacles infrarouge et de recherche consiste à émettre l'analogique signal à travers la photodiode pour juger de l'intensité lumineuse de l'environnement environnant. Quand le la photorésistance est éclairée par une forte lumière, sa valeur de résistance chute rapidement et le courant qui passe augmente. La résistance de la photorésistance augmente rapidement dans l'environnement sombre, et le courant qui passe diminue et le panneau de commande principal détermine s'il existe une source de lumière. Tension de travail: 0,3-10V.

Étant donné que la photodiode SG-PT3528 est utilisée, elle peut simuler la sensibilisation de l'œil humain, la sensibilité maximale la longueur d'onde est de 590 nm, il peut résister aux interférences infrarouges, et la vitesse de réponse est rapide et les performances est stable. De plus, le module est facile à assembler et à utiliser et peut être largement utilisé dans les veilleuses, la pelouse lampes, lampes solaires et similaires.

I.3.2. Évitement des obstacles infrarouges et paramètres du module de recherche

- Tension de travail: 5V
- Plage de distance efficace de détection infrarouge: 2 ~ 30 cm
- Longueur d'onde de sensibilité maximale: 590 nm
- Signal d'évitement d'obstacles infrarouge de sortie: signal numérique
- Signal lumineux de sortie: signal analogique

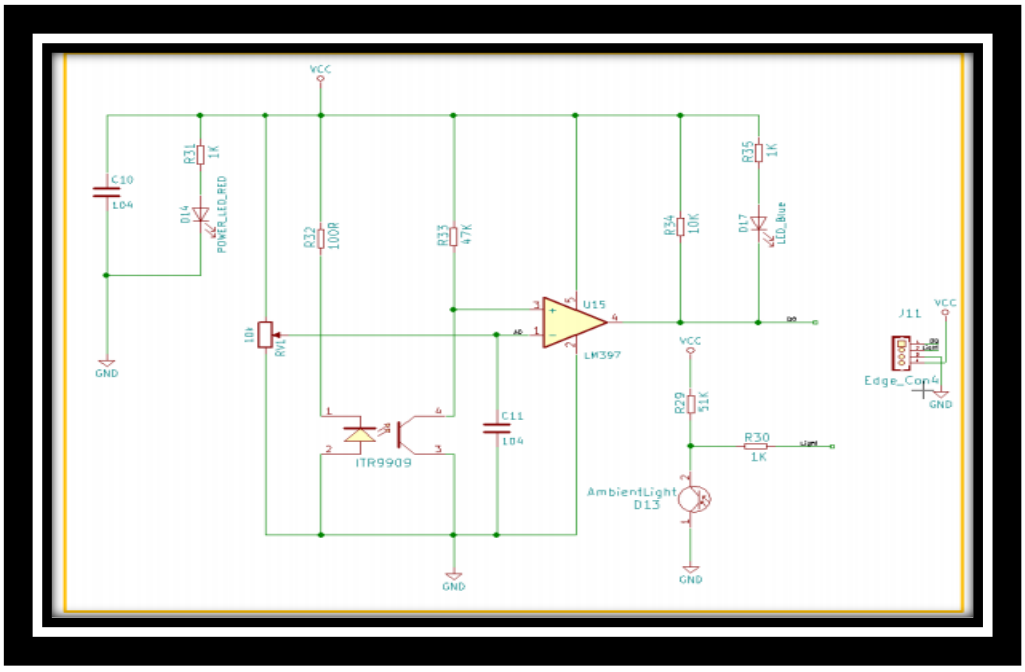


Figure I.16 : Diagramme schématique du module de détection et d'évitement d'obstacles infrarouge

Remarque : ce module peut ajuster la distance de détection d'évitement d'obstacle infrarouge en potentiomètre. La distance de détection est de 2 ~ 30 cm. Si vous trouvez que la mesure de distance n'est pas très sensible pendant l'utilisation, vous pouvez utiliser le potentiomètre du trimmer pour obtenir le résultat souhaité potentiomètre, la distance de détection est augmentée; le potentiomètre antihoraire est utilisé pour réduire la distance de détection), comme le montre la figure.

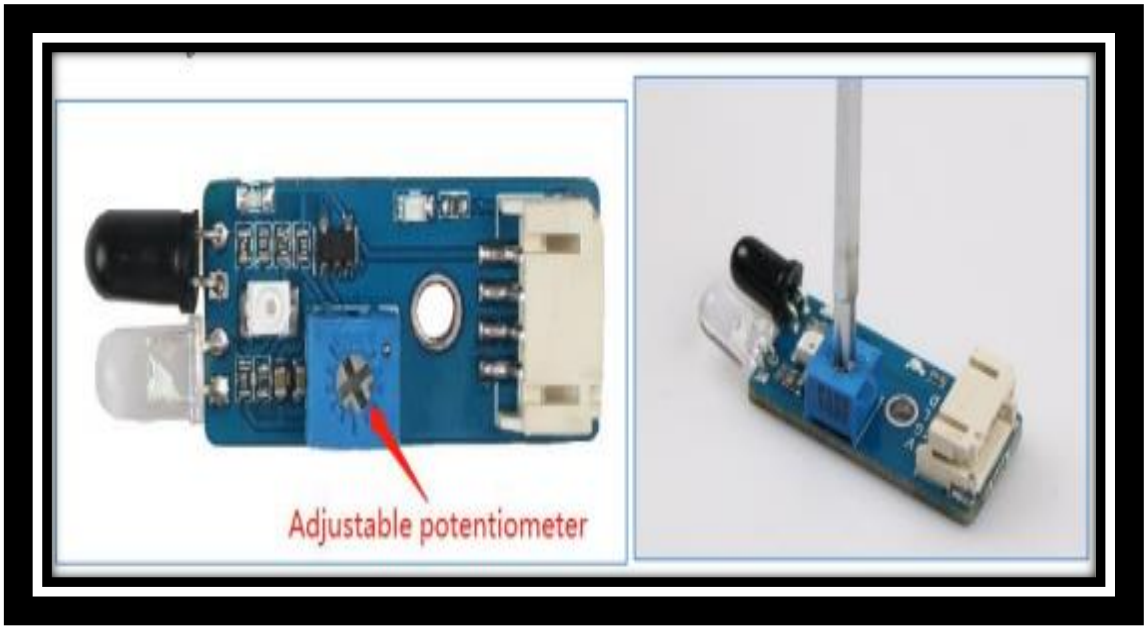


Figure I.17 : Diagramme schématique du réglage de la distance de détection

I.3.3. Évitement des obstacles infrarouges et connexion des câbles du module de recherche de lumière

Tableau I.3 : Évitement des obstacles infrarouges et connexion des câbles du module de recherche de lumière

Obstacle infrarouge gauche	Arduino UNO	Obstacle infrarouge droit	Arduino UNO
module d'évitement		module d'évitement	
V	VCC	V	VCC
G	GND	G	GND
L	A3	L	A4
D	12	D	A5

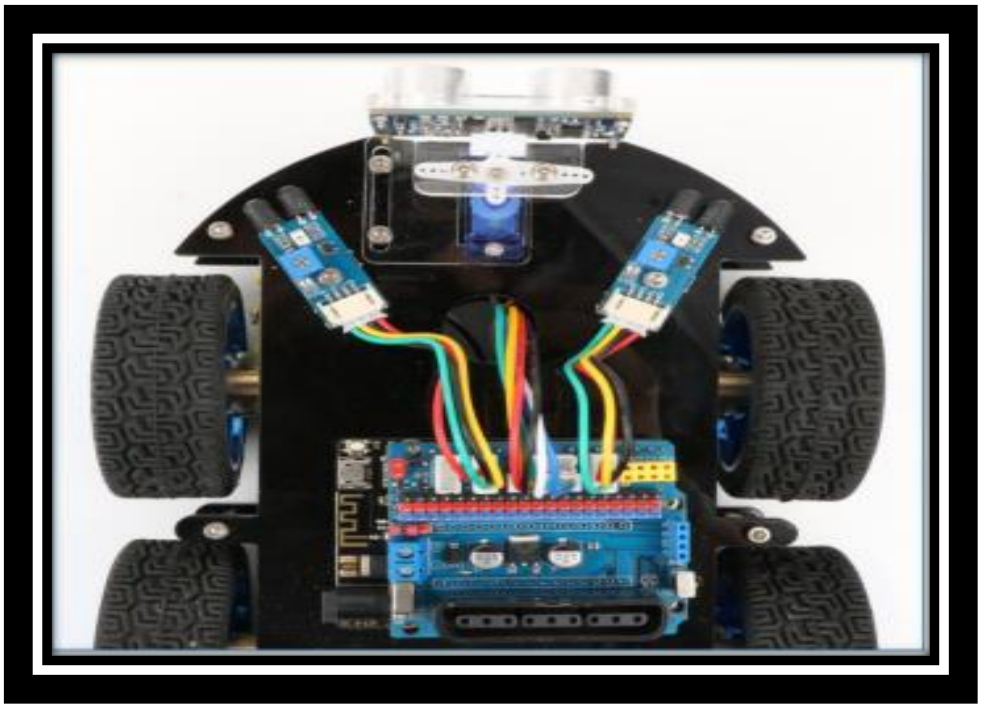


Figure I.18 : Diagramme schématique du fil du module d'évitement d'obstacles infrarouge et de recherche de rayons lien

I.4. Servomoteur

I.4.1. Introduction de l'appareil à gouverner

L'appareil à gouverner est également appelé servomoteur. Il a d'abord été utilisé pour réaliser sa fonction de direction sur le navire. Parce qu'il peut contrôler en continu son angle de rotation à travers le programme, il est largement utilisé pour réaliser le direction et les différents mouvements articulaires du robot. L'appareil à gouverner est le contrôle de la direction du chariot.

Le mécanisme présente les caractéristiques suivantes: petite taille, couple élevé, conception mécanique externe simple et une grande stabilité. Que ce soit dans la conception matérielle ou logicielle, la conception de l'appareil à gouverner est un partie du contrôle de la voiture. D'une manière générale, l'appareil à gouverner est principalement composé des éléments suivants:

Les composants sont composés du volant, du réducteur, du potentiomètre de retour de position, du moteur à courant continu, circuit de commande, etc., comme illustré à la Figure I.21 La Figure I.22 montre la carte physique des plus courantes utilisées l'appareil à gouverner 9G à ce stade.

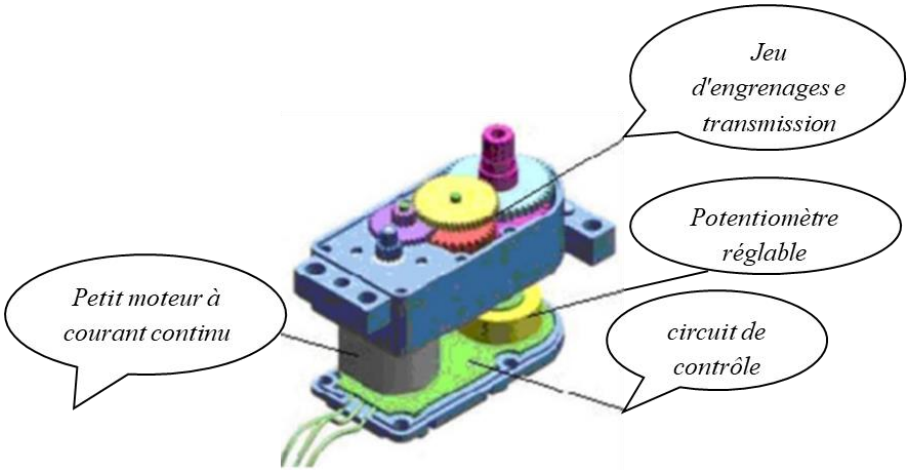


Figure I.21 : Schéma de principe de l'appareil à gouverner

I.4.2. Paramètres de l'appareil à gouverner SG90

Il existe trois lignes d'entrée pour l'appareil à gouverner SG90, Le milieu du rouge est la ligne électrique, et d'un côté est la ligne de masse. Cette ligne offre la garantie énergétique le plus élémentaire appareil à gouverner, principalement la rotation du moteur. Consommation, l'orange est la ligne de signal. L'alimentation a deux spécifications, l'une est de 4,8 V, l'autre de 6,0 V, ce qui correspond à différentes normes de couple, c'est-à-dire le couple de sortie est différent et le 6,0 V correspondant est plus grand, selon les conditions d'application.

I.5. Expérience de test du module d'évitement d'obstacles à ultrasons RGB

I.5.1. Introduction au module d'évitement d'obstacles ultrasonique RGB

Le module d'obstacles à ultrasons RGB intègre des lumières LED RGB et des modules de mesure à ultrasons, et Les lumières LED RGB peuvent être utilisées dans une variété de couleurs.

Le module d'évitement d'obstacles à ultrasons est un appareil qui convertit d'autres formes d'énergie en ultrasons énergie d'une fréquence souhaitée ou d'autres formes d'énergie qui convertit l'énergie ultrasonore en la même la fréquence. Actuellement, il existe deux types de capteurs à ultrasons couramment utilisés, à savoir le type électroacoustique et type dynamique fluide.

Les types électroacoustiques comprennent principalement:

Les capteurs piézoélectriques, capteurs magnétostrictifs et capteurs électrostatiques.

Le type hydrodynamique comprend à la fois un sifflet à gaz et à liquide.

La plupart des capteurs à ultrasons à ce stade utilisent des capteurs piézoélectriques pour fonctionner. Sa portée est un point chaud pour l'échographie. Dans ce projet, nous utilisons le module à ultrasons pour mesurer la distance entre le véhicule et l'obstacle.

Le module peut fournir une fonction de détection de distance sans contact de 2 cm à 400 cm, la précision de la plage peut atteindre 3 mm;

La compensation corrige le résultat de télémétrie et utilise la méthode de communication GPIO avec le microcontrôleur.

Le module comprend un émetteur à ultrasons, un récepteur et un circuit de commande. Comme la mesure de distance et la direction de voitures intelligentes, ou certains projets, est souvent utilisé. La mesure de distance de voiture intelligente peut trouver les obstacles devant vous dans le temps, de sorte que le la voiture intelligente peut tourner dans le temps et éviter les obstacles. La figure 4.8.1 est la carte physique des ultrasons RGB module d'évitement d'obstacles.

I.5.2 Paramètres du module d'évitement d'obstacles à ultrasons

- Tension de travail: 4,5 V ~ 5,5 V. En particulier, il ne doit absolument pas dépasser 5,5V.
- Courant de consommation électrique: minimum 1mA, maximum 65mA
- Fréquence Fréquence de résonance: 40 KHz
- Plage de détection: 2 cm à 4 m. Erreur: 0,1 ± 1%
- (En particulier, la détection de la distance la plus proche est de 4 mm,
- La distance maximale est de 4 mètres, les données sont sortie en continu, aucun réglage n'est nécessaire.)
- Plage de températures de fonctionnement: -40 ° C à + 100 ° C
- Dimensions: 46 mm * 21 mm * 20 mm (H)
- Taille du trou fixe 2 * Φ2mm Pas: 42x16mm.

I.5.3. Introduction aux lumières LED RGB

Le WS2812B est un circuit dédié de commande de commande de LED à trois canaux. La puce contient une numérique intelligente interface, signal de verrouillage de données, mise en forme et amplification du circuit de commande.

Il comprend également des oscillateurs et pilote de sortie à courant constant programmable haute tension 15V.

Dans le même temps, afin de réduire l'ondulation de l'alimentation, les trois canaux ont une certaine fonction de conduction de retard, qui peut réduire l'installation d'ondulation du circuit plus facilement pendant le rafraîchissement du cadre. Contrairement au RVB traditionnel, le WS2812B intègre la puce de contrôle du pilote LED WS2812B, qui peut contrôler une perle de lampe LED ou plusieurs modules LED avec une seule ligne de signal. Ses principales caractéristiques sont les suivantes:

- Tension de tenue du port de sortie 15V
- Régulateur de tension intégré à la puce, l'alimentation 24 V ne nécessite qu'une résistance de chaîne à la broche IC VDD, non régulateur de tension externe
- Circuit de réglage des niveaux de gris (256 niveaux de niveaux de gris réglables)
- Circuit de mise en forme du signal intégré
- Mise en forme et sortie de la forme d'onde, pour garantir que la distorsion de la forme d'onde de la ligne ne s'accumulera pas
- Réinitialisation à la mise sous tension et circuit de réinitialisation à la mise hors tension intégrés; Terminal Le terminal de commande PWM peut atteindre 256 niveaux de réglage, la fréquence de balayage n'est pas inférieure à 400 Hz / s
- Interface série, peut recevoir et décoder des données via une ligne de signal
- Distance de transmission de deux points de plus de 10 mètres sans ajouter de circuit
- Lorsque le taux de rafraîchissement est de 30 images / seconde, la cascade du mode basse vitesse n'est pas inférieure à 512 points, le mode haute vitesse n'est pas inférieur à 1024 points et la vitesse de transmission des données est de 800 Kbps.

I.6. Test de la télécommande infrarouge

I.6.1. Introduction à la télécommande infrarouge

Le kit de télécommande infrarouge sans fil se compose d'une mini télécommande infrarouge ultra-mince et d'un Récepteur infrarouge 38KHz. La télécommande infrarouge ultra-mince Mini a 17 touches de fonction et une portée de transmission jusqu'à 8 mètres.

Il est très approprié pour le contrôle à distance de divers appareils à l'intérieur. Le module de réception infrarouge peut recevoir le signal de télécommande modulé standard à 38 kHz. Par la programmation du BLE-UNO, le décodage du signal de télécommande peut être réalisé, de sorte que divers des robots télécommandés et des œuvres interactives peuvent être réalisés. Le kit est illustré à la Figure I.30.

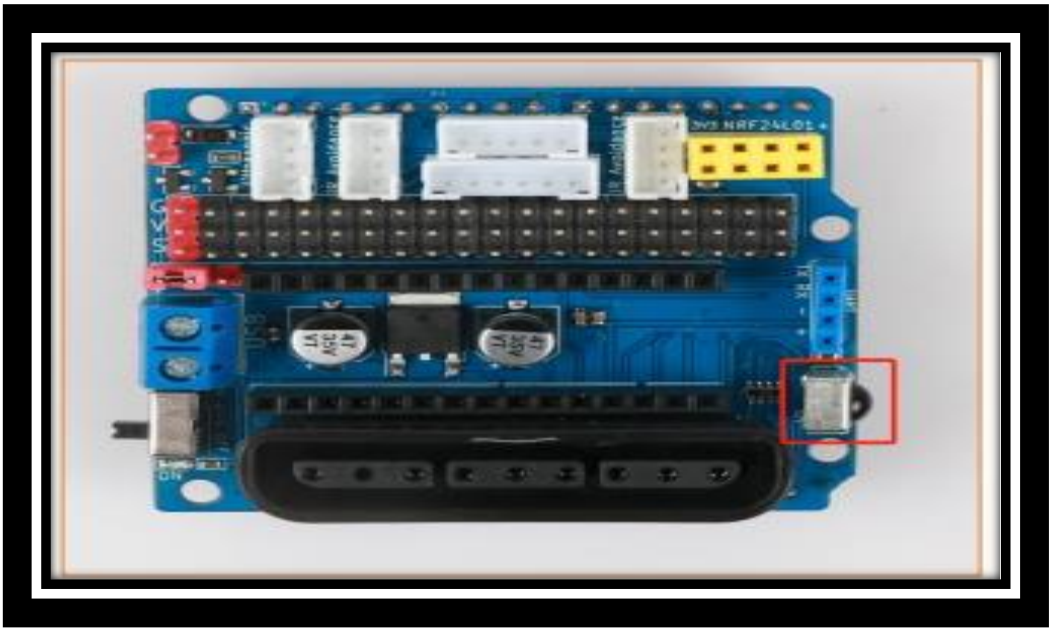


Figure I.30 : Carte physique du kit de télécommande infrarouge

Le système de télécommande infrarouge commun est principalement divisé en trois parties: modulation, transmission et réception. La partie émettrice de la télécommande infrarouge est principalement composée d'un clavier matrice, une télécommande ASIC, un excitateur et une diode électroluminescente infrarouge.

La télécommande ASIC est la partie centrale du système de transmission, et sa partie interne est composée d'un circuit oscillant, une synchronisation circuit, un générateur de signaux de balayage, un codeur d'entrée clé, un décodeur de commande, un convertisseur de code utilisateur, un circuit de modulation numérique et un amplificateur tampon.

Il peut générer le signal d'impulsion de balayage clé et peut décoder le code clé du bouton, puis obtenir la commande à distance de la clé (impulsion code télécommande) à travers l'encodeur de commande à distance, la modulation d'amplitude d'impulsion par la porteuse de 38KHZ, et la télécommande commande Le signal modulé excite la diode infrarouge pour émettre un signal de télécommande infrarouge.

Dans le récepteur infrarouge, un dispositif de conversion photoélectrique (généralement une photodiode ou un phototransistor, que nous utilisons ici est une photodiode PIN) convertit le signal de commande de lumière infrarouge reçu en un signal électrique correspondant. Étant donné que le signal reçu est très faible et que les interférences sont particulièrement grande, afin d'obtenir une détection et une conversion précises du signal, en plus de dispositif de conversion photoélectrique infrarouge haute performance, il est également raisonnable de sélectionner et de concevoir un circuit forme avec de bonnes performances. Le dispositif de conversion photoélectrique le plus couramment utilisé est une photodiode. Lorsque la surface photosensible de la jonction PN de la photodiode est exposée à la lumière, le semi-conducteur

Le matériau de la jonction PN absorbe l'énergie lumineuse et convertit l'énergie lumineuse en énergie électrique. Lorsqu'un une tension inverse est appliquée à la photodiode, le courant inverse dans la diode changera à mesure que la lumière incidente changements d'intensité. Plus l'irradiance de la lumière est grande, plus le courant inverse est important. Autrement dit, le courant inverse de la photodiode change avec la fréquence de l'impulsion lumineuse incidente.

Dans le chariot véhicule, le récepteur infrarouge intégré a trois broches, y compris l'alimentation broches, mise à la terre et broches de sortie de signal. Son circuit est illustré à la Figure I.31.

Le condensateur céramique 104 est un condensateur de découplage qui filtre les interférences du signal de sortie.

La 1ère extrémité est l'extrémité de sortie du signal démodulé, qui est directement connecté au port n° 2 de l'Arduino du puce unique micro-ordinateur. En cas de transmission d'un signal codé infrarouge, après avoir été traité par l'infrarouge connecteur, la sortie est un signal carré après la détection et la mise en forme, et est directement fournie au micro-ordinateur mono puce, et l'opération correspondante est effectuée pour atteindre l'objectif de contrôler le moteur.

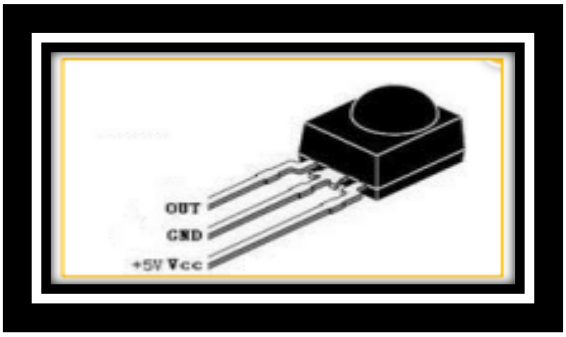


Figure I.31 : Diagramme des broches du récepteur infrarouge

Chapitre II :

Fonctionnement et programmes

II.1. Introduction

Dans ce chapitre nous allons essayer de donner, brièvement, quelques notions des principes de fonctionnement des différents compartiments de notre véhicule ainsi que les programmes utilisées pour leur mise en marche.

II.2. Moteur à courant continu

II.2.1. Principe de contrôle de la vitesse du moteur à courant continu

Quatre moteurs à courant continu avec entraînement MX1616L haute puissance permettent au "Véhicule" de fonctionner plus rapidement que voiture classique à deux roues, le temps d'accélération est plus court et la structure est plus stable. Cependant, application réelle, nous devons ajuster la vitesse de la voiture en raison de facteurs environnementaux ou autres, mais cela n'affecte pas la direction avant, arrière, arrêt et flexible de la voiture, nous utilisons donc PWM pour contrôler la vitesse du moteur (Remarque: PWM est un moyen de simuler la sortie de simulation via des ondes carrées avec un devoir différent cycles.), le port Arduino PWM produit une série d'ondes carrées à fréquence fixe, la puissance et le courant du moteur peuvent être amplifiés après réception du signal, modifiant ainsi la vitesse du moteur. La vitesse, la coordination de deux moteurs sur les roues droite et gauche peut atteindre l'avant, l'arrière, le virage et autres fonctions de la voiture. La figure I.13 montre le diagramme de séquence des cycles de service PWM.

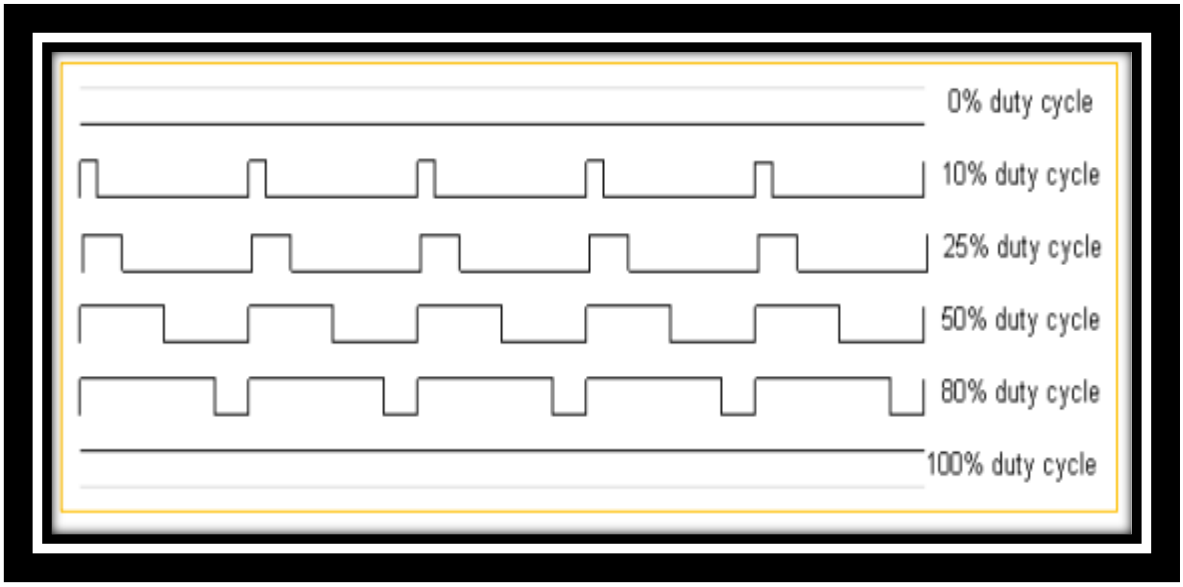


Figure II.1 : Diagramme de séquence des cycles de service PWM

Dans Arduino, la tension analogique ne peut pas être sortie, seulement une valeur de tension numérique de 0 ou 5 V, nous pouvons utiliser une tension élevée compteur de résolution et les rapports cycliques de la méthode de modulation d'onde carrée pour coder un niveau spécifique de signal analogique.

Le signal PWM est toujours numérique, car à tout moment, la pleine amplitude de l'alimentation DC est soit 5V (ON) ou 0V (OFF).

La source de tension ou de courant est ajoutée à la charge analogique avec un ON ou OFF séquence d'impulsions répétitives. Lorsque l'alimentation CC est ajoutée à la charge, l'alimentation est sous tension, sinon l'alimentation est coupée. Tant que la bande passante est suffisante, toute valeur analogique peut utiliser PWM pour encoder. La valeur de la tension de sortie est calculée par le temps d'activation et de désactivation.

Tension de sortie = (temps d'activation / temps d'impulsion) * tension maximale. La figure 2 montre la tension correspondante au changement d'impulsion.

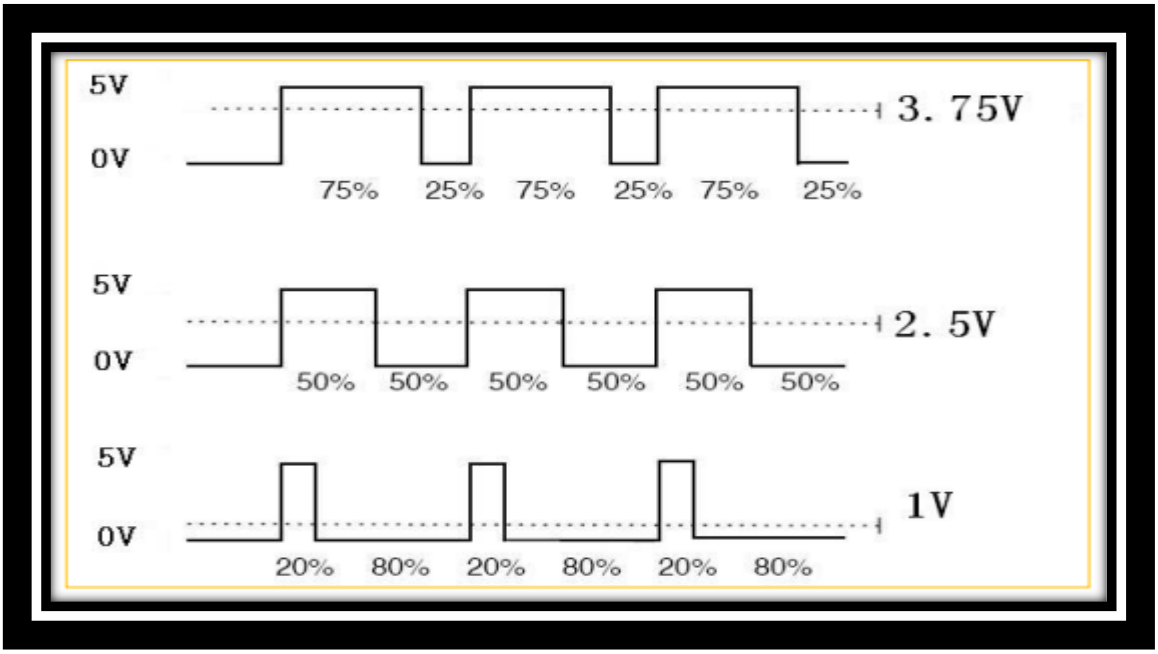


Figure II.2 : Relation entre impulsion et tension

II.2.2. Procédure expérimentale de test moteur

Une fois la connexion terminée, nous ne savons toujours pas si le moteur peut fonctionner normalement et la direction est cohérente, nous devons donc faire un test simple. Les étapes du test sont les suivantes:

- On connecte la carte de contrôle BLE-UNO et l'ordinateur via un câble USB
- On ouvre le programme de test moteur sur la carte de commande principale BLE-UNO (programme ci-dessous).
- On met l'appareil sous tension pour tester et observer la rotation des roues en utilisant le programme ci-après.

```
#define IN1_PIN 6
#define IN2_PIN 10
#define IN3_PIN 5
                          #define IN4_PIN 9
void setup() {
  Serial.begin(9600);
 pinMode(IN1_PIN, OUTPUT);
 digitalWrite(IN1_PIN, LOW); // When not sending PWM, we want it low
 pinMode(IN2_PIN, OUTPUT);
 digitalWrite(IN2_PIN, LOW); // When not sending PWM, we want it low
 pinMode(IN3_PIN, OUTPUT);
 digitalWrite(IN3_PIN, LOW); // When not sending PWM, we want it low
 pinMode(IN4_PIN, OUTPUT);
 digitalWrite(IN4_PIN, LOW); // When not sending PWM, we want it low
}
void loop() {
 analogWrite(IN1_PIN, 200);
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, 200);
 delay(5000);
 //******** ******************************//forward
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, LOW);
 delay(1000);//*****************************************//stop
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, 200);
 analogWrite(IN3_PIN, 200);
 analogWrite(IN4_PIN, LOW);
 delay(5000);//*********************************************//back
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, LOW);
 delay(1000);
 //******* ************************************//stop
 analogWrite(IN1_PIN, 200);
analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, 200);
 analogWrite(IN4_PIN, LOW);
 delay(3000);
 //*****************************************//left
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, LOW);
 delay(1000); //*****************************************//stop
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, 200);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, 200);
 delay(3000);//*** ************************************//right
}
```

II.3. Évitement des obstacles infrarouges et étapes expérimentales du test du module de recherche

- On fixe les deux capteurs sur le chariot et on les connecte à la carte d'extension BLE-UNO avec des fils (terminé).
- On ouvre le programme de test des modules infrarouges d'évitement d'obstacles et de recherche sur BLE-UNO
- On ouvre le moniteur série.
- On ouvre l'interrupteur du boîtier de batterie. À ce moment, le voyant d'alimentation du module s'allume. On place l'obstacle à 10 cm devant le tube émetteur infrarouge et le tube récepteur, et on affine le potentiomètre jusqu'à ce que le voyant de sortie s'allume. On observe à ce moment la sortie de l'obstacle infrarouge. Le signal d'évitement sur le moniteur série est 0. Lorsque le signal d'évitement d'obstacle infrarouge sur le moniteur série de port de l'obstacle est retiré, la fonction d'évitement d'obstacles infrarouge s'est avérée être normale.
- On utilise un couvercle pour bloquer la lumière près de la photorésistance, puis on observe que la valeur de sortie du le signal de recherche de lumière sur le moniteur série est relativement grand, puis on retire l'obstruction. On constate que la valeur de sortie du signal de recherche de lumière sur le moniteur série devient plus petite. Nous pouvons également utiliser la lampe de poche du téléphone mobile pour éclairer la photorésistance pour observer le changement de la valeur de sortie du signal de recherche de lumière sur le moniteur série.

```
const int LeftAvoidancePin = 12;
const int RightAvoidancePin = A5;
const int LeftLightPin = A3;
const int RightLightPin = A4;
int dl, dr, LL, LR;
void setup() {
 Serial.begin(9600);
 pinMode(LeftAvoidancePin, INPUT);
 pinMode(RightAvoidancePin, INPUT);
 pinMode(LeftLightPin, INPUT);
 pinMode(RightLightPin, INPUT);
 delay(1000);
}
void loop() {
 dl = digitalRead(LeftAvoidancePin);
 dr = digitalRead(RightAvoidancePin);
 LL = analogRead(LeftLightPin);
 LR = analogRead(RightLightPin);
 Serial.print("LeftAvoidance:");
 Serial.print(dl);
 Serial.print(" ");
 Serial.print("RightAvoidance:");
 Serial.println(dr);
 Serial.print("LeftLight:");
 Serial.print(LL);
 Serial.print(" ");
 Serial.print("RightLight:");
 Serial.println(LR);
 delay(1000);
}
```

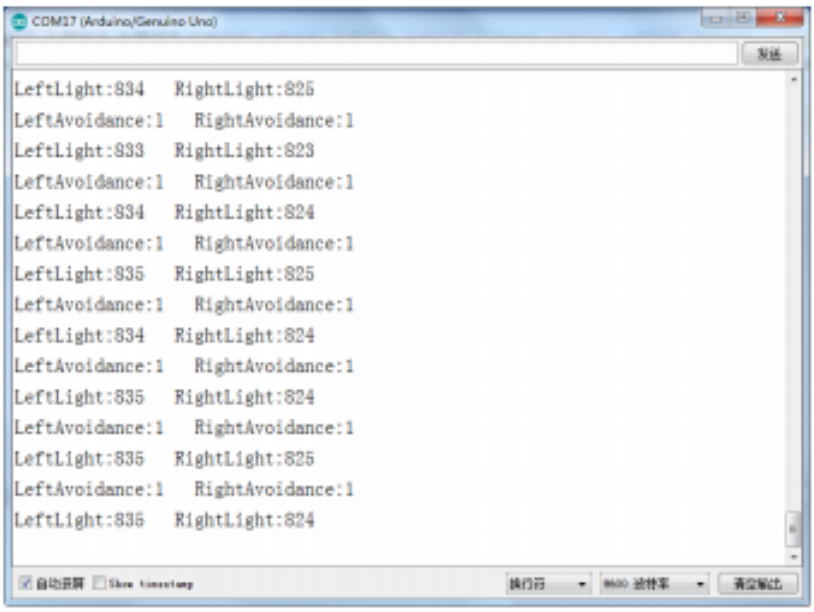


Figure II.3 : Diagramme schématique des données en l'absence d'obstacle ou de lumière ambiante

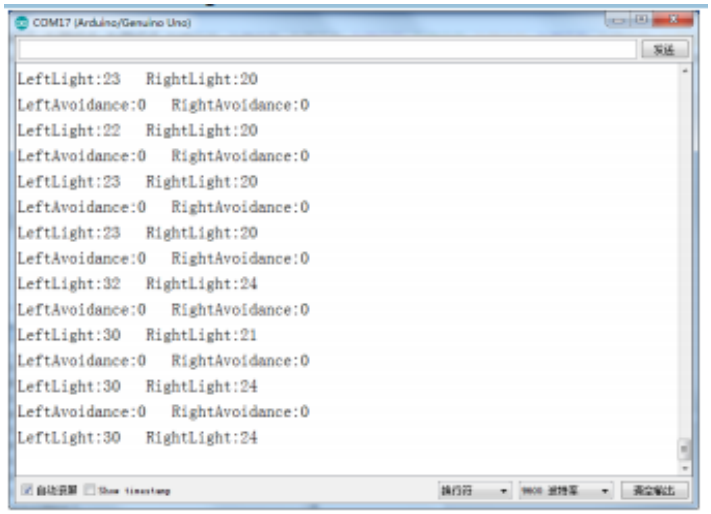


Figure II.4 : Diagramme schématique des obstacles et d'une forte lumière ambiante

II.3.1. Logique du logiciel d'évitement des obstacles infrarouges

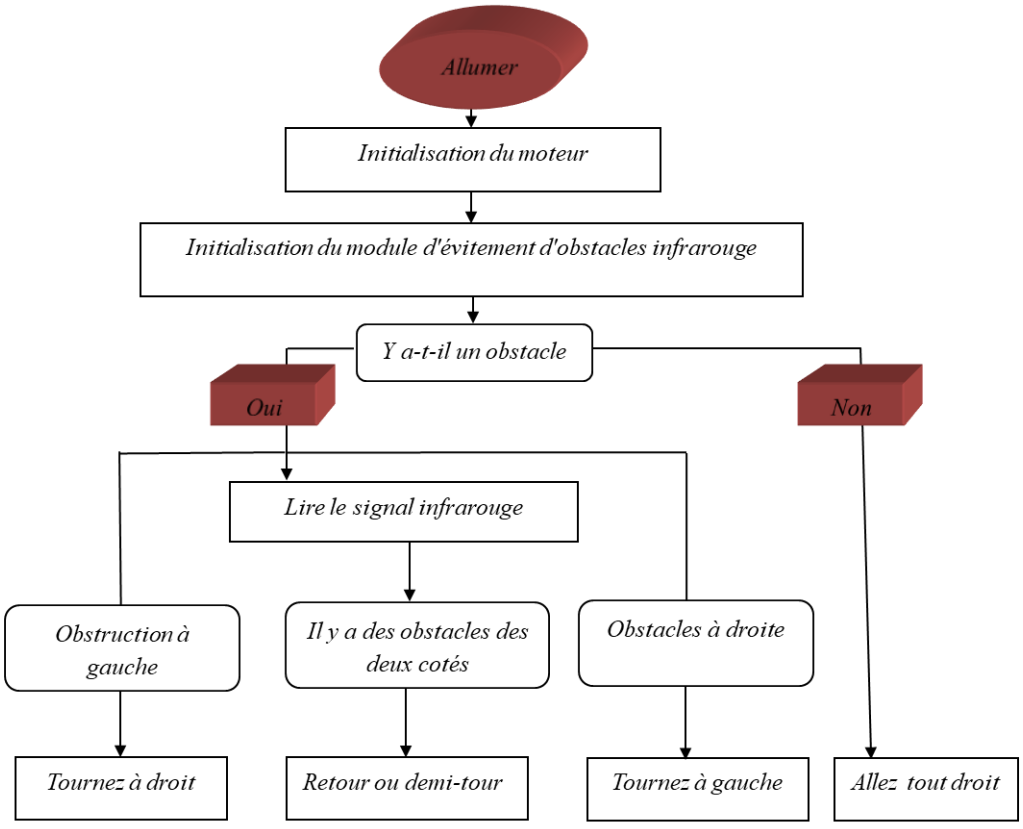


Figure II.5 : Organigramme logique d'évitement des obstacles infrarouges

II.3.3. Code de programme de la fonction d'évitement d'obstacles infrarouge

Comprendre la structure de programmation de la fonction d'évitement d'obstacles infrarouge, nous brûlons d'abord un programme d'évitement d'obstacles infrarouge au tableau de commande principal du véhicule;

```
#define IN1_PIN 6 // PWMB
#define IN2_PIN 10 // DIRB --- right
#define IN4_PIN 9 // PWMA
#define IN3_PIN 5 // DIRA --- left
const int leftPin = 12;
const int rightPin = A5;
byte LeftValue,RightValue;
void setup()
{
 Serial.begin(9600);
 pinMode(leftPin, INPUT);
  pinMode(rightPin, INPUT);
 delay(1000);
}
void loop()
{
 LeftValue = digitalRead(leftPin);
 RightValue = digitalRead(rightPin);
 if (LeftValue >= 1 && RightValue >= 1)
 {
 analogWrite(IN1_PIN, 180); //the speed value of motorA is val
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, 180); //the speed value of motorB is val
 Serial.print(LeftValue);
 Serial.print(" ");
 Serial.print(RightValue);
 Serial.print(" ");
 Serial.println("go");//*****************************************//forward
 } else if (LeftValue >= 1 && RightValue < 1) {
 analogWrite(IN1_PIN, 200);//the speed value of motorA is 200
 analogWrite(IN2_PIN, 0);
 analogWrite(IN3_PIN, 200);//the speed value of motorB is 200
 analogWrite(IN4_PIN, 0);
 Serial.print(LeftValue);
 Serial.print(" ");
 Serial.print(RightValue);
 Serial.print(" ");
 Serial.println("Turning left");
 delay(300);
 analogWrite(IN1_PIN, 0);
 analogWrite(IN2_PIN, 0);
 analogWrite(IN3_PIN, 0);
 analogWrite(IN4_PIN, 0);
 delay(1000); //*****************************************//Turning left
 } else if (LeftValue < 1 && RightValue < 1) {
 analogWrite(IN1_PIN, 0);
 analogWrite(IN2_PIN, 255); //the speed value of motorA is 255
 analogWrite(IN3_PIN, 0);
 analogWrite(IN4_PIN, 255); //the speed value of motorB is 255
 Serial.print(LeftValue);
 Serial.print(" ");
 Serial.print(RightValue);
 Serial.print(" ");
 Serial.println("Turning around");
```

```
delay(500);
 analogWrite(IN1_PIN, 0);
 analogWrite(IN2_PIN, 0);
 analogWrite(IN3_PIN, 0);
 analogWrite(IN4_PIN, 0);
 delay(1000); //*****************************************//Turning
around
 } if (LeftValue < 1 && RightValue >= 1) {
 analogWrite(IN1_PIN, 0);
 analogWrite(IN2_PIN, 200);//the speed value of motorA is 200
 analogWrite(IN3_PIN, 0);
 analogWrite(IN4_PIN, 200);//the speed value of motorB is 200
 Serial.print(LeftValue);
 Serial.print(" ");
 Serial.print(RightValue);
 Serial.print(" ");
 Serial.println("Turning right");
 delay(300);
 analogWrite(IN1_PIN, 0);
 analogWrite(IN2_PIN, 0);
 analogWrite(IN3_PIN, 0);
 analogWrite(IN4_PIN, 0);
 delay(1000); //*****************************************//Turning
right
 }
}
```

II.3.4. Expérience de la fonction de recherche de lumière

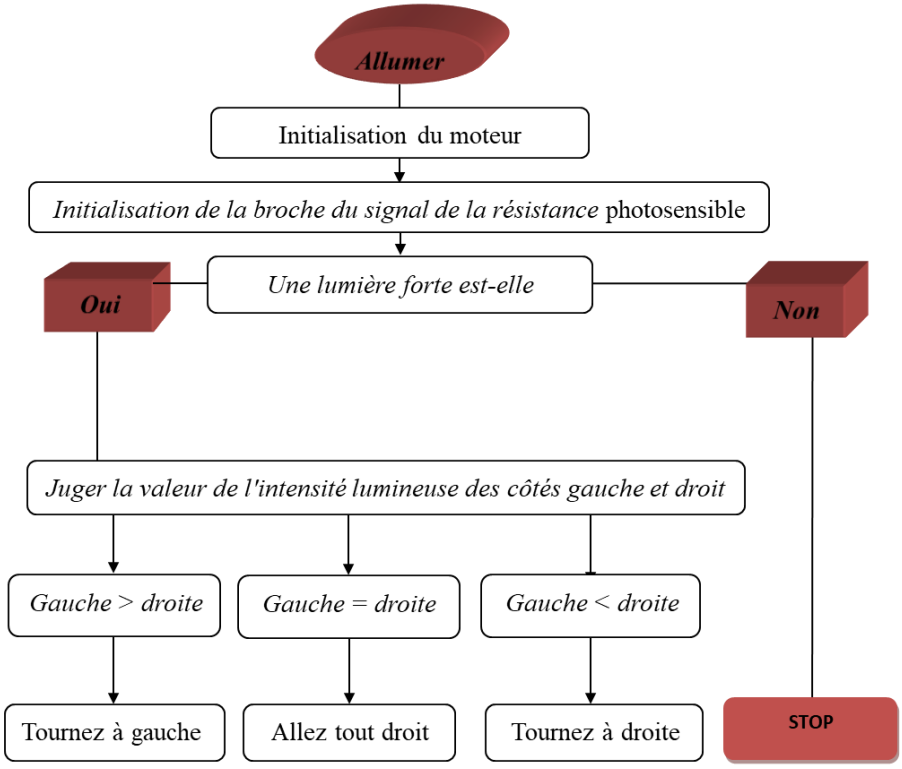


Figure II.6 : Organigramme de conception du logiciel de la fonction Finder

II.3.5. Code de fonction de recherche de lumière véhicule

Comprendre la structure de programmation de la fonction de recherche de lumière, nous gravons d'abord notre programme de fonction de recherche de lumière sur la carte de commande principale du véhicule. On place la voiture sur un sol relativement plat, on allume l'alimentation et on allume la photorésistance sur le module avec une lampe de poche pour observer la progression de la voiture. On déconnecte le module de suivi infrarouge lorsqu'on utilise la fonction de référencement, car l'obstacle infrarouge les broches du module d'évitement et de recherche sont multiplexées avec les broches du module de suivi infrarouge pour provoquer ingérence. Par conséquent, veuillez déconnecter le module de suivi infrarouge lors de l'utilisation de l'obstacle infrarouge module d'évitement et le module de recherche. En observant les résultats de la procédure d'évitement d'obstacles de la voiture, nous examinerons le programme pour approfondir notre compréhension.

```
#define IN1_PIN 6 // PWMB
#define IN2_PIN 10 // DIRB --- right
#define IN4_PIN 9 // PWMA
#define IN3_PIN 5 // DIRA --- left
const int leftPin = A3;
const int rightPin = A4;
float LeftValue,RightValue;
int Angle;
float f;
// When using this program, please pull out the connection line of the
infrared tracing
module, just pull one end of the expansion board.
void setup()
{
 Serial.begin(9600);
 pinMode(leftPin, INPUT);
 pinMode(rightPin, INPUT);
 delay(1000);
}
void loop()
{
      LeftValue = analogRead(leftPin) / 10;
 RightValue = analogRead(rightPin) / 10;
 if ( (LeftValue > 50) && (RightValue > 50)) {
 analogWrite(IN1_PIN, LOW); //the speed value of motorA is val
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, LOW); //the speed value of motorB is val
 } else {
 if (LeftValue > RightValue) {
 Angle = ((float)(RightValue/LeftValue)) * 90;
 } else if (LeftValue < RightValue) {
 Angle = (90 - ((float)(LeftValue/RightValue)) * 90) + 90;
 }
 Serial.println(Angle);
 if (Angle <= 90) {
 f = (float)(Angle) / 90;
 analogWrite(IN1_PIN, (float)(200 * f)); //the speed value of motorA is val
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, 200); //the speed value of motorB is val
 }
 if (Angle > 90) {
 f = (float)(180 - Angle) / 90;
 analogWrite(IN1_PIN, 200); //the speed value of motorA is val
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW );
 analogWrite(IN4_PIN, (float)(200 * f)); //the speed value of motorB is val
 }
 }
}
```

II.4. Servomoteur

II.4.1. Fonctionnement de l'appareil à gouverner

Le signal de commande entre dans la puce de modulation du signal à partir du canal à l'extrémité de réception pour obtenir une tension de polarisation CC.

Il a un circuit de référence à l'intérieur, qui génère un signal de référence avec une période de 20 ms et une largeur de 1,5 milliseconde, et compare la tension de polarisation CC obtenue avec la tension du potentiomètre pour obtenir une sortie de différence de tension. Enfin, la sortie de tension positive et négative du à la différence de tension avec la puce d'entraînement du moteur détermine l'avant et l'arrière du moteur.

La vitesse est constante, le potentiomètre est tourné par le réducteur en cascade, de sorte que lorsque la tension la différence est 0, le moteur cesse de tourner. Lorsque la carte de circuit de commande reçoit le signal de commande de la ligne de signal, le moteur est commandé pour tourner, et le moteur entraîne une série de trains d'engrenages pour ralentir, puis conduit au volant de sortie. L'arbre de sortie de l'appareil à gouverner et le potentiomètre de retour de position sont connectés. Quand le volant tourne, le potentiomètre de retour de position est entraîné.

Le potentiomètre affichera un signal de tension à la carte de circuit de commande pour rétroaction, puis la carte de circuit de commande détermine le moteur selon la position. Le sens et la vitesse de rotation pour atteindre l'arrêt cible. Le workflow est:

Signal de commande → carte de circuit de commande → rotation du moteur → décélération du train d'engrenages → volant rotation → potentiomètre de retour de position → retour du circuit imprimé de commande.

La période du signal de commande du servo est un signal de modulation de largeur d'impulsion (PWM) de 20 ms, dans lequel l'impulsion la largeur est de 0,5 à 2,5 ms, et la position du volant correspondante est de 0 à 180 degrés, ce qui varie linéairement. C'est-à-dire, lui donner une certaine largeur d'impulsion, son arbre de sortie maintiendra une certaine correspondance l'angle, quel que soit le changement de couple externe, jusqu'à ce qu'il reçoive un signal d'impulsion d'une autre largeur, il ?

Changer l'angle de sortie sur le nouveau. La position correspondante est indiquée dans la Figure I.23. Il y a une référence circuit à l'intérieur de l'appareil à gouverner pour générer un signal de référence avec une période de 20 ms et une largeur de 1,5 ms.

Il y a un comparateur qui compare le signal appliqué avec le signal de référence pour déterminer la direction et la taille pour produire le signal de rotation du moteur.

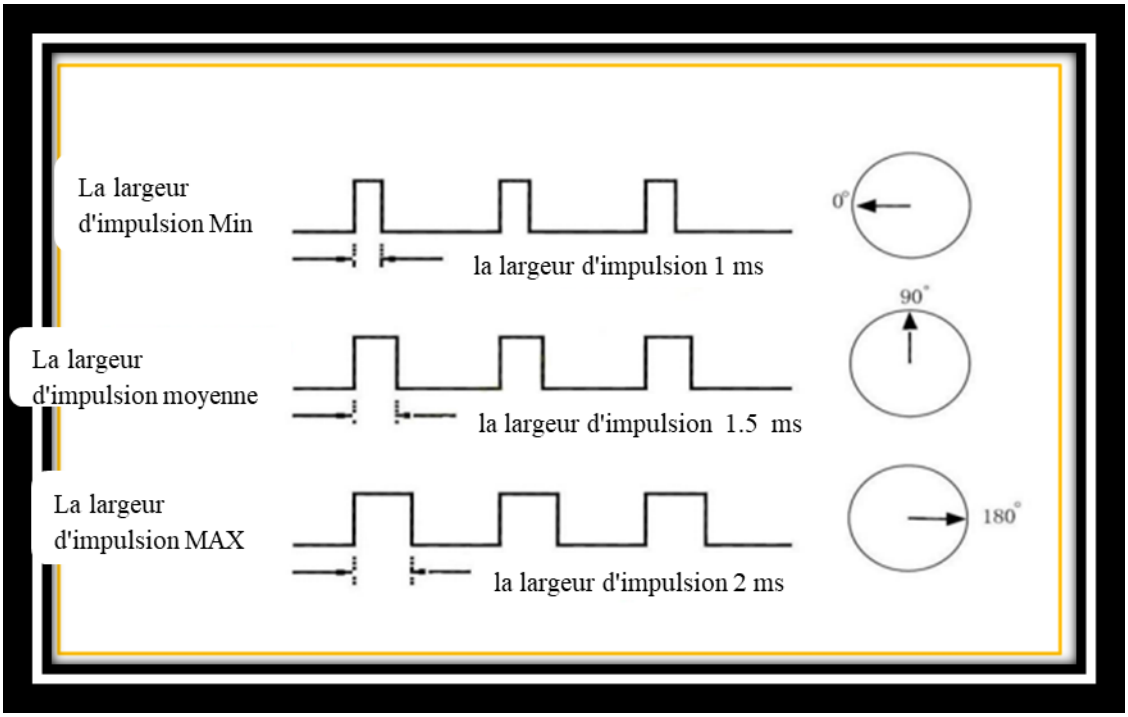


Figure II.7 : Relation entre l'angle de sortie du servo et l'impulsion d'entrée

II.4.2. Correction de l'angle de l'appareil à gouverner

Tard afin de réduire le réglage de l'angle de braquage, nous aurons besoin des procédures suivantes :

- Installer le programme test du Servo sur le panneau de commande,
- l'appareil à gouverner de trois lignes : fil de signal (orange), l'alimentation (rouge), GND (marron), puis mettez les lignes de signal servo (orange) reçu de la bouche 13 Arduino IO, installez l'hélice du gouvernail ouvert ne vis pas fixe Arduino IED moniteur de port série, comme le montre la figure 4.7.4**.**

Figure II.8 : Diagramme schématique du moniteur série ouvert

Après avoir ouvert le moniteur de port série, le moniteur de port série reçoit successivement 0, 90, 180, 90 et la valeur angulaire d'un tel appareil à gouverner, comme le montre la figure II. 4.7.5

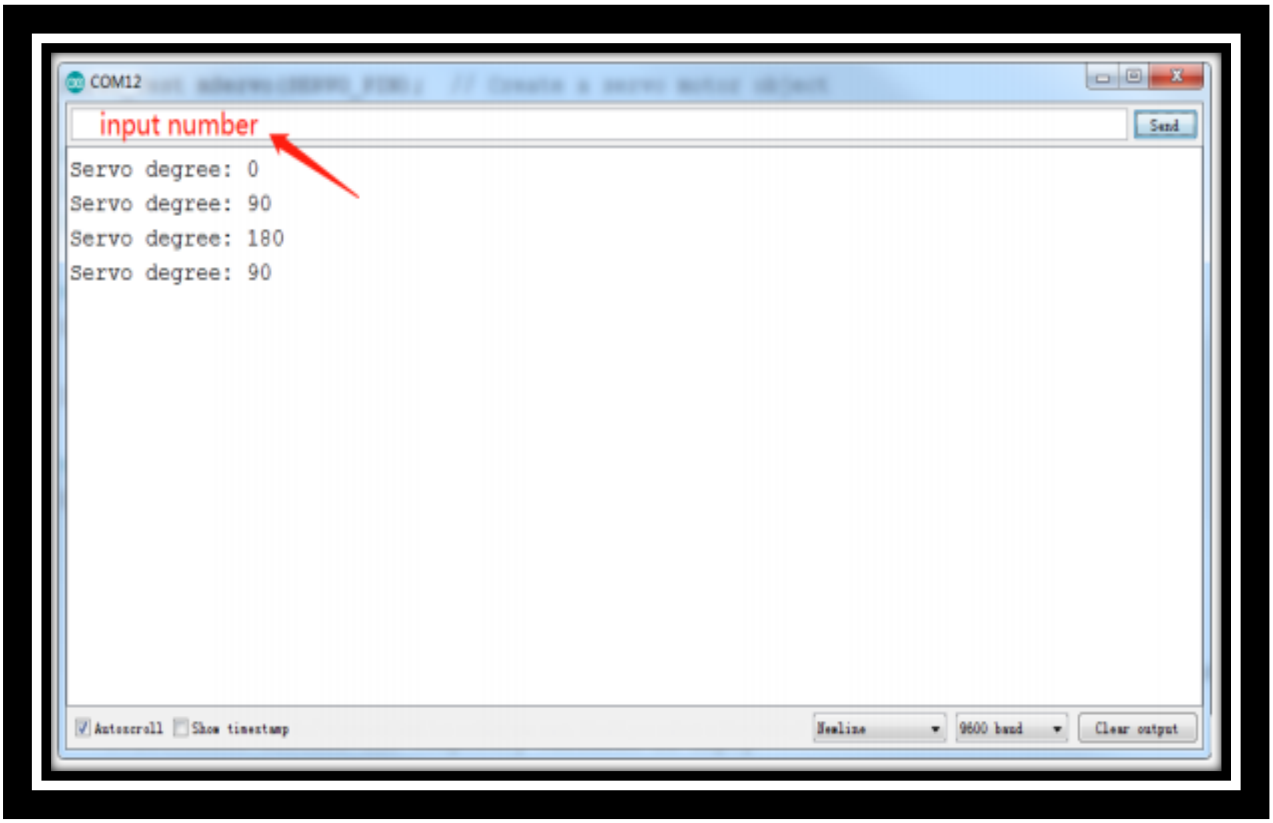


Figure II.9 : Appareil à gouverner d'entrée de port série : Valeur d'angle

Après une entrée d'angle de direction du moniteur de port série respectivement supérieure à la valeur, le boîtier de direction tourner, si après avoir entré 90, l'appareil à gouverner au volant sans perspective figure 4.7.6, la nécessité de garder ne tournez pas l'appareil à gouverner et le bras du pitman sera retiré, puis réinstallez-le dans la figure 4.7.6, telle l'étalonnage est terminé, le boîtier de direction peut être vissé bon boîtier de direction, pour la prochaine étape de fonctionnement.

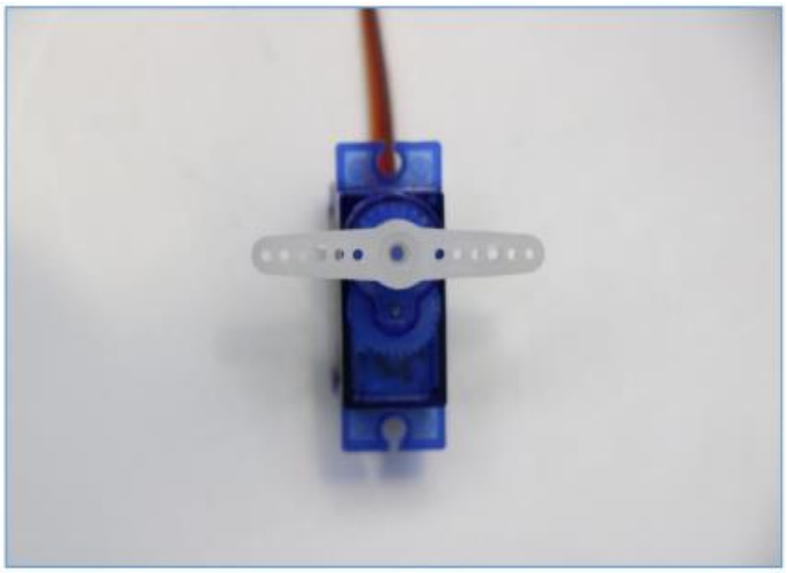


Figure II.10 : Tableau d'étalonnage de l'appareil à gouverner

II.5. Expérience de test du module d'évitement d'obstacles à ultrasons RGB

II.5.1. Le module d'évitement d'obstacles à ultrasons RGB fonctionne

La méthode des ultrasons la distance est la détection d'écho, c'est-à-dire que l'émetteur à ultrasons émet des ondes ultrasonores dans une certaine direction. À en même temps que le temps d'émission, la minuterie commence à compter et les ondes ultrasonores se propagent dans l'air. Lorsque l'obstacle est bloqué sur le chemin, il est immédiatement réfléchi. Lorsque le récepteur reçoit l'onde ultrasonore réfléchie, il arrête immédiatement le chronométrage.

Le diagramme de séquence de travail est illustré à la Figure II.24.

La vitesse de propagation de l'onde ultrasonore dans l'air est de 340 m/s. Selon le temps t enregistré par la minuterie, la distance d du point d'émission de la surface de l'obstacle peut être calculée, c'est-à-dire d = 340. t/2.

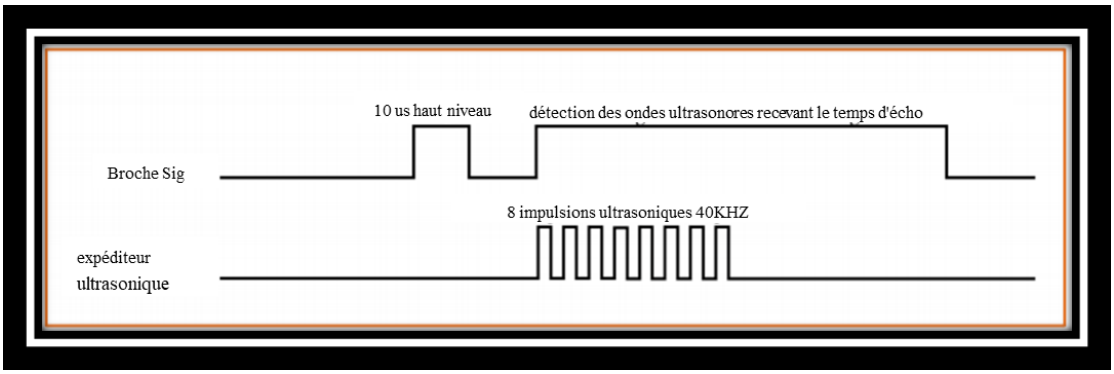


Figure II.11 : Diagramme de séquence de travail ultrasonique.

Analysons ce chronogramme. Tout d'abord, le module de mesure à ultrasons est démarré par le signal de déclenchement. C'est-à-dire que l'hôte doit d'abord envoyer un niveau élevé d'au moins 10 us pour déclencher le module ultrasonore.

Le signal envoyé par le module est automatiquement répondu par le capteur. Le signal d'écho est ce à quoi nous devons prêter attention.

Le niveau élevé de la sortie du signal est le temps nécessaire à l'onde ultrasonore à renvoyer à la réception. Avec une minuterie, vous pouvez enregistrer ce temps et calculer la distance. N'oubliez pas que le résultat est divisé par 2, car le temps total est la somme du temps de transmission et réception.

L'onde ultrasonore étant également une sorte d'onde sonore, sa vitesse acoustique V est liée à la température. Dans utilisation, si la température du milieu de propagation ne change pas beaucoup, on peut estimer que la vitesse ultrasonique est sensiblement constante pendant la propagation. Si la précision de la télémétrie est très élevée, les résultats de mesure doivent être corrigés numériquement au moyen d'une compensation de température. Une fois la vitesse du son déterminée, la distance peut être déterminée en mesurant le temps d'aller-retour ultrasonique. C'est le principe de base de la télémétrie ultrasonique. Comme le montre la Figure I.25:

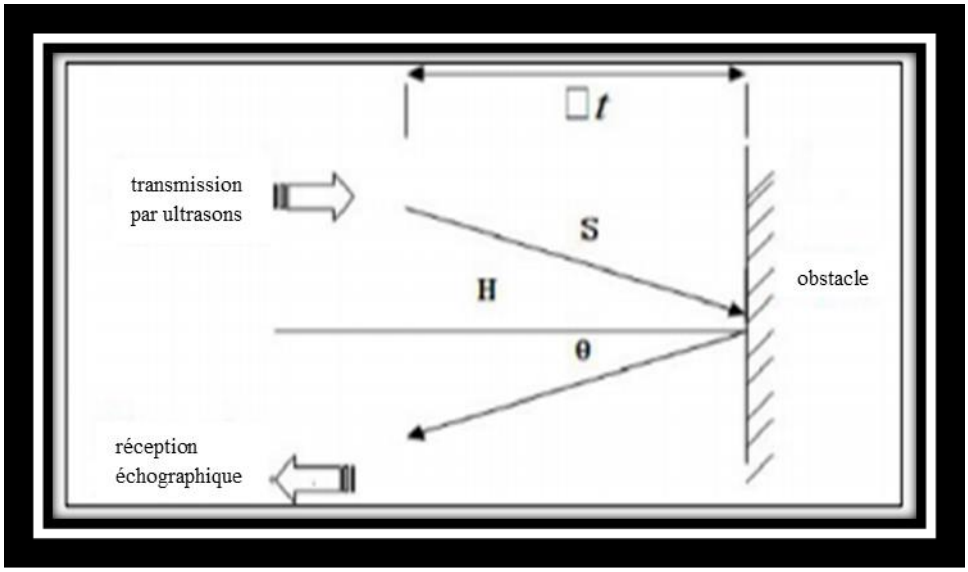


Figure II.12 : Principe de la mesure de distance des ondes ultrasonores.

II.5.2. Schéma de bloc d'obstacles ultrasonique RGB

L'échographie est principalement divisée en deux parties, l'une est le circuit d'émission et l'autre est la réception.

Le circuit de transmission est principalement composé de l'onduleur SP232EN et transducteur de transmission ultrasonique.

Le signal carré de la sortie de 40 kHz du port Arduino est envoyé à une électrode du transducteur à ultrasons à travers l'onduleur primaire, et l'autre est à travers l'onduleur à deux étages. Après avoir été envoyée à l'autre électrode du transducteur à ultrasons, l'onde carrée le signal est appliqué aux deux extrémités du transducteur à ultrasons par cette forme de poussée, et l'intensité d'émission de l'onde ultrasonique peut être améliorée. La sortie utilise deux onduleurs en parallèle pour améliorer la capacité d'entraînement. Le circuit de réception est composé d'un capteur à ultrasons, d'un circuit amplificateur à deux étages et d'un circuit en boucle à verrouillage de phase.

Le signal d'onde réfléchi reçu par le capteur à ultrasons est très faible et le circuit d'amplification à deux étages est utilisé pour amplifier le signal reçu par le capteur.

La boucle à verrouillage de phase, le circuit envoie une requête d'interruption au microcontrôleur après avoir reçu le signal avec la fréquence correspondante à la fréquence centrale de l'oscillateur commandé en tension interne de la boucle à verrouillage de phase LM324, et la bande passante de verrouillage est liée à C15. La fréquence ultrasonore transmise étant de 40 kHz,

les composants concernés sont ajustés pour que la fréquence centrale de la boucle à verrouillage de phase soit de 40 kHz, et seulement le signal de la fréquence est répondu, évitant ainsi l'interférence d'autres signaux de fréquence.

Lorsque le capteur à ultrasons reçoit le signal ultrasonique, il est envoyé à l'amplificateur à deux étages pour l'amplification et le signal amplifié entre dans la détection de boucle à verrouillage de phase. Si la fréquence est de 40 kHz le signal de demande d'interruption de bas niveau est envoyé de la broche 1 à la borne P3.5 du puce unique micro-ordinateur. Le BLE-UNO arrête la minuterie après avoir détecté un niveau bas.

II.5.3. Introduction au mode de contrôle LED RGB du pilote WS2812B

Le protocole de données WS2812B adopte le mode de communication de retour à zéro sur une seule ligne. Après la réinitialisation à la mise sous tension, la borne DIN accepte les données transmises par le contrôleur.

Les 24 premiers bits de données sont extraits par le premier pixel et envoyés au pixel. Les données se verrouillent, les données restantes sont façonnées et amplifié par le circuit de traitement de mise en forme interne, puis la sortie est transmise au suivant en cascade pixel via le port DO, et le signal est réduit de 24 bits chaque fois qu'un pixel est transmis. Le pixel adopte une technologie de mise en forme et de transfert automatique, de sorte que le nombre de pixels en cascade n'est pas limité par transmission du signal, et seule la vitesse de transmission du signal est limitée.

Le verrou de données à l'intérieur de la puce génère différents signaux de contrôle du rapport cyclique aux points OUTR, OUTG et terminaux de commande OUTB en fonction des données 24 bits reçues. Lorsque la borne DIN entre le RESET signal, toutes les puces envoient de manière synchrone les données reçues par chacune. La puce acceptera à nouveau le nouveau données après la fin du signal, après avoir reçu les premières données 24 bits, le port de données est transmis par le DO port, la sortie d'origine des broches OUTR, OUTG, OUTB est conservée avant que la puce ne reçoive le RESET code. Aucun changement, lors de la réception d'un code RESET de bas niveau supérieur à 50 μs, la puce émettra le PWM 24 bits largeur d'impulsion de données qui vient d'être reçue aux broches OUTR, OUTG, OUTB. Les broches et les fonctions de la puce sont illustrées dans Figure I.25 et tableau I.4.

Tableau I.4 Tableau des fonctions des broches de puce WS2812B

Numéro de série	Symbole	Nom de broche	Description de la fonction
1	D0	Lecteur LED	Afficher la sortie de cascade de données
2	GND	Ground	Ground, enfin en cascade
3	IN	Lecteur LED	Afficher la saisie de données
4	VDD	Puissance logique	Alimentation IC

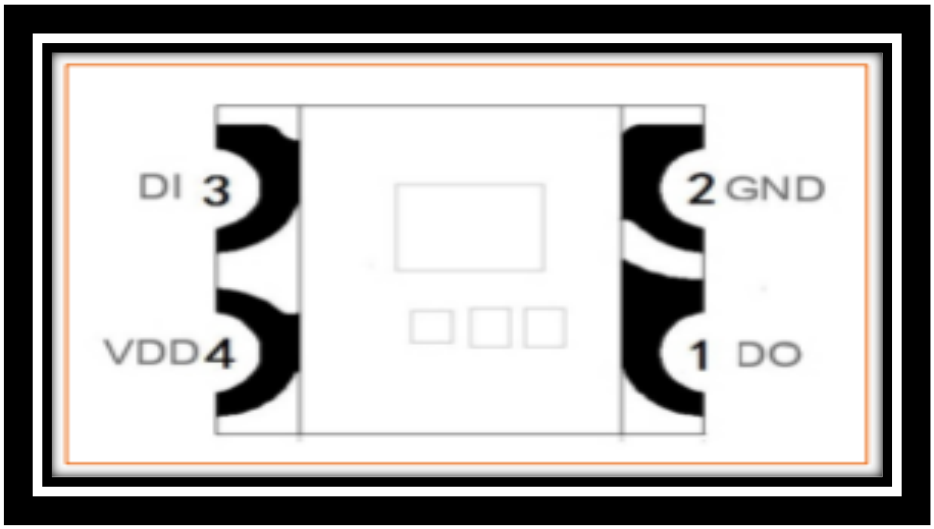


Figure II.13 : Diagramme des broches WS2812B.

Le niveau bas WS2812B est représenté par T0, qui est un niveau bas 0,5µs haut 2µs. T1 se compose d'un faible niveau de 2 µs et faible niveau de 0,5 µs.

Temps de faible niveau supérieur à 50 µs pendant la réinitialisation.

Tableau I.5 : Le niveau bas WS2812B est représenté par T0

T0H	0 mètres, temps haute tension	0.5µs
T0H	1 mètre, temps haute tension	2µs
T0L	0 mètre, temps basse tension	2µs
T0L	1 mètre, temps basse tension	0.5µs
RES	Code de réinitialisation, temps de base tension	>50µs

Le diagramme de forme d'onde de synchronisation est le suivant:

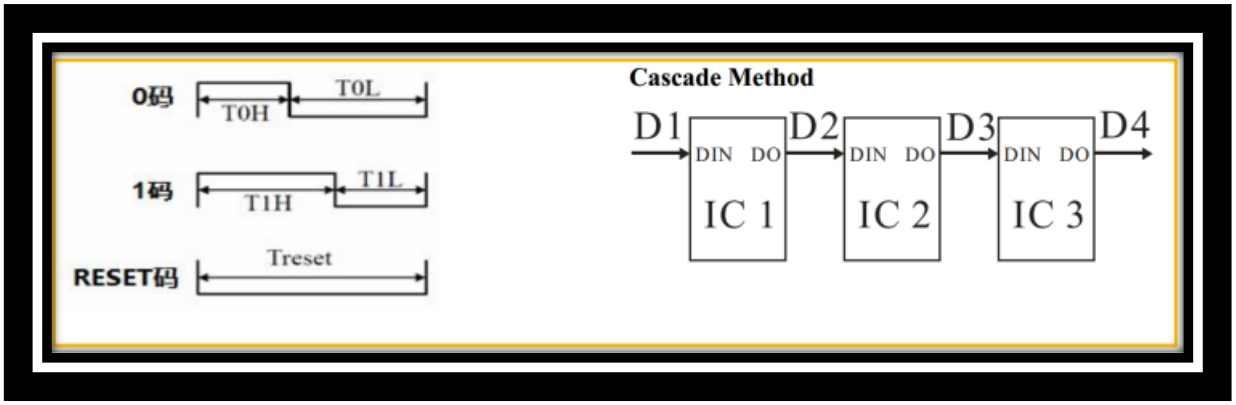


Figure II.14 : Diagramme de synchronisation WS2812B et méthode de connexion

Structure de données 24 bits:

R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

Remarque: ordre élevé en premier, envoyez les données dans l'ordre RGB

II.5.4. Schéma de connexion du module d'évitement d'obstacles à ultrasons RGB et Carte d'extension Arduino

Tableau I.6 : Tableau de connexion du module d'évitement d'obstacles à ultrasons RGB et Carte d'extension Arduino

Ultrasonique RGB	Arduino UNO
VCC	VCC
GND	GND
SIG	3
RGB	2

Figure II.15 : Schéma de principe du câblage du module d'évitement d'obstacles ultrasonique RGB

II.5.5. Expérience de test du module d'évitement d'obstacles à ultrasons RGB

- On installe l'appareil à gouverner et les ultrasons sur le chariot et le connecte correctement.
- On connecte la carte de commande principale de la voiture et l'ordinateur via USB;
- On ouvre le programme de test du module d'évitement d'obstacles à ultrasons RGB sur BLE-UNO

- On ouvre le moniteur série;
- On aligne le module d'évitement d'obstacles à ultrasons RGB avec un obstacle plan, puis on observe la distance entre le module d'évitement d'obstacles à ultrasons RGB et l'obstacle en continu sur le moniteur série (figure II.16); Dans le même temps, la LED RGB sur l'ultrason RGB le module d'évitement d'obstacles s'allume. Cela prouve que le module d'évitement d'obstacles à ultrasons **RGB** fonctionne normalement

```
#include "RGBLed.h"
#define RGB_RED 0xFF0000
#define RGB_GREEN 0x00FF00
#define RGB_BLUE 0x0000FF
#define RGB_YELLOW 0xFFFF00
#define RGB_PURPLE 0xFF00FF
#define RGB_WHITE 0xFFFFFF
const int SingPin = 3;
const int RgbPin = 2;
float distance;
RGBLed mRgb(RgbPin,6);
void setup() {
 Serial.begin(9600);
 Serial.println("Ultrasonic sensor:");
 mRgb.setColor(1,RGB_RED);
 mRgb.setColor(2,RGB_GREEN);
 mRgb.setColor(3,RGB_BLUE);
 mRgb.setColor(4,RGB_YELLOW);
 mRgb.setColor(5,RGB_PURPLE);
 mRgb.setColor(6,RGB_WHITE);
 mRgb.show();
}
void loop() {
 digitalWrite(SingPin, LOW);
 delayMicroseconds(2);
 digitalWrite(SingPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(SingPin, LOW);
 pinMode(SingPin, INPUT);
 distance = pulseIn(SingPin, HIGH) / 58.00;
 Serial.print("distance is :");
 Serial.print(distance);
 Serial.print("cm");
 Serial.println();
 delay(1000);
}
```

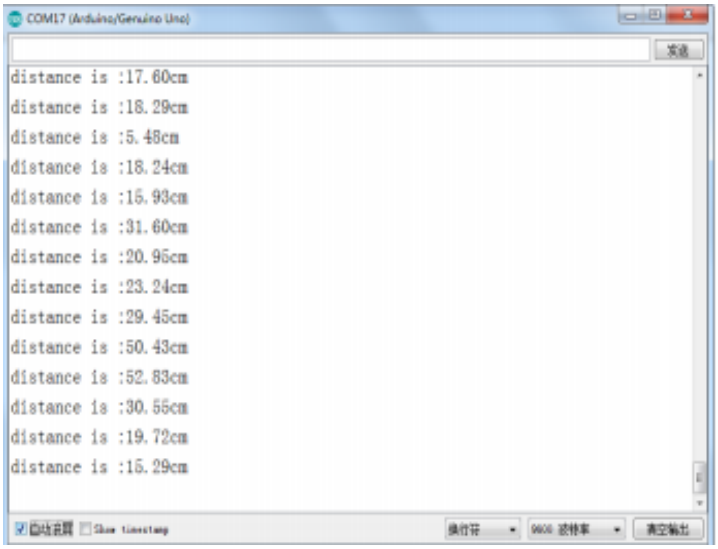


Figure II.16 : Le moniteur série imprime la distance entre le module d'évitement d'obstacles à ultrasons RVB et l'obstacle

II.5.6. Logique logicielle de la fonction d'évitement d'obstacles à ultrasons RGB

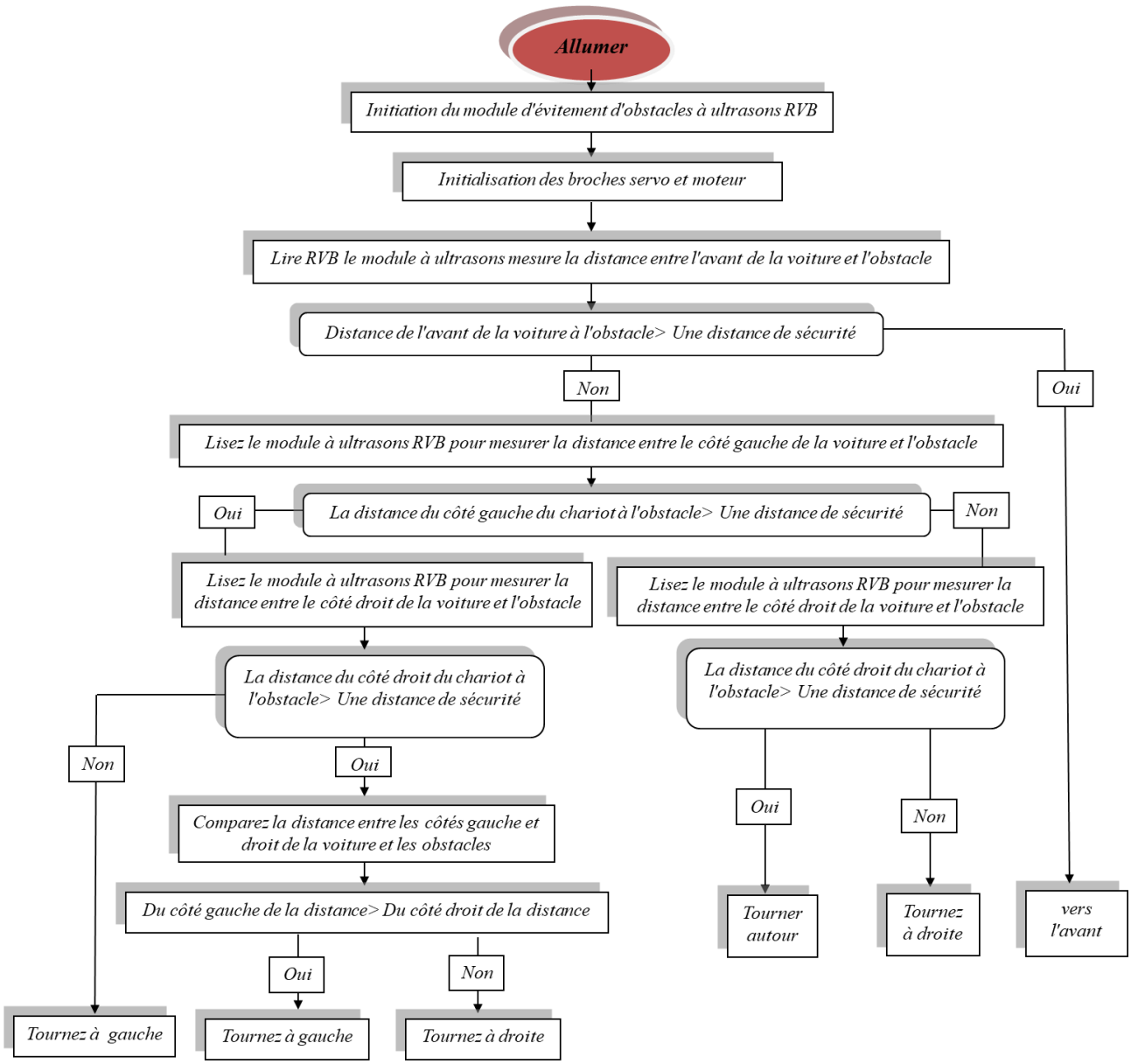


Figure II.17 : Organigramme d'exécution du programme de fonction d'évitement d'obstacles par ultrasons

II.5.7. modules d'évitement d'obstacles ultrasoniques RGB et connexion de fil d'appareil à gouverner

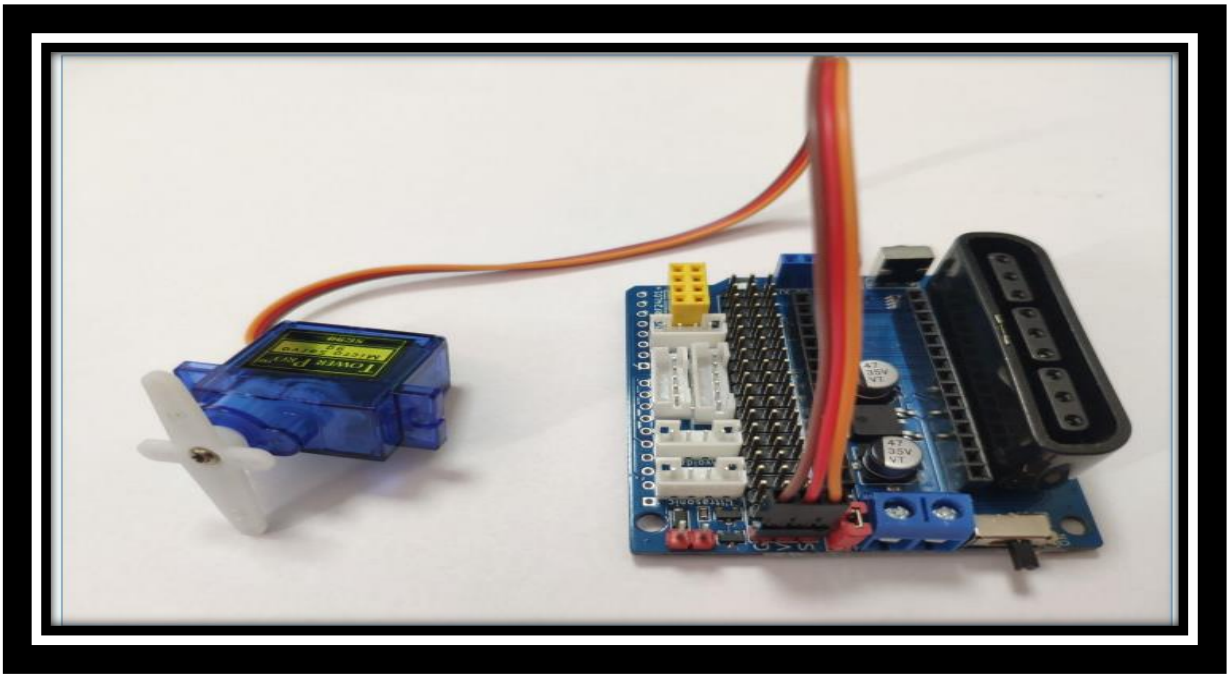


Figure II.18 : Schéma de câblage du servo, du module d'évitement d'obstacles à ultrasons RGB et de la carte d'extension

II.5.8. Fonction d'évitement d'obstacles ultrasonique RGB véhicule Procédure expérimentale

- On ouvre le programme Fonction d'évitement ultrasonique sur la carte de commande principale BLE-UNO;
- On met l'appareil sous tension et on observe la progression de la voiture. En observant les résultats de la procédure d'évitement d'obstacles de la voiture, nous examinerons le programme pour approfondir notre compréhension.

```
#include "RgbUltrasonic.h"
#define IN2_PIN 10 // PWMB
#define IN1_PIN 6 // DIRB --- right
#define IN4_PIN 9 // PWMA
#define IN3_PIN 5 // DIRA --- left
#define SERVO_PIN 13
#define SING_PIN 3
#define RGB_PIN 2
#define UL_LIMIT_MID 12
#define UL_LIMIT_MAX 400
RgbUltrasonic mRgbUltrasonic(SING_PIN, RGB_PIN, SERVO_PIN);/*Define
ultrasonic and servo
pins*/
void setup()
{
 Serial.begin(9600);
 pinMode(IN1_PIN, OUTPUT);
 pinMode(IN2_PIN, OUTPUT);
 pinMode(IN3_PIN, OUTPUT);
 pinMode(IN4_PIN, OUTPUT);
 mRgbUltrasonic.SetServoBaseDegree(90);/*Adjust the initial angle of the
steering gear
according to the steering gear error*/
 mRgbUltrasonic.SetServoDegree(90);/*Set the servo angle*/
}
void loop()
{
 uint16_t FrontDistance, LeftDistance, RightDistance;
 FrontDistance = mRgbUltrasonic.GetUltrasonicFrontDistance();/*The
ultrasonic module
collects the front data*/
 delay(50);
 if ((FrontDistance > UL_LIMIT_MID) && (FrontDistance < UL_LIMIT_MAX))
 /*According to the data collected by the ultrasonic module and the
infrared obstacle
avoidance module,
 it is judged whether there is an obstacle in front, and if there is no
obstacle, go
straight.*/
 {
 analogWrite(IN1_PIN, 200);
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, 200);
 }
 else if ((FrontDistance < UL_LIMIT_MID) || (FrontDistance >
UL_LIMIT_MAX))
 /*According to the data collected by the ultrasonic module and the
infrared obstacle
```

```
analogWrite(IN1_PIN, 0);
 analogWrite(IN2_PIN, 0);
 analogWrite(IN3_PIN, 0);
 analogWrite(IN4_PIN, 0);
 RightDistance = mRgbUltrasonic.GetUltrasonicRightDistance();/*The
ultrasonic module
collects the right side*/
 LeftDistance = mRgbUltrasonic.GetUltrasonicLeftDistance();/*The
ultrasonic module
collects the left side*/


 delay(10);
 if ((RightDistance > UL_LIMIT_MID) && (RightDistance < UL_LIMIT_MAX)
&&
(RightDistance > LeftDistance))
 /*According to the ultrasonic module to collect the data on the left
and right sides
to determine whether there is an obstacle on the right side.*/
 {
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, 200);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, 200);
 Serial.println("testRight");
 delay(380);
 }
 else if ((LeftDistance > UL_LIMIT_MID) && (LeftDistance <
UL_LIMIT_MAX) &&
(LeftDistance > RightDistance))
 /*According to the ultrasonic module to collect the data on the left
and right sides
to determine whether there is an obstacle on the left side.*/
 {
 analogWrite(IN1_PIN, 200);
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, 200);
 analogWrite(IN4_PIN, LOW);
 Serial.println("testLeft");
 delay(380);
 }
 else if ((RightDistance < UL_LIMIT_MID) && (LeftDistance <
UL_LIMIT_MID) )
 /*According to the ultrasonic module to collect the data on the left
and right sides
to determine whether there are obstacles on the left and right sides*/
 {
 analogWrite(IN1_PIN, 200);
 analogWrite(IN2_PIN, 0);
 analogWrite(IN3_PIN, 200);
 analogWrite(IN4_PIN, 0);
 delay(760);
 }
 }
}
```

II.6. Test de la télécommande infrarouge

II.6.1. Principe de fonctionnement de la télécommande infrarouge

Le système de télécommande se compose généralement d'une télécommande (émetteur) et d'un récepteur. Quand vous appuyez sur n'importe quel bouton de la télécommande, la télécommande génère une impulsion de code correspondante pour sortir divers signaux d'impulsion de commande à base infrarouge. L'impulsion est un code de commande informatique.

Le moniteur infrarouge la diode surveille le signal infrarouge puis envoie le signal à l'amplificateur et au limiteur.

Le limiteur contrôle l'amplitude de l'impulsion à un certain niveau indépendamment de la distance entre l'émetteur infrarouge et le récepteur. Le signal CA entre dans le filtre passe-bande. Le filtre passe-bande peut passer l'onde de charge de 30KHZ à 60KHZ, entrez le comparateur par le circuit de démodulation et le circuit d'intégration, et le comparateur délivre des niveaux haut et bas pour restaurer la forme d'onde du signal de l'émetteur. Comme le montre le schéma fonctionnel du système de 4.10.3

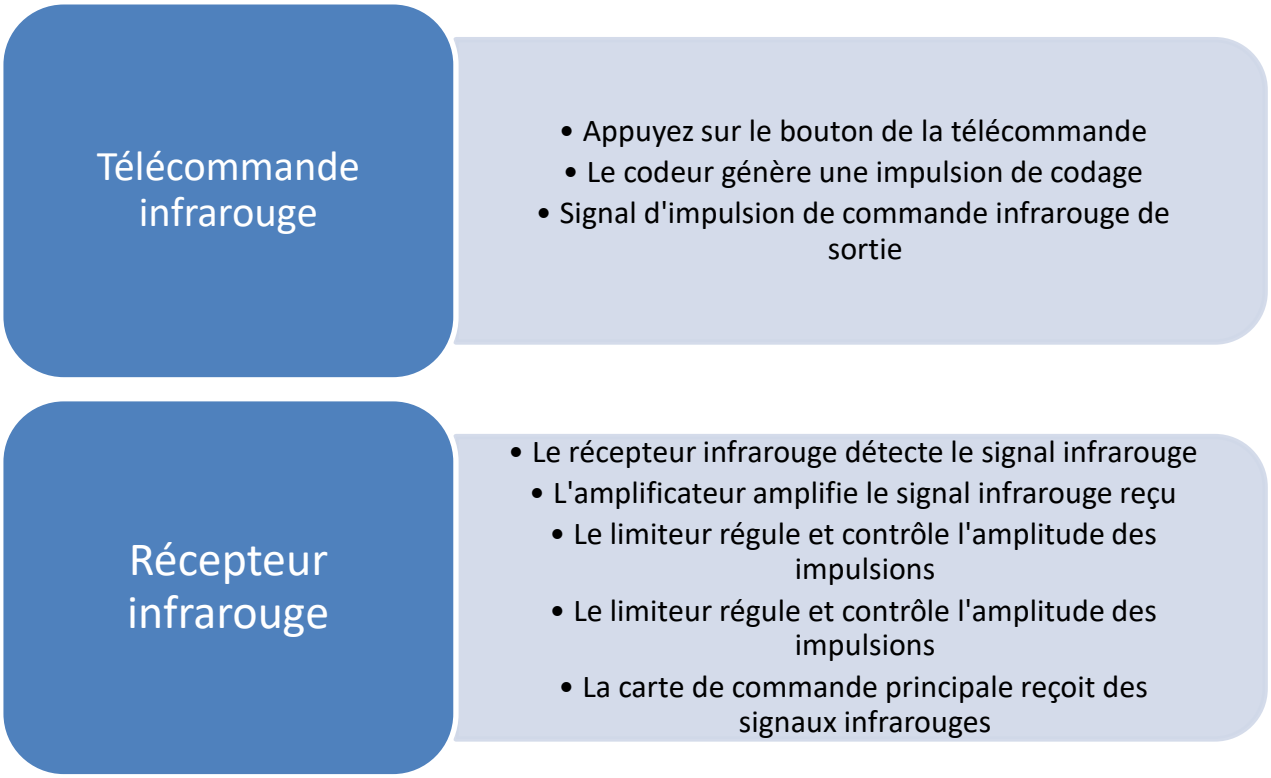


Figure II.19 : Schéma fonctionnel du système de télécommande infrarouge

II.6.2. Étapes expérimentales du test de la télécommande infrarouge

- On connecte la carte de commande principale de la voiture et l'ordinateur via USB;
- On ouvre le programme Touche pressé IR sur BLE-UNO;
- On ouvre le moniteur série
- On active le coupe-batterie;
- On appuie sur le bouton de la télécommande infrarouge contre la voiture pour observer la valeur imprimée sur la série moniteur de port (comme le montre la figure 4.10.4). Si la valeur du bouton de la télécommande infrarouge est la identique à la valeur imprimée sur le moniteur du port série, la communication de la télécommande infrarouge est normale.

```
#include "IRremote.h"
IRremote ir(11);
unsigned char keycode;
char str[128];
void setup() {
 Serial.begin(9600);
 ir.begin();
}
void loop()
{
 if (keycode = ir.getCode()) {
 String key_name = ir.getKeyMap(keycode);
 sprintf(str, "Get ir code: 0x%x key name: %s \n", keycode, (char
*)key_name.c_str());
 Serial.println(str);
 } else {
 // Serial.println("no key");
 }
 delay(110);
}
```

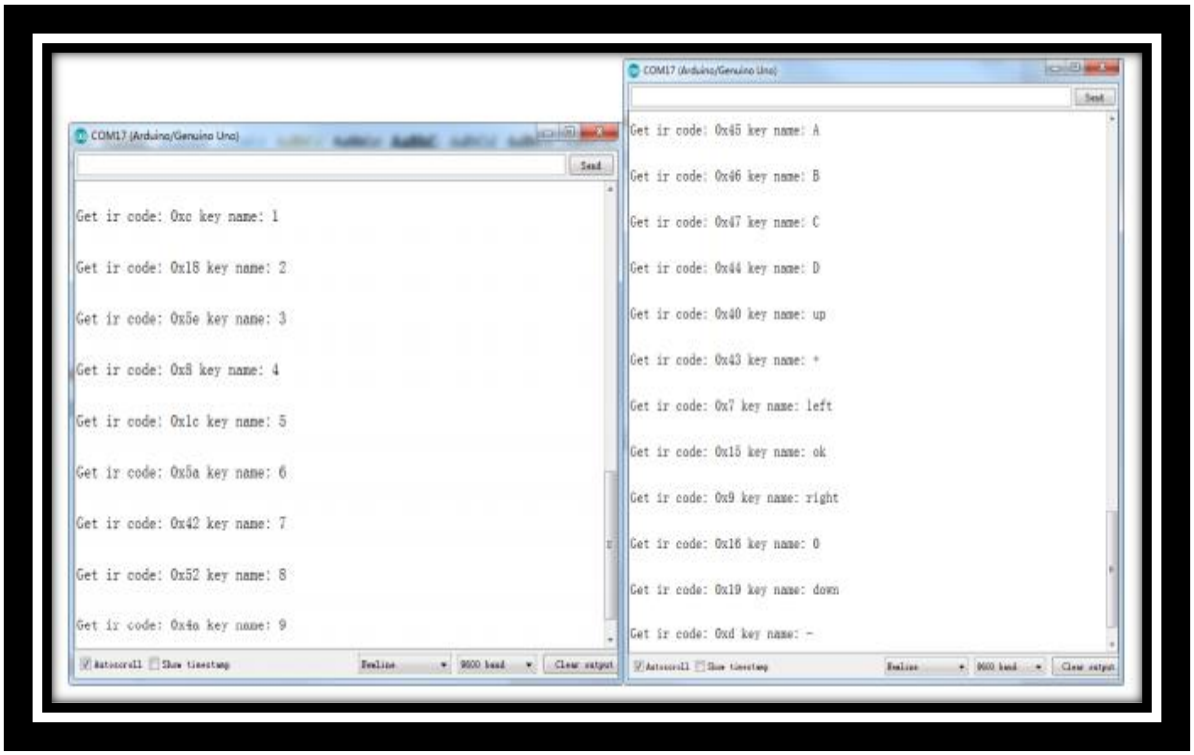


Figure II.20 : Requête de code de télécommande

Dans la figure 4.10.4, nous pouvons voir les deux valeurs du code Ir "0xc" et du nom de clé "1", où "0x45" est le code d'un bouton de la télécommande et «1» est le nom de la fonction du bouton de la télécommande

```
ST_KEY_MAP irkeymap[KEY_MAX] = {
  {"A", 0x45},
  {"B", 0x46},
  {"C", 0x47},
  {"D", 0x44},
  {"up", 0x40},
  {"+", 0x43},
  {"left", 0x07},
  {"ok", 0x15},
  {"right", 0x09},
  {"0", 0x16},
  {"down", 0x19},
  {"-", 0x0d},
  {"1", 0x0c},
  {"2", 0x18},
  {"3", 0x5e},
  {"4", 0x08},
  {"5", 0x1c},}
{"6", 0x5A},
  {"7", 0x42},
  {"8", 0x52},
  {"9", 0x4A}
};
```

II.6.3. Expérience de la fonction de télécommande infrarouge du véhicule

II.6.3.1. Logique du logiciel de la fonction de télécommande infrarouge

La télécommande infrarouge, comme son nom l'indique, consiste à contrôler à distance la voiture via la télécommande. Comment rend-il le véhicule "obéissant" grâce à une petite télécommande? C'est le problème que nous devons explorer. Pour que le véhicule "obéisse", vous devez d'abord le véhicule entendre les commandes de la télécommande contrôler tout le temps, c'est-à-dire que les deux doivent être dans une certaine plage de communication, et deuxièmement, le véhicule peut comprendre". La télécommande indique que vous pouvez comprendre la signification de chaque instruction. Finalement, le véhicule contrôle les membres (quatre roues) selon les instructions de la télécommande pour faire l'action correspondante. C'est le principe de base de la réalisation de la fonction de télécommande infrarouge. La logique logicielle est illustrée à la figure II.21.

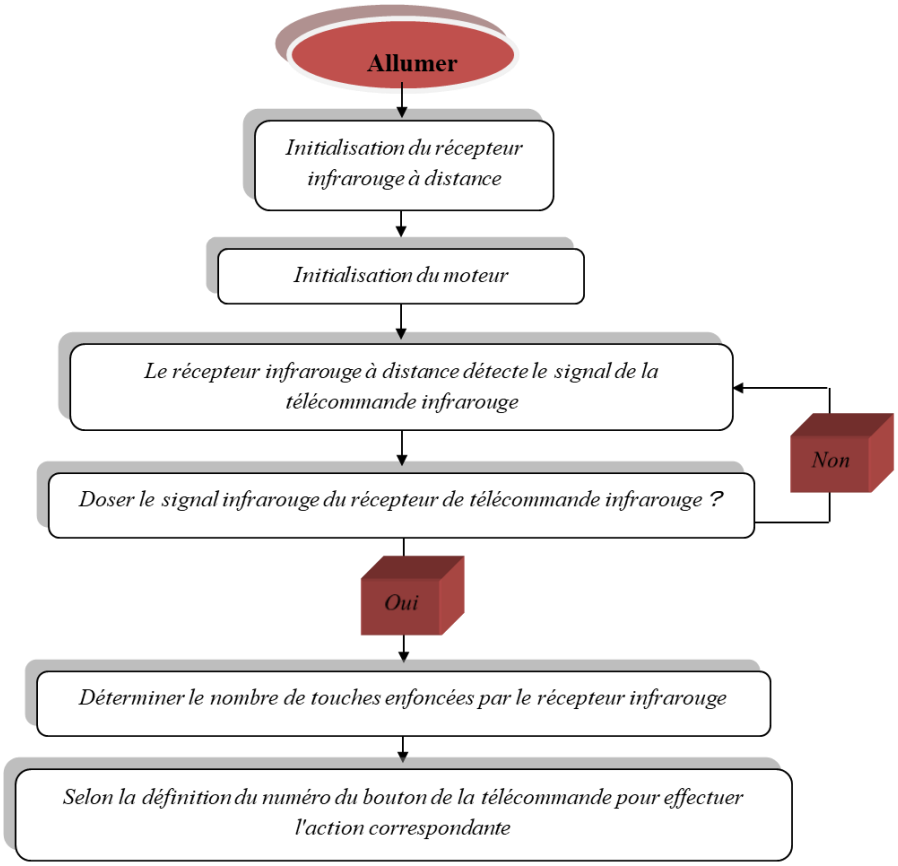


Figure II.21 : Organigramme de la logique du logiciel de télécommande infrarouge

II.6.3.2. Fonction d'évitement d'obstacles ultrasonique RGB véhicule expérimentale

- On ouvre le programme de la fonction de contrôle IR sur la carte de commande principale BLE-UNO;
- On met sous tension et on appuie sur le bouton de fonction défini sur la télécommande (comme illustré à la figure II.22) pour observer la progression de la voiture.

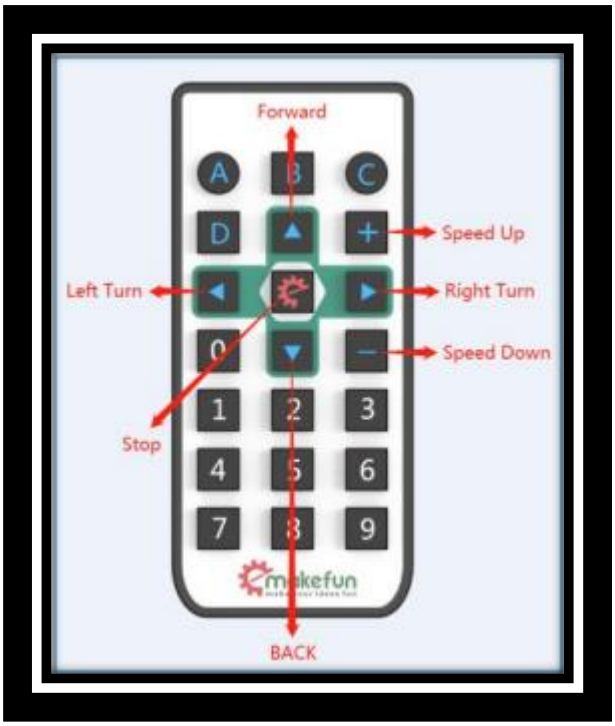


Figure II.22 : Définition de la fonction du bouton de la télécommande infrarouge

En observant les résultats de la procédure d'évitement d'obstacles de la voiture, nous examinerons le programme pour approfondir notre compréhension

```
#include "IRremote.h"
#define IN1_PIN 6 // PWMB
#define IN2_PIN 10 // DIRB --- right
#define IN4_PIN 9 // PWMA
#define IN3_PIN 5 // DIRA --- left
int RECV_PIN = 11;//Define the infrared receiver pin to 12
long expedite1 = 0x43;
long expedite2 = 0x0d;
long up = 0x40;
long down = 0x19;
long stop = 0x15;
long left = 0x07;
long right = 0x09;
static int val = 160;
IRremote irrecv(RECV_PIN);
void setup() {
 Serial.begin(9600);
 irrecv.begin();
}
```

```
void loop() {
 byte irKeyCode;
 if (irKeyCode = irrecv.getCode())
 {
 if (irKeyCode == up) {
 analogWrite(IN1_PIN, val);//the speed value of motorA is val
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, val); //the speed value of motorA is val
 }
 else if (irKeyCode == expedite1) {
 val += 20;
 if (val >= 240)
 {
 val = 255;
 }
 }
 else if (irKeyCode == expedite2) {
 val -= 20;
 if (val <= 20)
 {
 val = 0;
 }
 }
 else if (irKeyCode == stop) {
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, LOW);
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, LOW);
 }
 else if (irKeyCode == left) {
 analogWrite(IN1_PIN, val);
 analogWrite(IN2_PIN, LOW); //the speed value of motorA is val
 analogWrite(IN3_PIN, val);
 analogWrite(IN4_PIN, LOW); //the speed value of motorA is val
 }
 else if (irKeyCode == right) {
 analogWrite(IN1_PIN, LOW);//the speed value of motorA is val
```

Chapitre II :

Fonctionnement et programmes

```
analogWrite(IN2_PIN, val);
 analogWrite(IN3_PIN, LOW);//the speed value of motorA is val
 analogWrite(IN4_PIN, val);
 }
 else if (irKeyCode == down) {
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, val); //the speed value of motorA is val
 analogWrite(IN3_PIN, val); //the speed value of motorA is val
 analogWrite(IN4_PIN, LOW);
 }
 } else {
 analogWrite(IN1_PIN, LOW);
 analogWrite(IN2_PIN, LOW); //the speed value of motorA is 0
 analogWrite(IN3_PIN, LOW);
 analogWrite(IN4_PIN, LOW); //the speed value of motorB is 0
 }
}
```

Chapitre III :

Modules et cartes de la commande gestuelle

III. 1. Introduction

Dans ce chapitre nous allons présenter les différents modules et cartes utilisés pour la réalisation de la commande gestuelle de notre système à véhicule.

III.2. La carte Arduino Nano

La carte Arduino Nano est un produit plus spécifique que les cartes Arduino Uno et Arduino Mega. Compacte, elle est parfaite pour les applications un peu plus petites, mais malgré sa petite taille, elle contient une puissance intéressante pour permettre la construction d'objets intelligents et portables (Fig. III.1).

Figure III.1 : La carte arduino NANO.

Le microprocesseur Arduino Nano est un ATmega328 (Nano3.0) avec une interface USB mini, qui a 14 broches d'entrée / sortie numériques (dont 6 peuvent être utilisées comme sortie PWM), 8 entrées analogiques et 16 MHz résonateur en céramique, 1 connexion mini-B USB, un en-tête ICSP et un bouton de réinitialisation.

III. 2.1 Présentation de RF-NANO

La carte de RF-NANO est intégrée dans la puce NRF24L01+, ce qui lui donne une fonction d'émetteur-récepteur illimitée, ce qui équivaut à combiner une carte NANO ordinaire et un module NRF24L01+ en une seule, comme illustré dans la figure III.2 et III.3. Il est plus pratique à utiliser et le rf-nano de petite taille est exactement le même que le commun broche de carte NANO, pratique pour la transplantation.

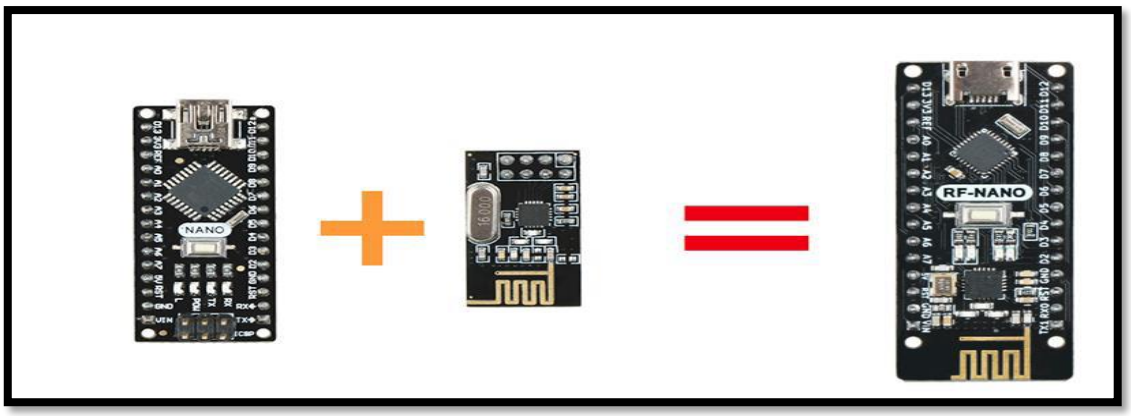


Fig. III.2 : Composition du RF-NANO

III. 2.2. Caractéristiques principales

- Le processeur ATmega328
- Tension de travail 5v
- Tension d'entrée (recommandée) 7-12v
- Tension d'entrée (plage) 6-20 v
- Broche IO numérique 14 (dont 6 peuvent être utilisées comme sortie PWM)
- Broche d'entrée analogique 6
- Broche IO DC 40 mA
- Mémoire flash 16 ou 32 Ko (dont 2 Ko pour le chargeur de démarrage)
- SRAM 1 Ko ou 2 Ko
- EEPROM 0,5 Ko ou 1 Ko （ATmega328）
- Puce USB vers port série CH340
- Horloge 16 MHz

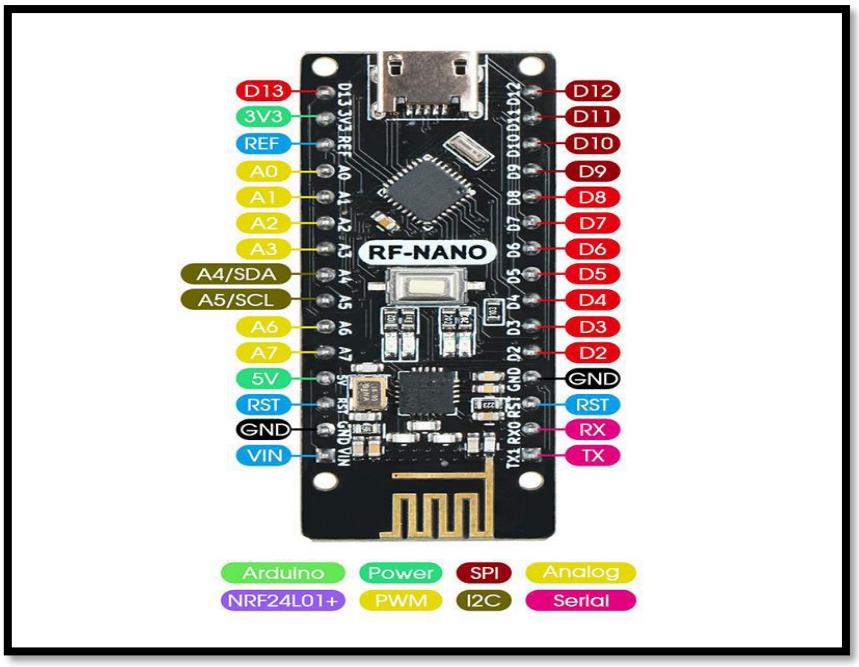


Fig. III.3 : Présentation de la carte RF-NANO

III. 2.2.1 Source de courant

Mode d'alimentation Arduino Nano: alimentation interface mini-B USB et connexion vin externe 7 ~ 12V alimentation DC externe.

III. 2.2.2 Mémoire

ATmega328 inclut une mémoire flash de 32 Ko sur la puce, dont 2 Ko sont utilisés pour Bootloader. Il existe également 2 Ko de SRAM et 1 Ko d'EEPROM.

III. 2.2.3 Entrée et sortie

14 ports d'entrée et de sortie numériques: La tension de fonctionnement est de 5 V, et chaque canal peut produire et accéder au courant maximal de 40 mA. Chaque canal est équipé d'une résistance de rappel interne de 20 à 50 K ohms (non connectée par défaut). De plus, certaines broches ont des fonctions spécifiques.

Signal série RX (0), TX (1): il fournit un signal de réception de port série avec un niveau de tension TTL, connecté à la broche correspondante du FT232Rl.

Interruption externe (n ° 2 et n ° 3): broche d'interruption de déclenchement, qui peut être réglée sur front montant, front descendant ou déclenchement simultané.

Modulation de largeur d'impulsion PWM (3, 5, 6, 9, 10, 11): Fournit des sorties PWM 6 canaux, 8 bits.

SPI 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK): Interface de communication SPI.

LED (n° 13): Arduino est spécialement utilisé pour tester l'interface réservée de la LED. Lorsque la sortie est élevée, la LED est allumée. Lorsque la sortie est faible, la LED est éteinte.

6 entrées analogiques A0 à A5: chaque canal a une résolution de 10 bits (c'est-à-dire que l'entrée a 1024 valeurs différentes), la plage de signal d'entrée par défaut est de 0 à 5 V et la limite supérieure d'entrée peut être ajustée par AREF. De plus, certaines broches ont des fonctions spécifiques.

Interface TWI (SDA A4 et SCL A5): prend en charge l'interface de communication (compatible avec le bus I2C).

AREF: La tension de référence du signal d'entrée analogique.

Réinitialiser: la puce du microcontrôleur est réinitialisée lorsque le signal est faible.

III. 2.2.4 Interface de Communication

Port série: l'UART intégré d'ATmega328 peut communiquer avec le port série externe via les ports numériques 0 (RX) et 1 (TX)

III. 2.2.5 ATmega328 avec NRF24L01+ de communication

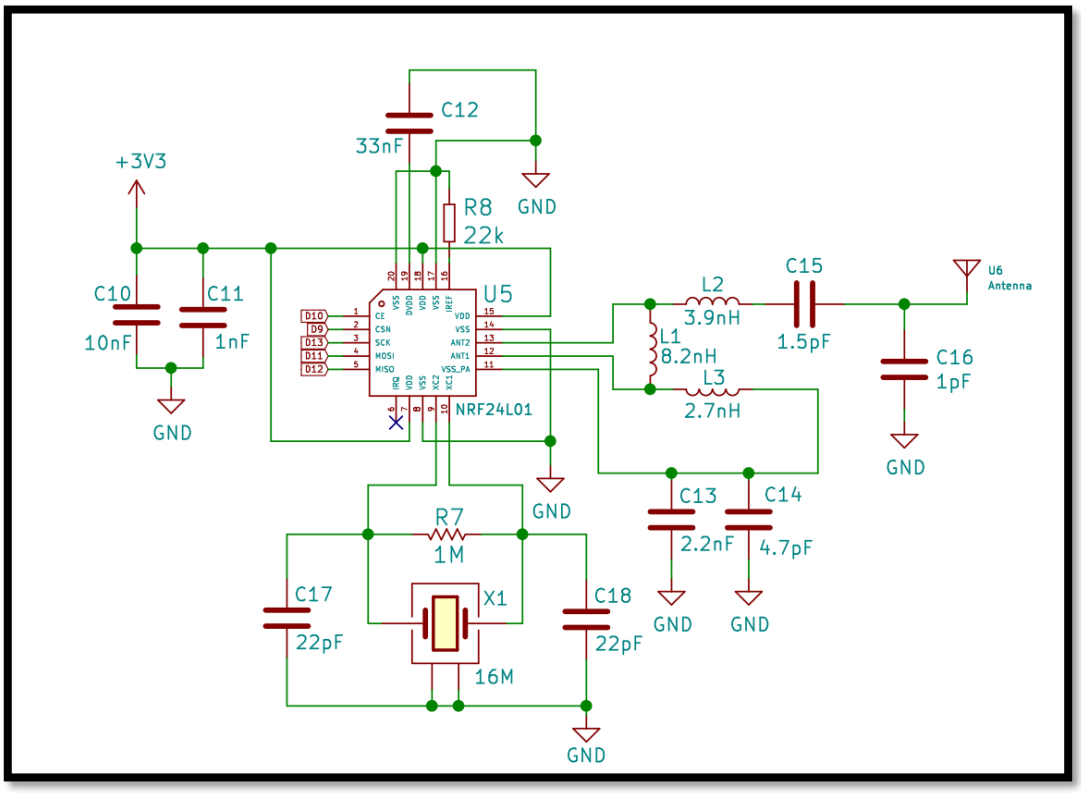


Fig. III.4 : ATmega328 avec NRF24L01+ de communication

III. 2.2.6. Connexion des broches de puce ATmega328 et NRF24L01+

ATmega328	NRF24L01+
+3.3V	VCC
GND	GND
D9	CSN
D10	CE
D11	MOSI
D12	MISO
D13	SCK

Remarque: les broches D9, D10, D11, D12 et D13 d'ATmega328 déjà occupées ne peuvent pas être réutilisées

III. 2.3. Application RF NANO

- Souris sans fil, clavier, joystick.
- Entrée sans clé.
- Communication de données sans fil.
- Systèmes d'alarme et de sécurité.
- Automatisation de la maison.

III. 2.4. Installation du pilote RF-NANO

Le fichier du pilote nommé CH341SER du pilote de la carte Arduino_Nano doit être installé pour que l'ordinateur connaisse la carte Arduino.

III.3. MPU6050

III.3.1. Introduction

MPU6050 est le premier composant de traitement de mouvement à 6 axes au monde avec 3 axes intégrés gyroscope et accélérateur 3 axes.

Il peut se connecter à d'autres capteurs magnétiques ou à d'autres capteurs.

Traitement numérique du mouvement (DMP) via un second port I2C. Le moteur d'accélération matérielle fournit principalement une technique de calcul de fusion à 9 axes complète au MCU hôte sous la forme d'un seul flux de données par le port I2C.

La puce MPU6050 est livrée avec un sous-module de traitement des données DMP, avec un matériel intégré algorithme de filtrage, en utilisant les données de sortie DMP a bien pu répondre aux exigences de nombreuses applications.

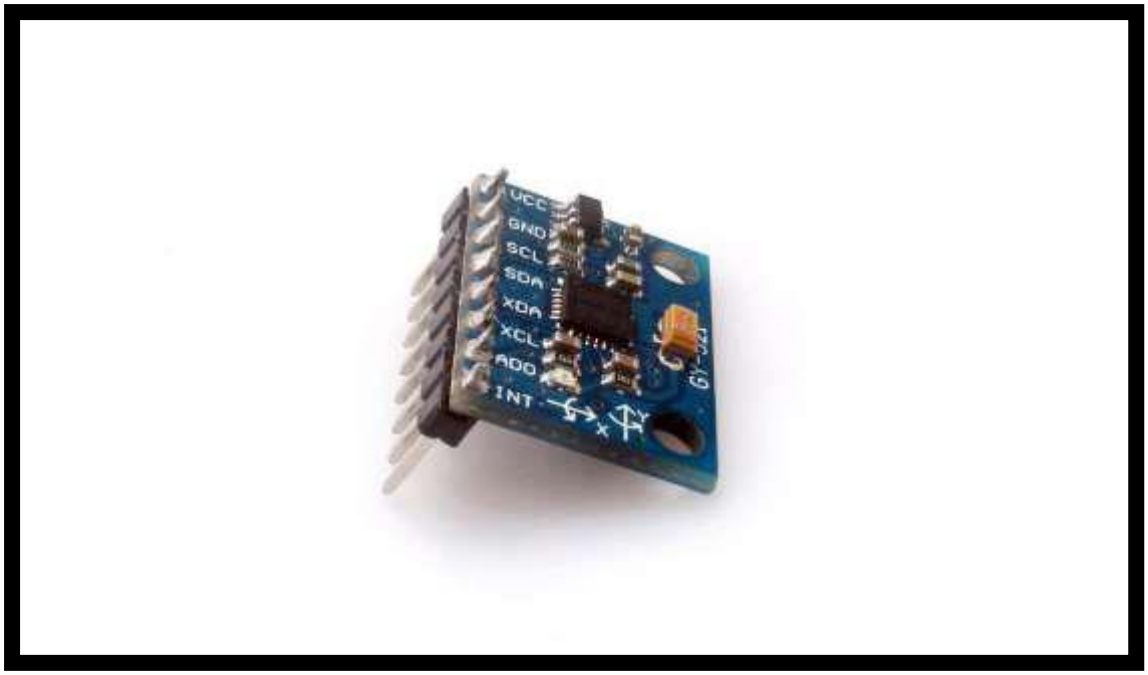


Fig. III.5 : Carte physique du module MPU6050

III.3.2. Caractéristiques

- Les données de calcul intégrées de la sortie numérique de la matrice de rotation 6 axes ou 9 axes, le quaternion et le format d'angle d'Euler.
- Capteur de vitesse angulaire à 3 axes (gyroscope) avec une sensibilité de 131 LSBs / ° / sec et une plage de détection complète de ± 250, ± 500, ± 1000 et ± 2000 ° / sec.
- Commande programmable, accélérateur 3 axes avec plage de contrôle de programme de ± 2g, ± 4g, ± 8g et ± 16g
- Le moteur DMP (Digital Motion Processing) réduit la charge des données de calcul de fusion complexes, la synchronisation des capteurs et la détection des gestes.
- La base de données de traitement des mouvements prend en charge Android, Linux et Windows.
- Les techniques d'étalonnage intégrées pour les écarts de temps de fonctionnement et le capteur magnétique éliminent le besoin supplémentaire d'étalonnage des utilisations.
- Capteur de température de sortie numérique.
- La tension d'alimentation du VDD est de 2,5 V ± 5% 、 3,0 V ± 5% 、 3,3 V ± 5% et le VDDIO est de 1,8 V ± 5%.

- Courant de fonctionnement du gyroscope: 5mA, courant de veille du gyroscope: 8A; courant de fonctionnement de l'accélérateur : 8A, courant du mode d'économie d'énergie de l'accélérateur: 8A @ 10Hz
- Interface I2C en mode rapide jusqu'à 400 kHz ou interface hôte série SPI jusqu'à 20 MHz.
- Emballage sur mesure le plus petit et le plus fin pour le produit portable (QFN 4x4x0,9 mm).

III.3.3. Schéma du module MPU6050

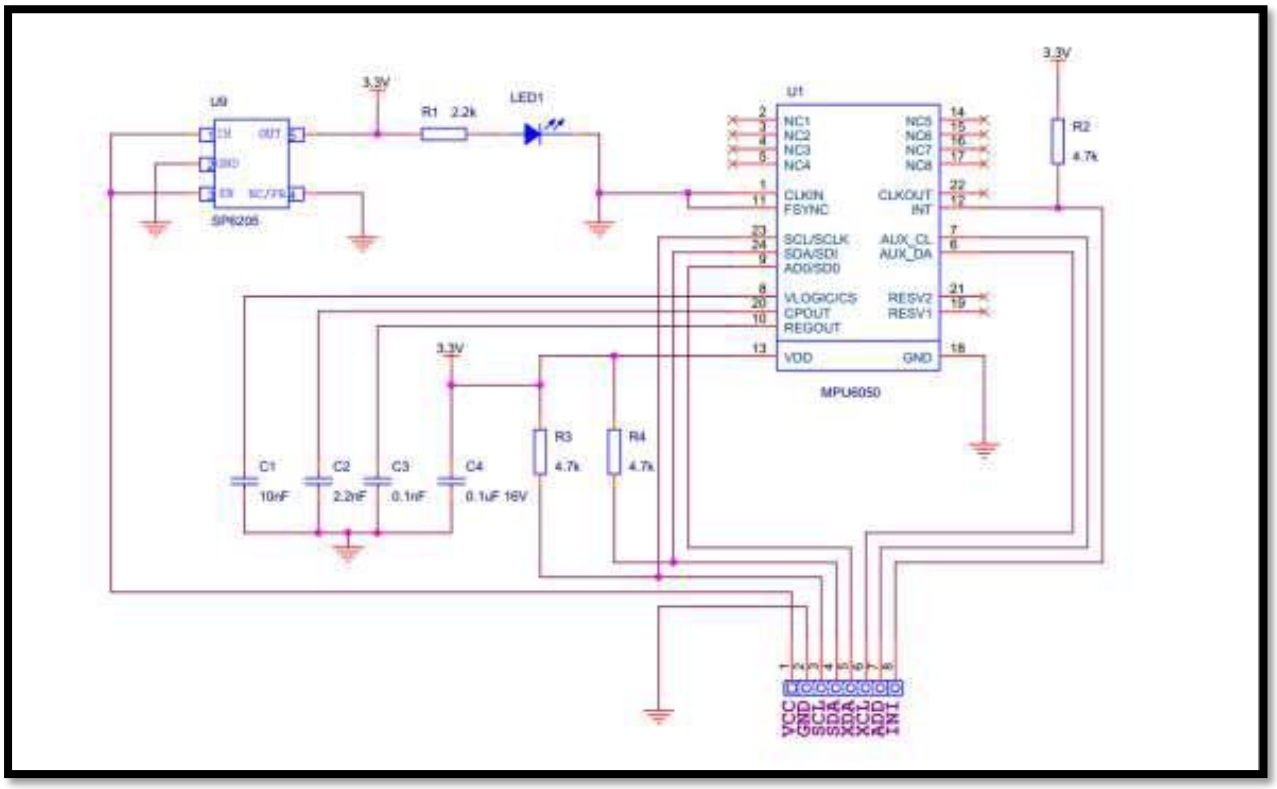


Fig. III.6 : Schéma du module MPU6050

III.3.4. Communication entre Nano et MPU6050

III.3.4.1. Connexion du circuit

L'interface de données du module MPU6050 intégré utilise le protocole de bus I2C, nous avons donc besoin de l'aide de la bibliothèque Wire pour communiquer entre NANO et le MPU6050. La connexion correspondante de la carte NANO est la suivante:

Module MPU6050	Arduino NANO
VCC	5V
GND	GND
SCL	A5
SDA	A4
XDA	NC
XCL	NC
ADD	NC
INT	NC/ GND

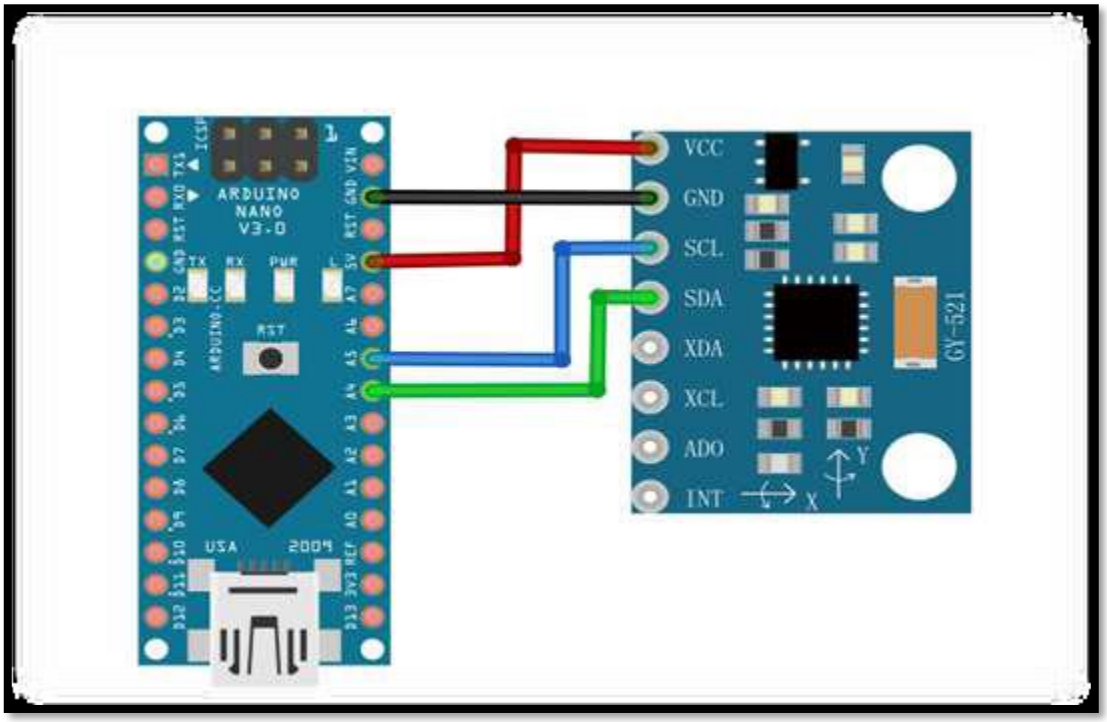


Fig. III.7 : Schéma de connexion du Nano et du MPU6050

III.3.4.2. Analyse des données de mouvement

Après avoir converti les données de lecture de l'accéléromètre et du compteur angulaire en valeurs physiques, les données sont interprétées différemment selon les différentes applications. Dans ce chapitre, le modèle de mouvement de l'avion est pris comme exemple pour calculer l'assiette de vol actuelle en fonction de l'accélération et de la vitesse angulaire.

III.4. Bluetooth HC-05

Le module Bluetooth HC-05 est un module Bluetooth SPP (Serial Port Protocol) facile à utiliser, conçu pour une configuration transparente de la connexion série sans fil. Sa communication se fait via une communication série qui facilite l'interface avec le contrôleur ou le PC. HC-05 Bluetooth Le module fournit un mode de commutation entre le mode maître et le mode esclave, ce qui signifie qu'il est capable de n'utiliser ni recevoir ni transmettre de données.

III.4.1. Spécification

- Modèle: HC-05.
- Tension d'entrée: DC 5V.
- Méthode de communication: communication série.

Les modes maître et esclave peuvent être commutés.

III.4.2. Définition des broches

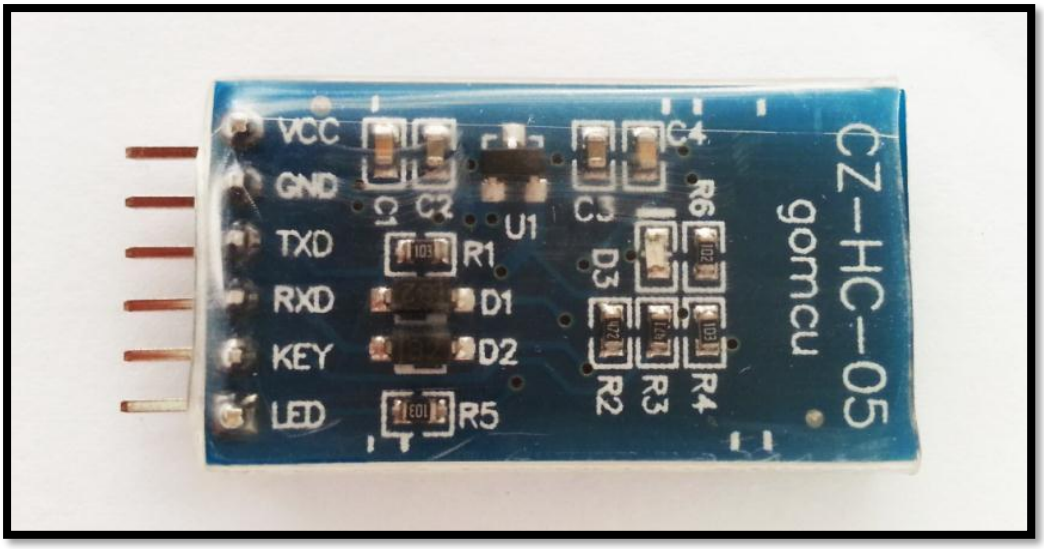


Fig. III.8 : Carte physique du module Bluetooth HC-05

Pin	Description	Function
VCC	+5V	Connect to +5V
GND	Ground	Connect to Ground
TXD	UART_TXD, Bluetooth serial signal sending PIN	Connect with the MCU's (Microcontroller and etc) RXD PIN.
RXD	UART_RXD, Bluetooth serial signal receiving PIN	Connect with the MCU's (Microcontroller and etc) TXD PIN.
KEY	Mode switch input	If it is input low level or connect to the air, the module is at paired or communication mode. If it's input high level, the module will enter to AT mode.

Chapitre IV :

Tests et commande gestuelle

IV.1. Introduction

L'un des robots contrôlés par mouvement fréquemment mis en œuvre est un robot contrôlé par les gestes de la main. Dans ce projet, un robot contrôlé par les gestes de la main est développé à l'aide du MPU6050, qui est un accéléromètre à 3 axes et un capteur de gyroscope à 3 axes et la partie contrôleur est RF- Nano.

Au lieu d'utiliser une télécommande avec des boutons ou un joystick, les gestes de la main sont utilisés pour contrôler le mouvement du robot.

Le projet est basé sur la communication sans fil, où les données des gestes de la main sont transmises au robot via les puce nRF24L01.

Le projet est divisé en sections d'envoi et de réception. Le schéma de circuit et les composants sont expliqués séparément pour les sections émetteur et récepteur, et ici nous expliquerons le circuit de commande qui se compose de la partie émetteur pour les commandes via la connexion sans fil.

IV.2. DESCRIPTION TEST LES MOTOR

Nous testons les moteurs de la voiture séparément dans toutes les directions en écrivant chaque code séparément pour une réponse optimale.

IV.2.1. Code les tests Motors

IV. 2.1.1.Test moteur à l'avant

Nous écrivons un programme simple pour tester un moteur qui avance pendant 5 secondes et s'arrête.

```
#define IN1_PIN 6
#define IN2_PIN 10
#define IN3_PIN 5
#define IN4_PIN 9

void setup() {
  Serial.begin(9600);
  pinMode(IN1_PIN, OUTPUT);
  digitalWrite(IN1_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN2_PIN, OUTPUT);
  digitalWrite(IN2_PIN, LOW); // Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN3_PIN, OUTPUT);
digitalWrite(IN3_PIN, LOW); // Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN4_PIN, OUTPUT);
  digitalWrite(IN4_PIN, LOW); // Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
}
void loop() {
  analogWrite(IN1_PIN, 100);
  analogWrite(IN2_PIN, LOW);
  analogWrite(IN3_PIN, LOW);
  analogWrite(IN4_PIN, 100);
  delay(5000);
}
  //******** ******************************//AVANT
```

IV.2.1.2. Test moteur à l'arrêt

Code de stationnement du véhicule

```
#define IN1_PIN 6

#define IN2_PIN 10
#define IN3_PIN 5
#define IN4_PIN 9

void setup() {
  Serial.begin(9600);
  pinMode(IN1_PIN, OUTPUT);
  digitalWrite(IN1_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN2_PIN, OUTPUT);
  digitalWrite(IN2_PIN, LOW); // Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN3_PIN, OUTPUT);
  digitalWrite(IN3_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN4_PIN, OUTPUT);
  digitalWrite(IN4_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
}
void loop() {
  analogWrite(IN1_PIN, HIGH);
  analogWrite(IN2_PIN, HIGH);
  analogWrite(IN3_PIN, HIGH);
  analogWrite(IN4_PIN, HIGH);
  delay(1000);//****************************************// arret
}
```

IV.2.1.3. Test moteur à gauche

Le véhicule tourne à gauche pendant deux secondes, puis s'arrête

```
#define IN1_PIN 6

#define IN2_PIN 10
#define IN3_PIN 5
#define IN4_PIN 9

void setup() {
  Serial.begin(9600);
  pinMode(IN1_PIN, OUTPUT);
  digitalWrite(IN1_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN2_PIN, OUTPUT);
  digitalWrite(IN2_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN3_PIN, OUTPUT);
  digitalWrite(IN3_PIN, LOW); // Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN4_PIN, OUTPUT);
  digitalWrite(IN4_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
}
void loop() {
 analogWrite(IN1_PIN, 200);
  analogWrite(IN2_PIN, LOW);
  analogWrite(IN3_PIN, 200);
  analogWrite(IN4_PIN, LOW);
  delay(3000);
  //*****************************************//GAUCHE
}
```

IV.2.1.4. Test moteur à droit

Le véhicule tourne à droite pendant deux secondes, puis s'arrête.

```
#define IN1_PIN 6

#define IN2_PIN 10
#define IN3_PIN 5
#define IN4_PIN 9

void setup() {
  Serial.begin(9600);
  pinMode(IN1_PIN, OUTPUT);
  digitalWrite(IN1_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN2_PIN, OUTPUT);
  digitalWrite(IN2_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN3_PIN, OUTPUT);
  digitalWrite(IN3_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN4_PIN, OUTPUT);
  digitalWrite(IN4_PIN, LOW); // Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
}
void loop() {
 analogWrite(IN1_PIN, LOW);
  analogWrite(IN2_PIN, 200);
  analogWrite(IN3_PIN, LOW);
  analogWrite(IN4_PIN, 200);
  delay(3000);//*** ************************************//adroit
```

IV.2.1.5. Test moteur arrière

Le véhicule reculera pendant 5 secondes et s'arrêtera

```
#define IN1_PIN 6

#define IN2_PIN 10
#define IN3_PIN 5
#define IN4_PIN 9

void setup() {
  Serial.begin(9600);
  pinMode(IN1_PIN, OUTPUT);
  digitalWrite(IN1_PIN, LOW); // Quand nous n'envoyons pas PWM, nous
voulons qu'il soit bas
  pinMode(IN2_PIN, OUTPUT);
  digitalWrite(IN2_PIN, LOW); //Quand nous n'envoyons pas PWM, nous voulons
qu'il soit bas
  pinMode(IN3_PIN, OUTPUT);
  digitalWrite(IN3_PIN, LOW); // Quand nous n'envoyons pas PWM, nous
voulons qu'il soit bas
  pinMode(IN4_PIN, OUTPUT);
  digitalWrite(IN4_PIN, LOW); // Quand nous n'envoyons pas PWM, nous
voulons qu'il soit bas
}
void loop() {
   analogWrite(IN1_PIN, LOW);
  analogWrite(IN2_PIN, 100);
  analogWrite(IN3_PIN, 100);
  analogWrite(IN4_PIN, LOW);
  delay(5000);//*******************************************//ARRIÈRE
```

Nous testons l'unité à ultrasons en écrivant le programme d'évitement d'obstacles qui se présente comme suit, lorsque l'échographie atteint une distance avant supérieure à 20, elle se poursuit.

Lorsqu'elle est inférieure à 20, elle permettra à l'échographie de mesurer les distances gauche et droite.

Lorsque vous aurez plus de 12 ans, vous vous tournerez de chaque côté, puis vous continuerez à mesurer la distance avant et à faire des jugements, puis le Hummerbot sera autorisé à prendre les mesures appropriées.

IV.3.1. Code ultrasons

```
#include "Hummerbot.h"
#include "BluetoothHandle.h"
#include "ProtocolParser.h"
#include "KeyMap.h"
#include "debug.h"

#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN3_PIN 5   // DIRA  ---  left
#define IN4_PIN 9   // PWMA

#define SERVO_PIN 13
#define UL_SING_PIN 3
#define UL_RGB_PIN 2

ProtocolParser *mProtocol = new ProtocolParser();
Hummerbot hbot(mProtocol, IN1_PIN, IN2_PIN, IN3_PIN, IN4_PIN);

void setup()
{
  Serial.begin(9600);
  hbot.init();

hbot.SetControlMode(E_ULTRASONIC_AVOIDANCE);//E_BLUETOOTH_CONTROL/E_INFRARED_TRACK
ING_MODE
  hbot.SetRgbUltrasonicPin(UL_SING_PIN, UL_RGB_PIN, SERVO_PIN);
  hbot.SetSpeed(0);
  hbot.mRgbUltrasonic->SetServoBaseDegree(90);
  hbot.mRgbUltrasonic->SetServoDegree(90);

}

void HandleUltrasonicAvoidance(void)
{
  uint16_t UlFrontDistance, UlLeftDistance, UlRightDistance;
  UlFrontDistance =  hbot.GetUltrasonicValue(FRONT);
  DEBUG_LOG(DEBUG_LEVEL_INFO, "UlFrontDistance =%d \n", UlFrontDistance);
   Serial.println(UlFrontDistance);
   if ((UlFrontDistance < UL_LIMIT_MIN))
  {
    hbot.SetSpeed(100);
    hbot.GoBack();
    delay(250);
  }
 else if (UlFrontDistance < UL_LIMIT_MID)
  {
    hbot.KeepStop();
    delay(100);
    UlLeftDistance = hbot.GetUltrasonicValue(LEFT);
    UlRightDistance = hbot.GetUltrasonicValue(RIGHT);
    if ((UlRightDistance > UL_LIMIT_MIN) && (UlRightDistance < UL_LIMIT_MAX) &&
(UlLeftDistance > UL_LIMIT_MIN) && (UlLeftDistance < UL_LIMIT_MAX))
    {
      if (UlRightDistance > UlLeftDistance)
      {
        hbot.SetSpeed(80);
        hbot.TurnRight();
        delay(310);
      }
```

```
else

      {
        hbot.SetSpeed(80);
        hbot.TurnLeft();
        delay(310);
      }
    }
    else if (((UlRightDistance > UL_LIMIT_MIN) && (UlRightDistance <
UL_LIMIT_MAX)) || ((UlLeftDistance > UL_LIMIT_MIN) && (UlLeftDistance <
UL_LIMIT_MAX)))
    {
      if ((UlLeftDistance > UL_LIMIT_MIN) && (UlLeftDistance < UL_LIMIT_MAX))
      {
        hbot.SetSpeed(80);
        hbot.TurnLeft();
        delay(310);
      }
      else if ((UlRightDistance > UL_LIMIT_MIN) && (UlRightDistance <
UL_LIMIT_MAX))
      {
        hbot.SetSpeed(80);
        hbot.TurnRight();
        delay(310);
      }
    }
    else if ((UlRightDistance < UL_LIMIT_MIN) && (UlLeftDistance <
UL_LIMIT_MIN) )
    {
      hbot.SetSpeed(80);
      hbot.Drive(0);
      delay(510);
    }
  }
  else
  {
    hbot.SetSpeed(80);
    hbot.GoForward();
  }
}

void loop()
{
  switch (hbot.GetControlMode()) {
    case E_ULTRASONIC_AVOIDANCE:
      DEBUG_LOG(DEBUG_LEVEL_INFO, "E_ULTRASONIC_AVOIDANCE \n");
      HandleUltrasonicAvoidance();
      break;
    default:
      break;
  }
```

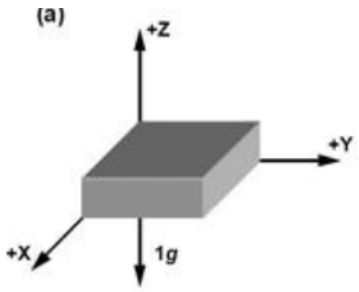
```
switch (hbot.GetStatus()) {

    case E_FORWARD:
      hbot.mRgbUltrasonic->SetRgbColor(E_RGB_ALL, RGB_WHITE);
      break;
    case E_LEFT:
      hbot.mRgbUltrasonic->SetRgbColor(E_RGB_LEFT, RGB_WHITE);
      break;
    case E_RIGHT:
      hbot.mRgbUltrasonic->SetRgbColor(E_RGB_RIGHT, RGB_WHITE);
      //   Mirage.Sing(S_OhOoh);
      break;
    case E_BACK:
      hbot.mRgbUltrasonic->SetRgbColor(E_RGB_ALL, RGB_RED);
      break;
    case E_STOP:
      hbot.mRgbUltrasonic->SetRgbColor(E_RGB_ALL, RGB_BLACK);
      break;
    case E_SPEED_UP:
      hbot.mRgbUltrasonic->SetRgbColor(E_RGB_ALL, hbot.GetSpeed() *
2.5);
      break;
    case E_SPEED_DOWN:
      hbot.mRgbUltrasonic->SetRgbColor(E_RGB_ALL, hbot.GetSpeed() *
2.5);
      break;
    default:
      break;
  }
}
```

IV.4. MODULES

L'accéléromètre est conservé dans la paume de l'utilisateur et le robot se déplace par étapes avec le mouvement de la paume. Dans cet article nous avons expliqué le rôle de 5 gestes distinctifs des personnes main c.-à-d. état d'arrêt, mouvement avant, mouvement arrière et tourne à droite et à gauche.

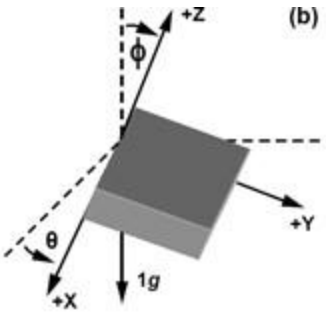
IV.4.1. Condition d'arrêt

L'utilisateur tient l'accéléromètre parallèlement au sol. A ce moment, le signal de l'accéléromètre est envoyé à l'Arduino et le robot cesse de bouger. Cet état est appelée ici condition d'arrêt.



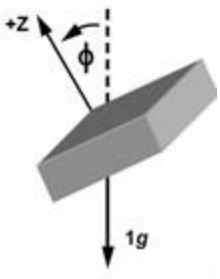
IV.4.2. Inclinaison vers l'avant

L'utilisateur tient l'accéléromètre et l'accéléromètre incliné vers l'avant, l'axe x, y, z est envoyé à l'Arduino. Si les axes x, y, z remplissent la condition x> - * 250, y> = 20, z> = 0, le robot se déplace vers l'avant.



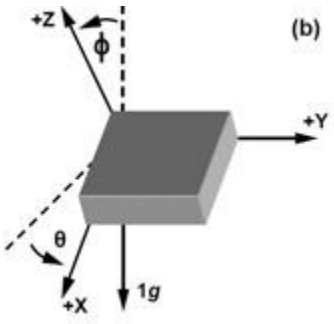
IV.4.3. Inclinaison vers l'arrière

L'utilisateur tient l'accéléromètre et l'accéléromètre incliné vers l'arrière, l'axe x, y, z est envoyé à l'Arduino. Si les axes x, y, z remplissent la condition x> = 0 et x <= 20, y <= 60 et y> = 20, z> = 150, le robot recule.



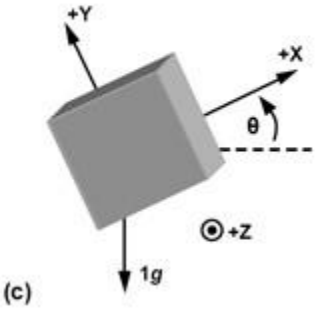
IV.4.4. Inclinaison à droite

L'utilisateur tient l'accéléromètre et l'accéléromètre incliné vers la droite, l'axe x, y, z est envoyé à l'Arduino. Si les axes x, y, z remplissent la condition x> = 0 et x <= 20, y> = 250, y> = 60, z> = 1, le robot se déplace vers la droite.



IV.4.5. Inclinaison gauche

L'utilisateur tient l'accéléromètre et l'accéléromètre incliné vers la gauche, l'axe x, y, z est envoyé à l'Arduino. Si les axes x, y, z remplissent la condition x> = 0, x <= 20, y> = - 3, y> = - 250, z> = 1, le robot se déplace vers la gauche.



IV.5. Description de la commande gestuelle

Ici, nous utilisons l'accéléromètre pour transférer la valeur de l'axe vers l'Arduino. L'axe est ajusté selon

L'inclinaison de la main de l'utilisateur et la valeur de pivotement sont fréquemment envoyées à Arduino.

- Si l'utilisateur fait tourner l'accéléromètre vers l'avant en même temps, le robot avance. Si l'utilisateur fait pivoter l'accéléromètre vers l'arrière, le robot recule en même temps.
- Si l'utilisateur tourne l'accéléromètre vers la gauche, le robot tourne simultanément vers la gauche. Si l'utilisateur déplace l'accéléromètre vers la droite, le robot tourne simultanément vers la droite.
- Si l'utilisateur maintient l'accéléromètre parallèle au sol, le robot cesse de bouger en même temps. Le mouvement du robot dépend de l'inclinaison de l'accéléromètre et de la connexion radio

IV.5.1. Code émetteur et récepteur

IV.5.1.1. Voiture de contrôle de code

```
/void ForWard()

//for Gesture controled Robotic Car
int lm1=5; //left motor output 1
int lm2=6; //left motor output 2
int rm1=9;  //right motor output 1
int rm2=10;  //right motor output 2
char d=0;
void setup()
{
pinMode(lm1,OUTPUT);
pinMode(lm2,OUTPUT);
pinMode(rm1,OUTPUT);
pinMode(rm2,OUTPUT);
Serial.begin(38400);
sTOP();
}
void loop()
{
if(Serial.available()>0)
{
d=Serial.read();
if(d=='F')
{
ForWard();
}
if(d=='B')
{
BackWard();
}
if(d=='L')
{
Left();
}
if(d=='R')
{
Right();
}
if(d=='S')
{
sTOP();
}
}
}
```

Code : émetteur/récepteur

```
/void ForWard()

{
digitalWrite(lm1,HIGH);
digitalWrite(lm2,LOW);
digitalWrite(rm1,HIGH);
digitalWrite(rm2,LOW);
}
void BackWard()
{
digitalWrite(lm1,LOW);
digitalWrite(lm2,HIGH);
digitalWrite(rm1,LOW);
digitalWrite(rm2,HIGH);
}
void Left()
{
digitalWrite(lm1,LOW);
digitalWrite(lm2,HIGH);
digitalWrite(rm1,HIGH);
digitalWrite(rm2,LOW);
}
void Right()
{
digitalWrite(lm1,HIGH);
digitalWrite(lm2,LOW);
digitalWrite(rm1,LOW);
digitalWrite(rm2,HIGH);
}
void sTOP()
{
digitalWrite(lm1,LOW);
digitalWrite(lm2,LOW);
digitalWrite(rm1,LOW);
digitalWrite(rm2,LOW);
}
```

IV.5.1.2. Remote contrôle

```
//program modified on 3/10/19 by // by Shubham Shinganapure.
//
//for Gesture controled Robotic Car (remote  )

#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"
//#include "MPU6050.h" // not necessary if using MotionApps include file

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for SparkFun breakout and InvenSense evaluation
board)
// AD0 high = 0x69
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 =
success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

VectorFloat gravity;
Quaternion q;
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container
and gravity vector

uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00,
'\r', '\n' };

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt
pin has gone high
void dmpDataReady() {
    mpuInterrupt = true;
}
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(2, 3); // RX | TX

int bt=8;
int x =1;
void setup() {
```

```
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE

        Wire.begin();
        TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (115200 chosen because it is required for Teapot Demo output, but it's
    // really up to you depending on your project)
    Serial.begin(115200);
    BTSerial.begin(38400);
   // while (!Serial); // wait for Leonardo enumeration, others continue
immediately

Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();

    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") :
F("MPU6050 connection failed"));

    // wait for ready

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

 // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788);

    // make sure it worked (returns 0 if so)
    if (devStatus == 0) {
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.println(F("Enabling interrupt detection (Arduino external
interrupt 0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();
```

```
  // set our DMP Ready flag so the main loop() function knows it's okay to use
it
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }

    // configure LED for output

  pinMode(bt,INPUT);

}



// ================================================================
// ===                      MAIN PROGRAM LOOP                   ===
// ================================================================

void loop() {
 if(digitalRead(bt)==HIGH)
 {
  x++;
  delay(150);
  }
  if((x%2)==0){
    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize) {
        // other program behavior stuff here
        // .
        // .
        // .
        // if you are really paranoid you can frequently test in between other
        // stuff to see if mpuInterrupt is true, and if so, "break;" from the
        // while() loop to immediately process the MPU data
        // .
        // .
        // .
    }

    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();
```

```
// check for overflow (this should never happen unless our code is too
inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));

    // otherwise, check for DMP data ready interrupt (this should
happen frequently)
    } else if (mpuIntStatus & 0x02) {
        // wait for correct available data length, should be a VERY
short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an
interrupt)
        fifoCount -= packetSize;

 #ifdef OUTPUT_READABLE_YAWPITCHROLL
            // display Euler angles in degrees
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
            Serial.print("ypr\t");
            Serial.print(ypr[0] * 180/M_PI);
            Serial.print("\t");
            Serial.print(ypr[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(ypr[2] * 180/M_PI);
            if((ypr[1] * 180/M_PI)<= -25)
            {BTSerial.write('F');
            }
            else if((ypr[1] * 180/M_PI)>= 25)
            {BTSerial.write('B');
            }
            else if((ypr[2] * 180/M_PI)<= -25)
            {BTSerial.write('L');
            }
            else if((ypr[2] * 180/M_PI)>= 20)
            {BTSerial.write('R');
            }
            else{
              BTSerial.write('S');
              }


        #endif

}
  }
  else{
              BTSerial.write('S');
              }
}
```

Conclusion générale

Grâce à ce projet, nous avons réussi à atteindre notre objectif d'être gouverné dans le robot de la voiture par des gestes manuels basés sur des connexions sans fil en utilisant la technologie d'accéléromètre. Les applications sont nombreuses et comprennent les systèmes d'aviation, téléphonie, commande de robot, et plusieurs autres applications.

Notre objectif principal de notre travail est de contrôler le robot de la voiture avec des gestes de la main. Nous avons fait d'autres applications telles que l'évitement d'obstacles en utilisant des ondes ultrasoniques et infrarouges et le contrôle du robot de la voiture en utilisant la télécommande qui a été couronnée de succès.

Les chapitres 1 et 2 fournissent des informations sur les composants et autres concepts liés à l'utilisation du robot du véhicule.

Les chapitres 3 et 4 fournissent des informations sur les composants utilisés dans le circuit de commande et les codes de programmation.

Bibliographie

[1] WalidBenlahcene : Un éclairageredondant, mémoire de fin d'étude pour d'ingénieur d'état en instrumentation, UniversitéBatna, (2007).

[2] Mr. DJAFRI MENAD, Mr. CHELOUCHE Djalal, Etude et Réalisation d'une Carte Arduino ; Mémoire de fin d'études pour l'obtention du diplôme de MASTER université A.MIRA DE BEJAIA; 2016.

[3] https://github.com/emakefun/hummer-bot/