III.1 Introduction

Pour faire varier la vitesse d'un moteur à courant continu, on varier la tension d'alimentation à ses bornes. Pour atteindre cet objectif on utilise un hacheur, ce dernier commande la variation de la tension moyenne aux bornes du moteur et par la suite la variation de la vitesse de rotation. On parle alors de Modulation par Largeur d'Impulsions (MLI).

Ce chapitre pris en charge la simulation et la réalisation du circuit permettant le pilotage du hacheur quatre quadrants dans le but de varier la vitesse et le sens du moteur à courant continu.

III.2 Partie de simulation :

Pour avoir les résultats attendus par la voie de simulation, nous allons d'abord projeter la lumière sur la carte d'acquisition qui est l'élément le plus important dans ce projet fin d'étude (‘Arduino’). Nous allons également expliquer le principe de fonctionnement de la carte Arduino sans oublier ses caractéristiques.

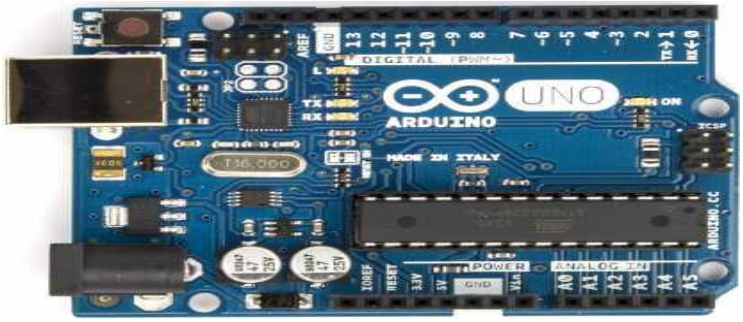
III.3 Arduino :

Le module Arduino est un circuit imprimé en matériel libre (plateforme de contrôle) dont les plans de la carte elle-même sont publiés en licence libre dont certains composants de la carte: comme le microcontrôleur et les composants complémentaires qui ne sont pas en licence libre. Un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diverses. Arduino est utilisé dans beaucoup d'applications comme l'électrotechnique industrielle et embarquée le modélisme, la domotique mais aussi dans des domaines différents comme l'art contemporain et le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur, commander des appareils mobiles (modélisme). Chaque module d'Arduino possède un régulateur de tension +5V et un oscillateur à quartez 16 MHz (ou un résonateur céramique dans certains modèles). Pour programmer cette carte, on utilise l'logiciel IDE Arduino [13].

Parmi ces types, nous avons choisi une carte Arduino UNO (carte Basique). L'intérêtprincipal de cette carte est de faciliter la mise en oeuvre d’une telle commande qui seradétaillée par la suite.

L'Arduino fournit un environnement de développement s'appuyant sur des outils open source comme interface de programmation. L'injection du programme déjà converti par l’environnement sous forme d’un code "HEX" dans la mémoire du microcontrôleur se fait d'une façon très simple par la liaison USB. En outre, des bibliothèques de fonctions "clé en

main" sont également fournies pour l'exploitation d'entrées-sorties. Cette carte est basée sur un microcontrôleur ATmega 328 et des composants complémentaires. La carte Arduino contient une mémoire morte de 1 kilo. Elle est dotée de 14 entrées/sorties digitales (dont 6 peuvent être utilisées en tant que sortie PWM), 6 entrées analogiques et un cristal à 16 MHz, une connexion USB et Possède un bouton de remise à zéro et une prise jack d'alimentation. La carte est illustrée dans la figure si dessous.



Figure(III.1): La carte Arduino UNO.

III.3.1 La constitution de la carte Arduino UNO :

Un module Arduino est généralement construit autour d'un microcontrôleur ATMELAVR, et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits. Chaque module possède au moins un régulateur linéaire 5V et un oscillateur à quartz 16 MHz (ou un résonateur céramique dans certains modèles). Le microcontrôleur est préprogrammé avec un boot loader de façon à ce qu'un programmateur dédié ne soit pas nécessaire.

III.3.1.1 Partie matérielle (Hard) :

Généralement tout module électronique qui possède une interface de programmation est basé toujours dans sa construction sur un circuit programmable ou plus.

III.3.1.1.1 Le Microcontrôleur ATMega328 :

Un microcontrôleur ATMega328 est un circuit intégré qui rassemble sur une puce plusieurs éléments complexes dans un espace réduit au temps des pionniers de l'électronique. Aujourd'hui, en soudant un grand nombre de composants encombrants, tels que les transistors; les résistances et les condensateurs tout peut être logé dans un petit boîtier en plastique noir muni d'un certain nombre de broches dont la programmation peut être réalisée en langage C. la figure I.2 montre un microcontrôleur ATmega 328, qu'on trouve sur la carte Arduino [14].

Le composant CMS Le composant classique

Figure (III.2): Microcontrôleur ATMega328.

Le microcontrôleur ATMega328 est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur. Globalement, l'architecture interne de ce circuit programmable se compose essentiellement sur:

- **La mémoire Flash**: C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible mémoire programme de 32Ko (dont boot loader de 0.5 ko).
- **RAM:** c'est la mémoire dite "vive", elle va contenir les variables du programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur. Sa capacité est 2 ko.
- **EEPROM**: C'est le disque dur du microcontrôleur. On y enregistre des infos qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme. [15][16]

III.3.1.1.2 Les sources de l'alimentation de la carte :

On peut distinguer deux genres de sources d'alimentation (entrée-sortie) et cela comme suit:

- **V_{IN}**. La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). On peut alimenter la carte à l'aide de cette broche, ou, si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.
- **5V**. La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (pour info : les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V

régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation V_{IN} via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.

- **3V3**. Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de votre ordinateur et le port série de l'ATmega) de la carte est disponible : ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V. L'intensité maximale disponible sur cette broche est de 50mA. [13][16]

III.3.1.1.3 Les entrées & sorties :

Cette carte possède 14 broches numériques (numérotée de 0 à 13) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions pinMode ( ), digitalWrite ( ) et digitalRead ( ) du langage Arduino. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40 mA d'intensité et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de 20-50 KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digital Write (broche, HIGH).

En plus, certaines broches ont des fonctions spécialisées :

- **Interruptions Externes**: Broches 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur. -Impulsion PWM (largeur d'impulsion modulée): Broches 3, 5, 6, 9, 10, et 11. Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analog Write ( ).
- **SPI** (Interface Série Périphérique): Broches 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches supportent la communication SPI (Interface Série Périphérique) disponible avec la librairie pour communication SPI. Les broches SPI sont également connectées sur le connecteur ICSP qui est mécaniquement compatible avec les cartes Mega.
- **I2C**: Broches 4 (SDA) et 5 (SCL). Supportent les communications de protocole I2C (ou interface TWI (Two Wire Interface - Interface "2 fils"), disponible en utilisant la librairie Wire/I2C (ou TWI - Two-Wire interface - interface "2 fils").
- **LED**: Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13.Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

La carte UNO dispose 6 entrées analogiques (numérotées de 0 à 5), chacune pouvant fournir une mesure d'une résolution de 10 bits (càd sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead ( ) du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction analog Reference ( ) du langage Arduino. La carte Arduino UNO intègre un fusible qui protège le port USB de l'ordinateur contre les surcharges en intensité (le port USB est généralement limité à 500 mA en intensité). Bien que la plupart des ordinateurs aient leur propre protection interne, le fusible de la carte fournit une couche supplémentaire de protection. Si plus de 500mA sont appliqués au port USB, le fusible de la carte coupera automatiquement la connexion jusqu'à ce que le court-circuit ou la surcharge soit stoppé.[17]

III.3.1.1.4 Les ports de communications :

La carte Arduino UNO a de nombreuses possibilités de communications avec l'extérieur. L'Atmega328 possède une communication série UART TTL (5V), grâce aux broches numériques 0 (RX) et 1 (TX). On utilise (RX) pour recevoir et (TX) transmettre (les données séries de niveau TTL).Ces broches sont connectées aux broches correspondantes du circuit intégré ATmega328. Programmé en convertisseur USB – vers – série de la carte, composant qui assure l'interface entre les niveaux TTL et le port USB de l'ordinateur. Comme un port de communication virtuel pour le logiciel sur l'ordinateur, La connexion série de l'Arduino est très pratique pour communiquer avec un PC, mais son inconvénient est le câble USB, pour éviter cela, il existe différentes méthodes pour utiliser ce dernier sans fil:

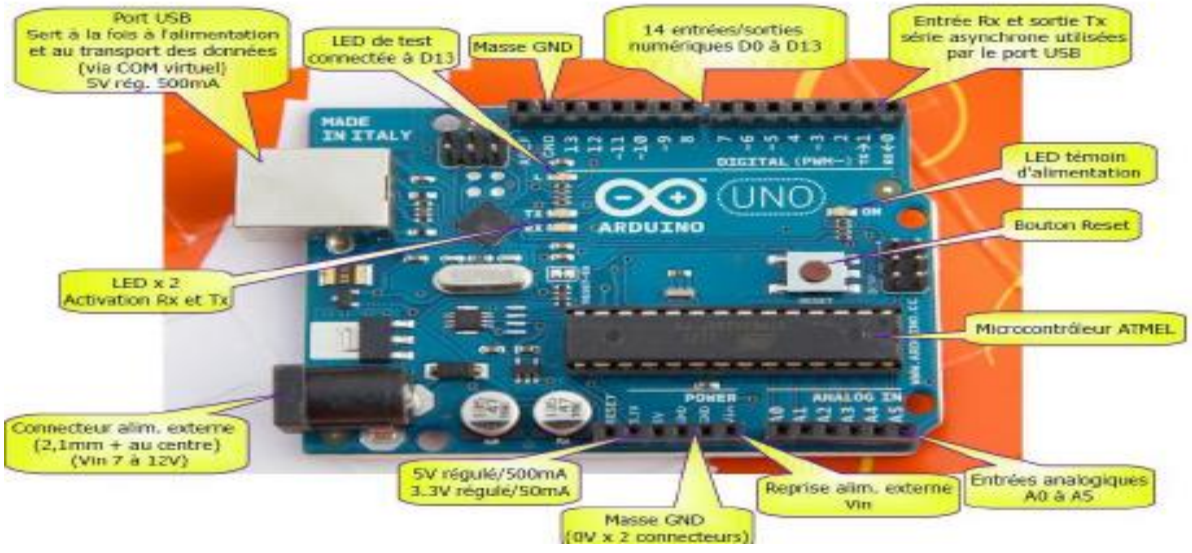


Figure (III.3): Constitution de la carte Arduino UNO.

III.3.1.2 Partie programme (Soft) :

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

III.3.1.2.1 L'environnement de la programmation :

Le logiciel de programmation de la carte Arduino sert d'éditeur de code (langage proche du C). Une fois, le programme tapé ou modifié au clavier, il sera transféré et mémorisé dans la carte au travers de la liaison USB. Le câble USB alimente à la fois en énergie la carte et transporte aussi l'information CE programme appelé IDE Arduino [16][18].

III.3.1.2.2 Structure générale du programme (IDE Arduino) :

Comme n'importe quel langage de programmation, une interface souple et simple est exécutable sur n'importe quel système d'exploitation Arduino basé sur la programmation en C.

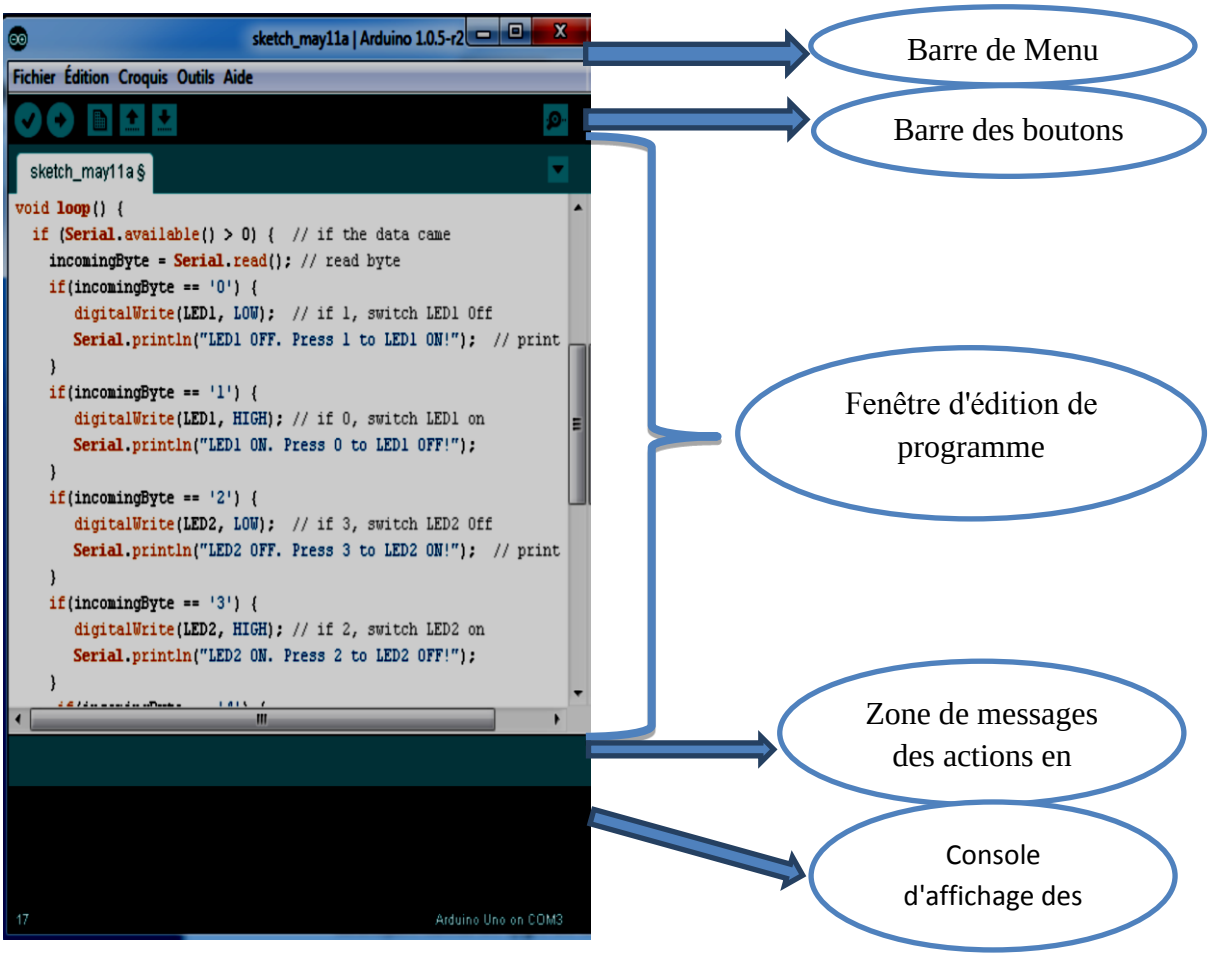


Figure (III.4): Interface IDE Arduino.

III.3.1.2.3 Injection du programme :

Avant d'envoyer un programme dans la carte, il est nécessaire de sélectionner le type de la carte (Arduino Uno) et le numéro de port USB (COM) comme à titre d'exemple cette figure.

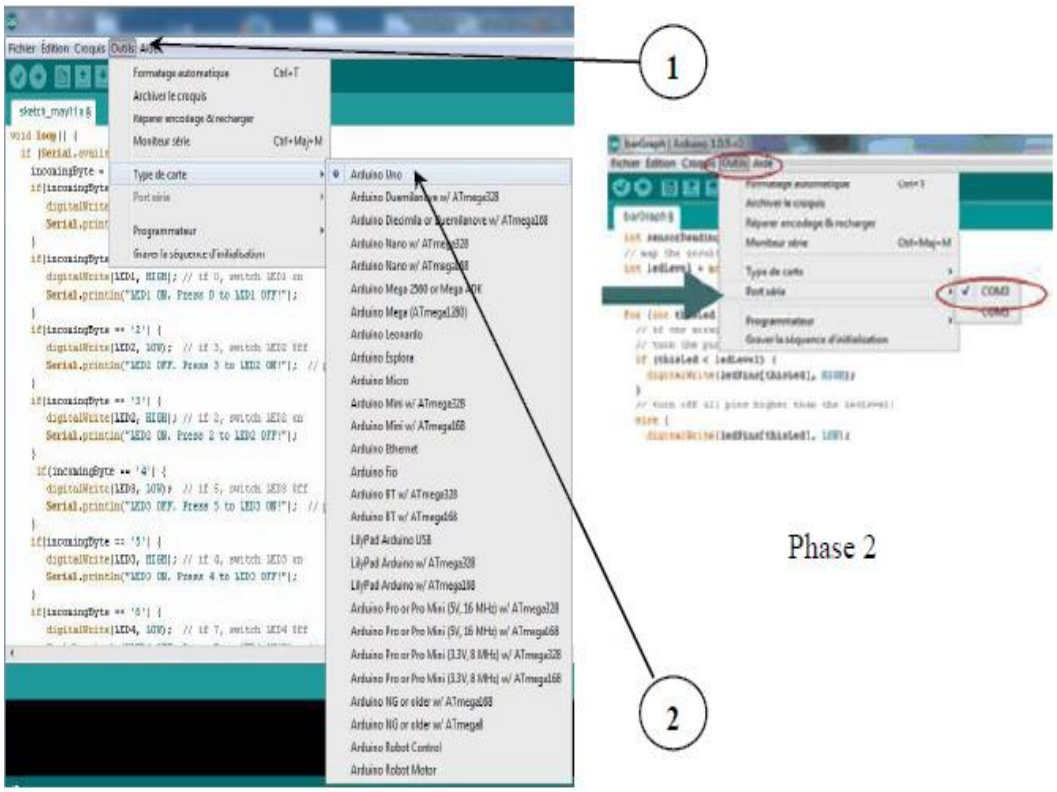


Figure (III.5): Paramétrage de la carte.

III.3.1.2.4 Les étapes de téléchargement du programme :

Une simple manipulation enchaînée doit être suivie afin d'injecter un code vers la carte Arduino via le port USB.

1. On conçoit ou on ouvre un programme existant avec le logiciel IDE Arduino.
2. On vérifie ce programme avec le logiciel Arduino (compilation).
3. Si des erreurs sont signalées, on modifie le programme.
4. On charge le programme sur la carte.
5. On câble le montage électronique.
6. L'exécution du programme est automatique après quelques secondes.
7. On alimente la carte soit par le port USB, soit par une source d'alimentation autonome (pile 9 volts par exemple).
8. On vérifie que notre montage fonctionne.

Figure (III.6): Les étapes de téléchargement du code.

III.3.1.2.5 Architecture interne :

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

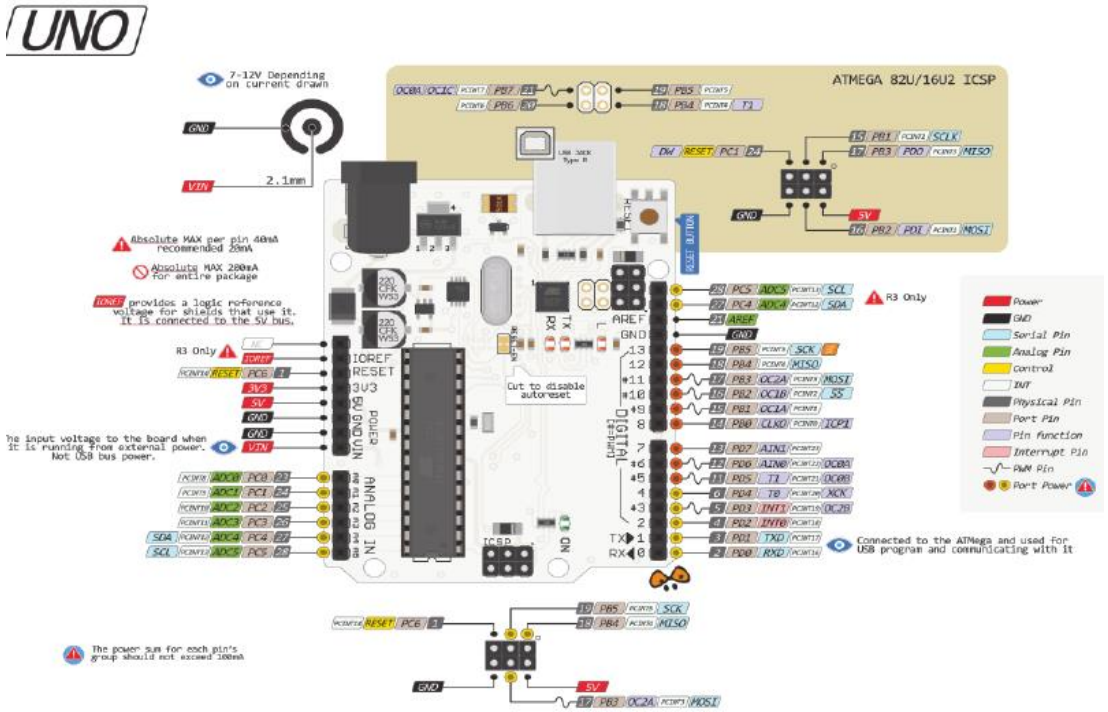


Figure (III.7): Architecture interne de la carte Arduino Uno.

III.4 Simulation sous Proteus :

Proteus est une suite logicielle destinée à l'électronique. Développés par la société **Labcenter** Electronics, les logiciels incluent dans Proteus permettent la CAO dans le domaine électronique. Deux logiciels principaux composent cette suite logicielle: ISIS, ARES, PROSPICE et VSM. Cette suite logicielle est très connue dans le domaine de l'électronique. De nombreuses entreprises et organismes de formation (incluant lycée et université) utilisent cette suite logicielle. Outre la popularité de l'outil, Proteus possède d'autres avantages: [12]

- Pack contenant des logiciels facile et rapide à comprendre et utiliser
- Le support technique est performant
- L'outil de création de prototype virtuel permet de réduire les coûts matériel et logiciel lors de la conception d'un projet

III.4.1 ISIS :

Le logiciel ISIS de Proteus est principalement connu pour éditer des schémas électriques. Par ailleurs, le logiciel permet également de simuler ces schémas ce qui permet de déceler certaines erreurs dès l'étape de conception. Indirectement, les circuits électriques conçus grâce à ce logiciel peuvent être utilisé dans des documentations car le logiciel permet de contrôler la majorité de l'aspect graphique des circuits.

III.4.2 ARES :

Le logiciel ARES est un outil d'édition et de routage qui complètement parfaitement ISIS. Un schéma électrique réalisé sur ISIS peut alors être importé facilement sur ARES pour réaliser le PCB de la carte électronique. Bien que l'édition d'un circuit imprimé soit plus efficiente lorsqu'elle est réalisée manuellement, ce logiciel permet de placer automatiquement les composants et de réaliser le routage automatiquement.

III.5 Simulation du pont quatre quadrants sous ISIS :

Simulation sous ISIS ne demande pas de programmation mais de connaître les composants électroniques, ces composants se trouvent dans une bibliothèque organisé selon le composant et le type, sa constructeur et les normes de support tension courant et autre norme.

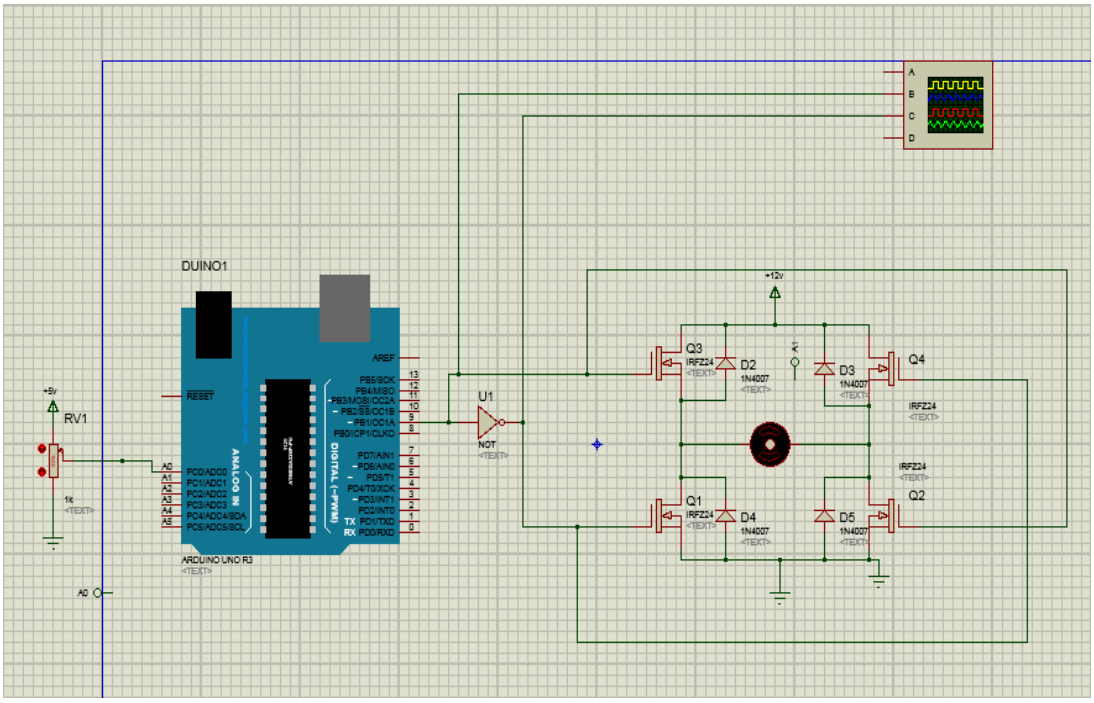


Figure (III.8): Un pont 4 quadrant sous ISIS conduit par Arduino.

Après la simulation sous ISIS nous pouvons vu le résultat par la rotation du moteur et visualisé l'oscilloscope:

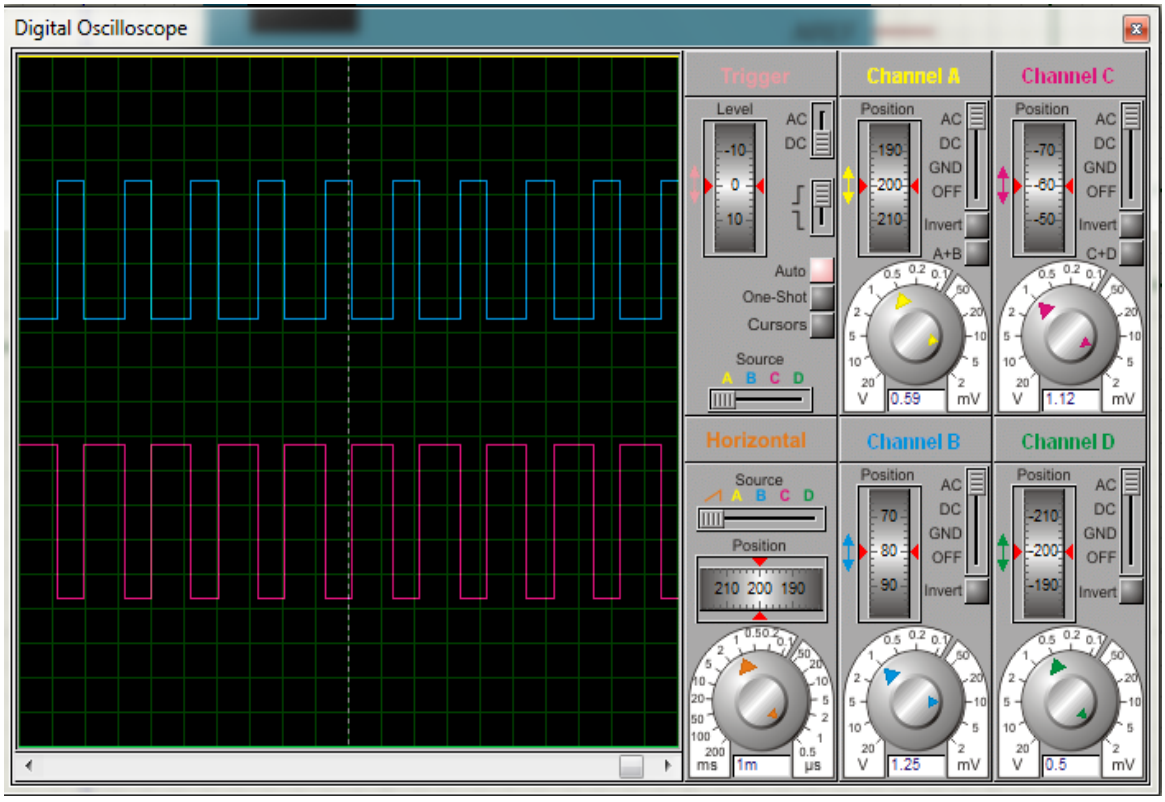


Figure (III.9): Sorties MLI et son complément.

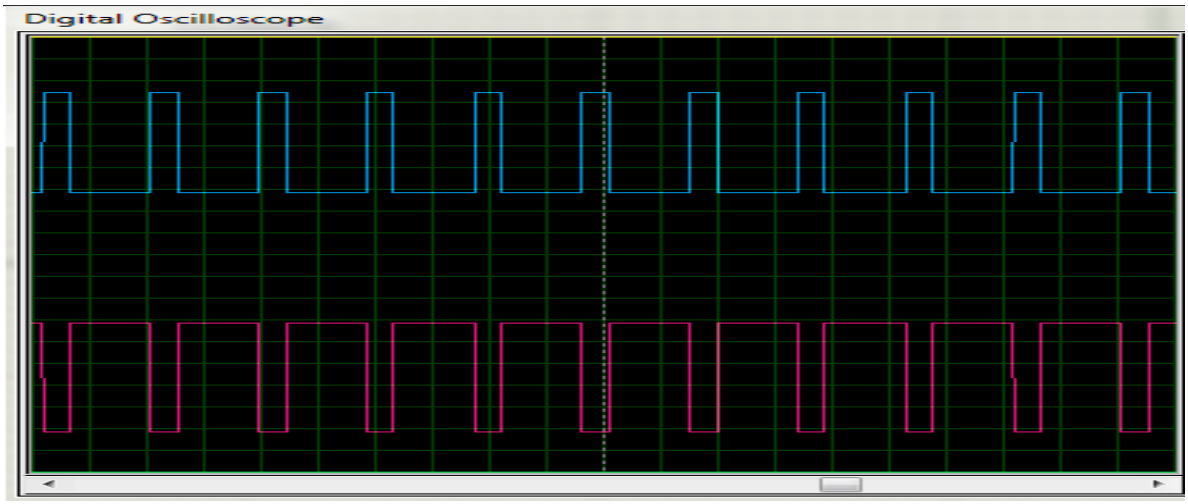


Figure (III.10) : La tension de Sortie avec un rapport cyclique de 75 %.

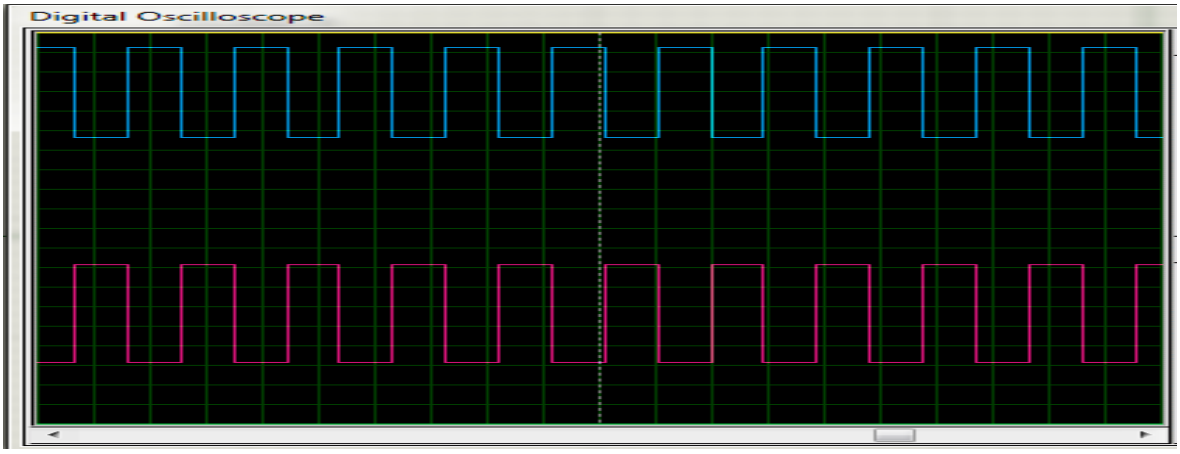


Figure (III.11) : La tension de Sortie avec un rapport cyclique de 50 %.

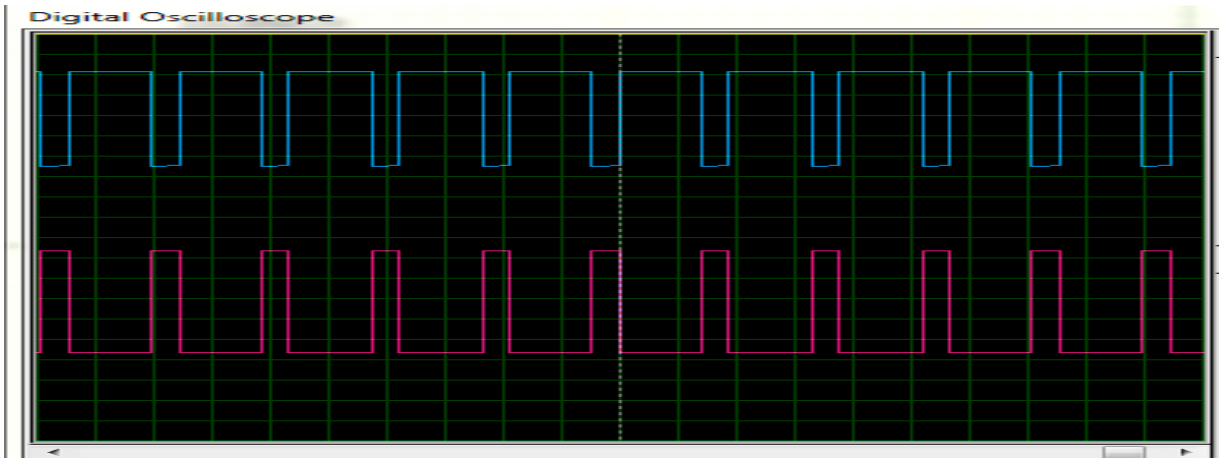


Figure (III.12) : La tension de Sortie avec un rapport cyclique de 25 %.

Explication : D'âpres la simulation qu'on a fait on trouve des résultats représenter par trois test obtenu par la variation de rapport cyclique l'un le rapport cyclique égale à 0.5 et 0.75 et 0.25 le programme que n a créée et donner dans l'annexe.

III.6 Simulation sous LabVIEW :

III.6.1 Les fenêtres de LabVIEW :

Le logiciel LabVIEW, grâce à son environnement à multiples fenêtres, permet de réaliser et d'exécuter rapidement des programmes simples, comparables aux fonctions d'un langage conventionnel, mais pouvant prendre à l'écran l'apparence d'un appareil de mesure, d'où leur appellation **Instrument Virtuel** (**V**irtuel **I**nstrument en anglais, ou "**VI**").

Le langage de programmation employé par LabVIEW est le langage graphique G. Ce langage est assimilable aux langages orientés objet, tel le C++, offrant des classes de données ayant des attributs spécifiques, ainsi que des opérateurs et des fonctions polymorphes agissant sur les données.

Tout programme exécutable construit en langage G comporte une interface utilisateur et un programme graphique situés dans deux fenêtres distinctes (**panneau avant** et **diagramme**).

L'environnement LabVIEW offre, dans trois autres fenêtres distinctes, des palettes indépendantes d'outils et d'objets permettant d'éditer les deux fenêtres du programme et de tester son fonctionnement.[19]

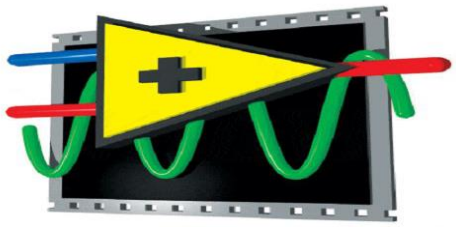


Figure (III.13): National instruments LabVIEW.

III.6.2 Les fenêtres du programme :

Un programme de LabVIEW comprend 2 fenêtres distinctes: le **panneau avant** servant d'interface avec l'utilisateur et le **diagramme** contenant le programme source en langage graphique G.

III.6.2.1 Le "panneau avant" :

Cette fenêtre, où apparaissent des objets sous forme de commandes d'entrée ou **contrôleurs** (*Controls*) ou d'**indicateurs** de sortie (*Indicators*), constitue l'interface interactive du programme. Le « panneau avant » vide apparaissant par défaut lors de la création d'un programme est indiqué sur la figure ci-dessous.

III.6.2.2 Le diagramme :

Cette fenêtre contient le code source graphique représentant le programme écrit en langage G. Le diagramme vide apparaissant par défaut lors de la création d'un programme est indiqué sur la figure ci-dessous.

III.6.3 Ce qui rend Arduino idéal pour LabVIEW :

La communauté Arduino est extrêmement vaste avec des milliers et même des centaines de des milliers de projets qui peuvent être trouvés en utilisant des recherches simples sur Google. En intégrant.

LabVIEW avec Arduino rend le prototypage encore plus simple en utilisant l'environnement guide de LabVIEW avec la plate-forme Arduino.

Officiellement, LabVIEW travaillera avec Uno et Mega 2560, nous devrions être capables de l'exécuter sur d'autres plates-formes Arduino telles que Nano. Construire votre propre Un conseil d'administration est tout aussi simple que lier Arduino à LabVIEW.

Présentation:

Le but de cet atelier est de découvrir comment se fait l'interfaçage entre le logiciel LabVIEW avec un Arduino. C'est aussi de comprendre comment récupérer les données de différents capteurs, comment par l'appuie sur un bouton il sera possible d'allumer une LED ou faire tourner un servomoteur. LabVIEW est un logiciel développé par National Instruments permettant de coder à l'aide de diagramme. Grâce à ce logiciel il est possible de créer des outils de mesure et de contrôle.

Pour établir cette communication, un certain nombre d'outils et logiciels sont nécessaires :

LabVIEW	Télécharger et installer LabVIEW (en s'inscrivant sur le site officiel de National Instruments, on peut télécharger une version d'évaluation de LabVIEW)
NI MAX	Télécharger et installer la dernière version des drivers NI-VISA : c'est un pilote qui permet à LabVIEW de communiquer avec les instruments comme l'Arduino (dans notre cas en utilisant une connexion USB
VI Package Manager	Télécharger et installer le module VIPM (VI Package Manager) permettant d'ajouter de nombreux compléments, dont ceux de l'Arduino.

III.6.4 Installation des packages :

Une fois tous les logiciels installés, lancez VIPM. Le but est d'installer dans LabVIEW les instruments virtuels (VI) d'Arduino.

Pour cela, il existe deux types de VI :

- LIFA : LabVIEW Interface For Arduino
- MakerHub(LINX).

Nous allons télécharger les deux types de VI ensuite nous verrons la différence entre les deux outils. Dans la zone de recherche de VIPM, tapez Arduino et chercher LabVIEW Interface For Arduino :

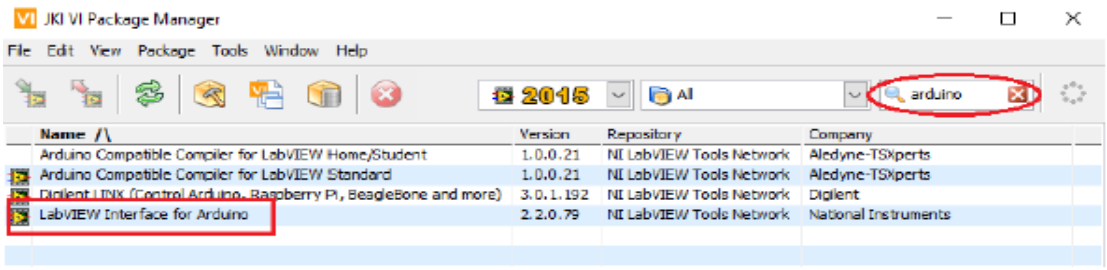


Figure (III.14): Recherche du VI LIFA.

III.6.5 Installation du firmwar LIFA :

Le firmware LIFA est disponible après installation de LabVIEW 2015 dans le répertoire «C:\Program Files (x86)\National Instruments\LabVIEW 2015\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA_Base». Au niveau de ce répertoire, on trouve plusieurs fichiers dont les trois principaux sont:

- le sketch "LIFA_Base.ino" : c'est ce sketch que vous compilerez et transfèrerez à la carte Arduino.
- le sketch "LabVIEWInterface.ino" : il contient l'implémentation de toutes les fonctions utilisées par LabVIEW pour dialoguer avec la carte Arduino
- le fichier d'inclusion "LabVIEW Interface. h" : ce fichier d'en-tête est particulièrement intéressant puisque c'est lui qui contient les "define" que vous pouvez modifier pour adapter LIFA.

Finalement, il faut charger le sketch "LIFA_Base.ino" dans l'environnement Arduino, le compilez puis programmez la carte Arduino ; ainsi la carte est prête à fonctionner avec LabVIEW

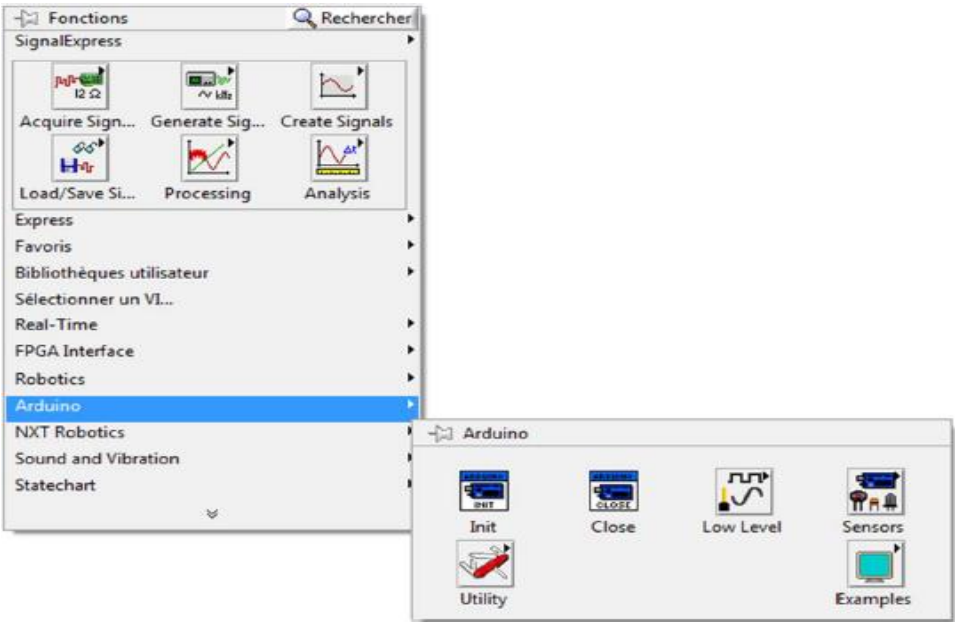


Figure (III.15): Bibliothèque de l'Arduino.

III.7 Programme sous LabVIEW :

III.7.1 Diagram de bloc :

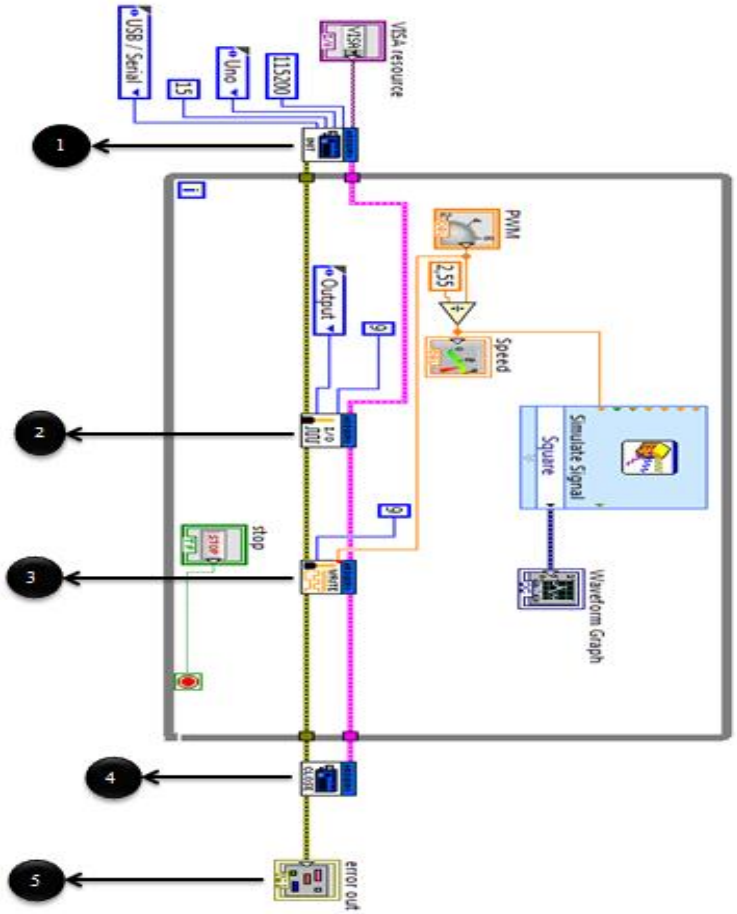


Figure (III.16): Diagramme de bloc Programme de MLI sous LabVIEW.

1 : Init – Initialise la carte.

2 : Set Pin Mode - Déclare si le Pin est une entrée ou une sortie. Dans notre cas, le Pin9est une sortie.

3 : Digital Write - Déclare le Pin spécifique qui est déclaré pour relier le Cnob avec l'afficheur de vitesse.

4 : Arduino Close – ferme la carte Arduino.

5 : Gestionnaire d'erreurs : se trouve dans la palette Programmation – Dialogue/IU (Interface utilisateur) – Gestionnaire d'erreurs simples. Affiche une erreur avec son code s'il y a lieu.

III.7.1.2 Câblages :

- ✓ magenta : déclaration de la carte Arduino utilisée.
- ✓ ocre : transfert des erreurs au gestionnaire d'erreurs.

Comportement attendu après lancement de l'exécution du VI :

- les diodes T_x et R_x clignotent, signe que le programme se télécharge dans la carte.
- la led L s'allume sur la carte Arduino.
- après appui sur le bouton stop, la Led s'éteint.

Remarque : la gestion des erreurs est facultative. Vous n'avez pas absolument besoin de relier les blocs avec la ligne Ocre, ni de positionner le gestionnaire d'erreurs. Cependant, les messages affichés peuvent être intéressants. Par exemple, il arrive parfois que LabVIEW ne trouve plus la carte. Dans ce cas, il faut fermer le VI et le relancer. Vous n'aurez un message explicite que si vous avez câblé le gestionnaire d'erreurs.

III.7.2 Face avant :

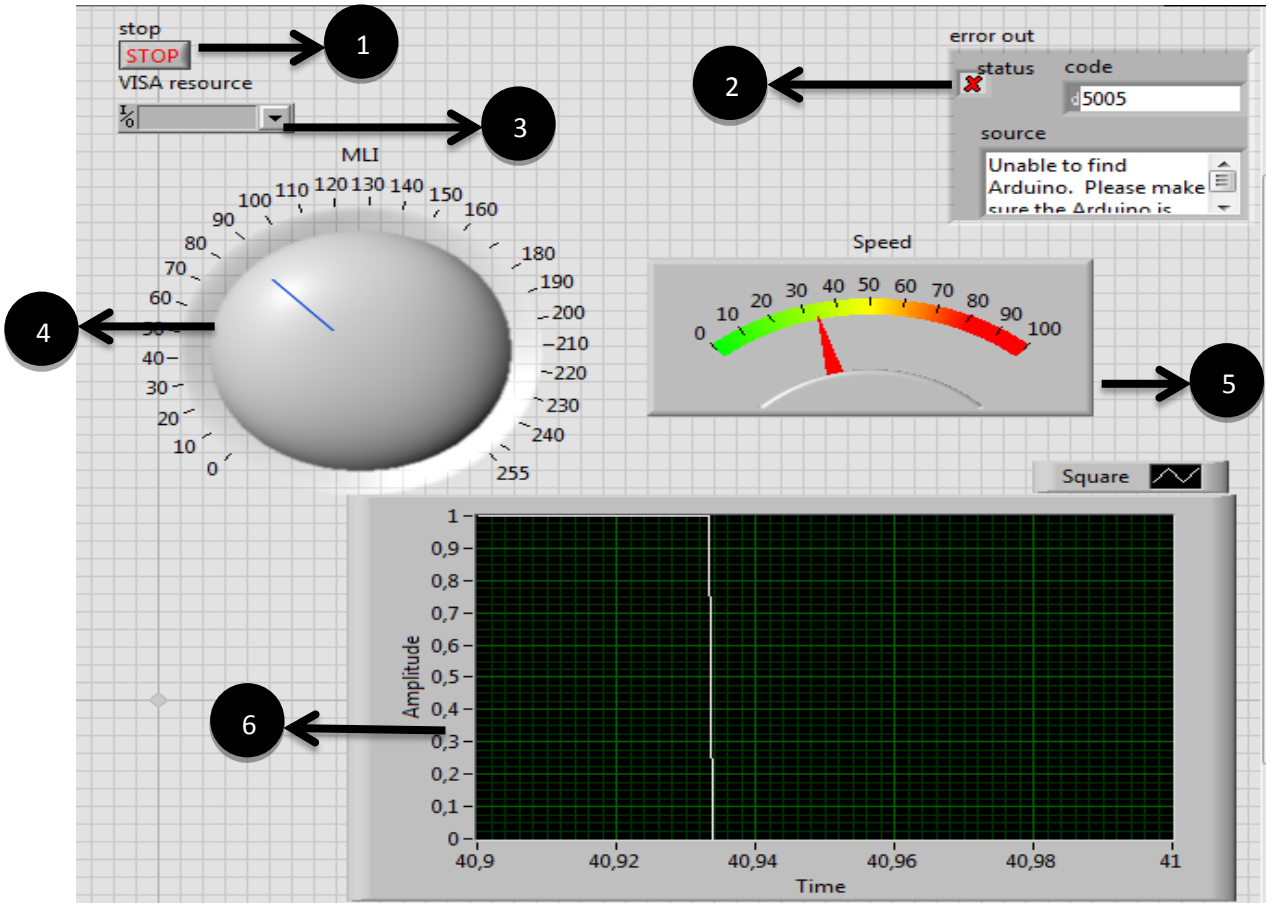


Figure (III.17): Face avant Programme de MLI sous LabVIEW.

1 : Botton (Stop) permis de stopper l'exécution du programme.

2 : Interface qui permit d'afficher l'erreur.

3 : VISA ressource (configuration du port).

4 : Potentiomètre permit de générer le signal MLI (PWM).

5 : Interface (Speed) permit d'afficher la vitesse du moteur.

6 : Interface d'affichage du signal MLI simulé.

III.8 Réalisation :

III.8.1 Partie expérimentale :

Tout système de conversion, par interrupteurs de puissance, doit être commandé par des signaux d'impulsion. Cependant, ces derniers doivent conditionnés et adaptés en assurant le niveau de tension, de courant nécessaires et les temps morts adéquats pour éviter tout défaut à différents régimes de fonctionnement. C'est pour cet objectif que cette deuxième partie du travail est consacrée pour la mise en œuvre expérimentale. Dans ce que suit, nous proposons une architecture de la carte de commande rapprochée de l'hacheur ainsi que sa réalisation.

III.8.2 Circuit de commande :

Notre commande a été réalisée par intermédiaire d'un microcontrôleur ATMega328, ce circuit est considère comme un driver de moteur car de son signal MLI (switched mode controller for DC Motor driver), mais avant introduire le circuit en réalisation nous utilisons logiciel LabVIEW pour générer le signal de commande MLI numérique, la méthode d'engendrer le signal sous LABVIEW est montrée dans ce chapitre.

III.8.4 Réalisation de la commande :

III.8.4.1 La carte I XDP :

Le composant principal utilisé dans la carte est *l'IXDP*. Il est destiné principalement à l'application en triphasé ou monophasé, en assurant le temps nécessaire (temps mort) pour séparer les deux signaux logiques (signal *MLI* directe *PWM* et $\overline{PWM}$ le complément).

En pratique, le contrôle d'un moteur nécessite de ***PWM*** pour le contrôle des quatre interrupteurs d'un hacheur quatre quadrants reliés aux enroulements du stator du moteur

Les quatre interrupteurs forment un " pont a H", qui peuvent être utilisés pour alimenter une charge.

Deux commutateurs sur le même "Pont a H" ne doit jamais être simultanément amorcé, sinon les bornes positive et négative seront court-circuités.

Remarque : la carte IXDP a été réalisé [21].

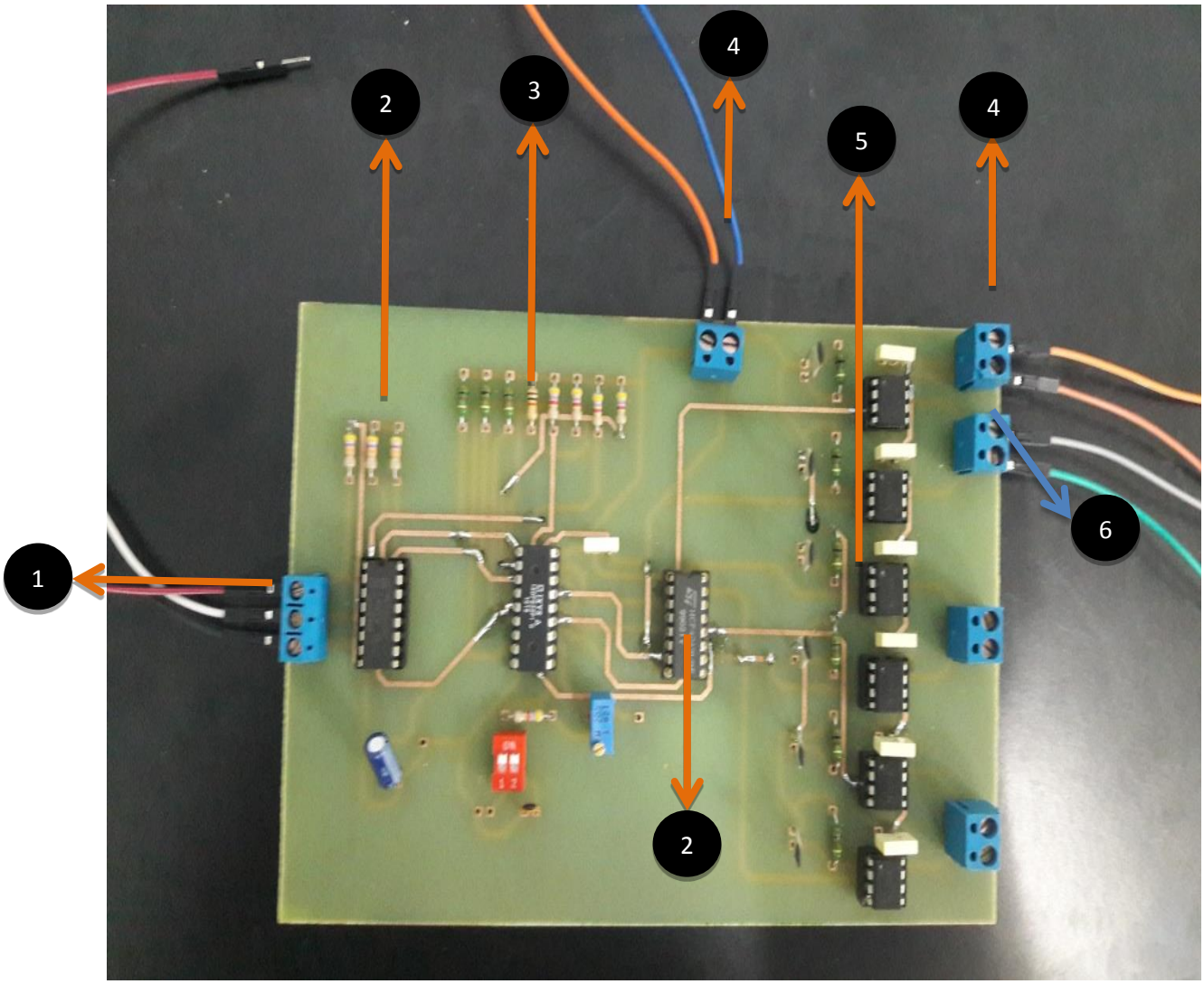


Figure (III.18): La carte de commande IXDP.

1 **:** L'entrée du signal PWM de L'Arduino

2 **: CD4049U** Hex/Converter et conversions au niveau

3 **: XP630**création du temps mort

4 **:** Entré d'alimentation 5v

5 **:** Les Optocoupleur de type **4N40** pour la protection

6 **:** Sortie du signal MLI et le complément

II.8.4.2 Les drivers :

Le schéma suivant représente le circuit que na créée qui permet de contrôler les IGBT du hacheur

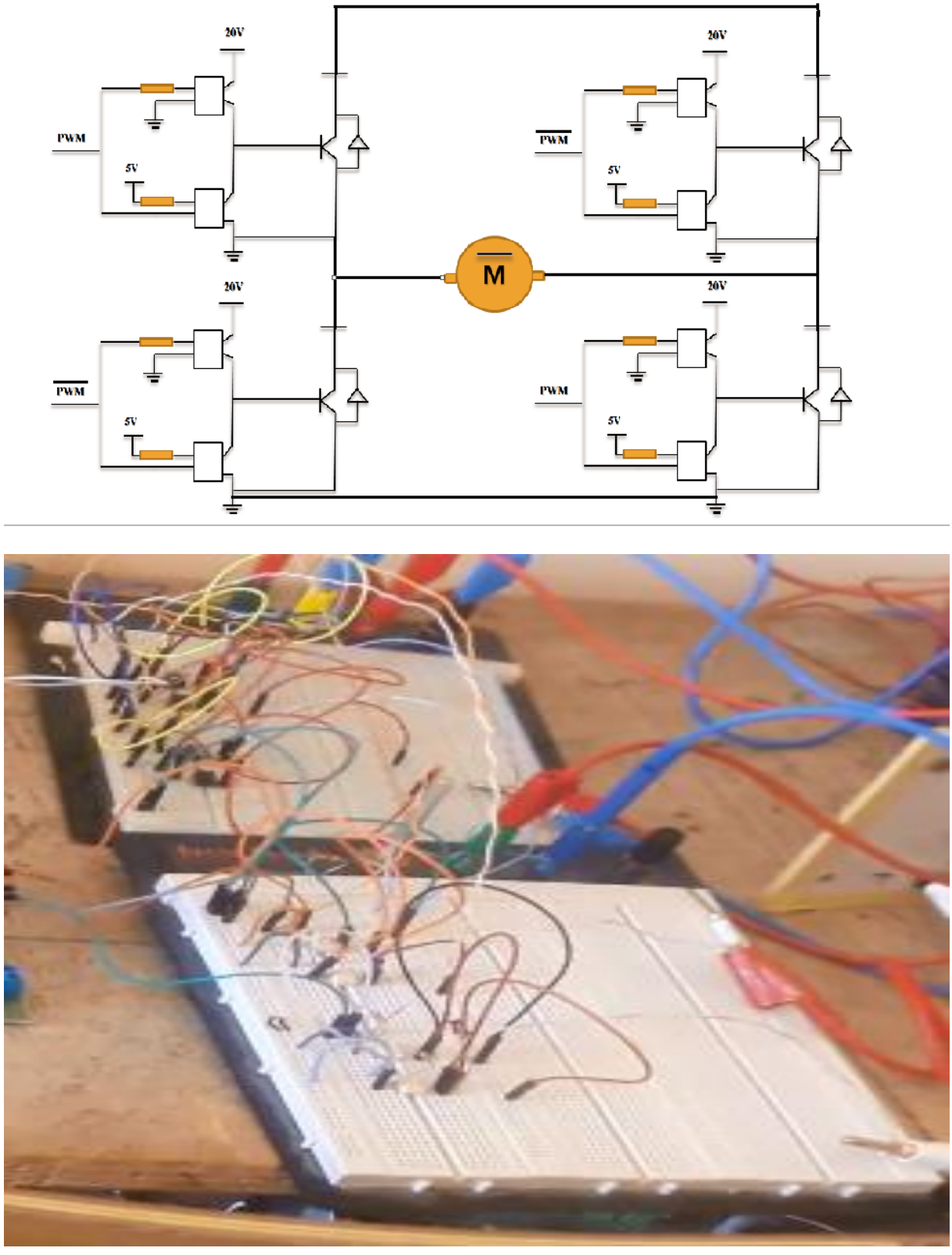


Figure (III.19): Schéma et la réalisation du circuit des drivers

Les driver est constitué de 8 optocoupleur de type **4N40.**

III.8.4.3 L'Optocoupleur de type 4N40 :

- **Description**

Les **4N40** ont un infrarouge d'arséniure de gallium diode électro luminescente couplée optiquement avec un silicium activé par la lumière redresseur commandé dans un boîtier à double rangée.

Applications :

- Circuits logiques de puissance faible,
- Matériel de télécommunications,
- Electroniques portatifs,
- Relais statiques.
- Les systèmes de couplage interfaçage des différents potentiels et impédances.

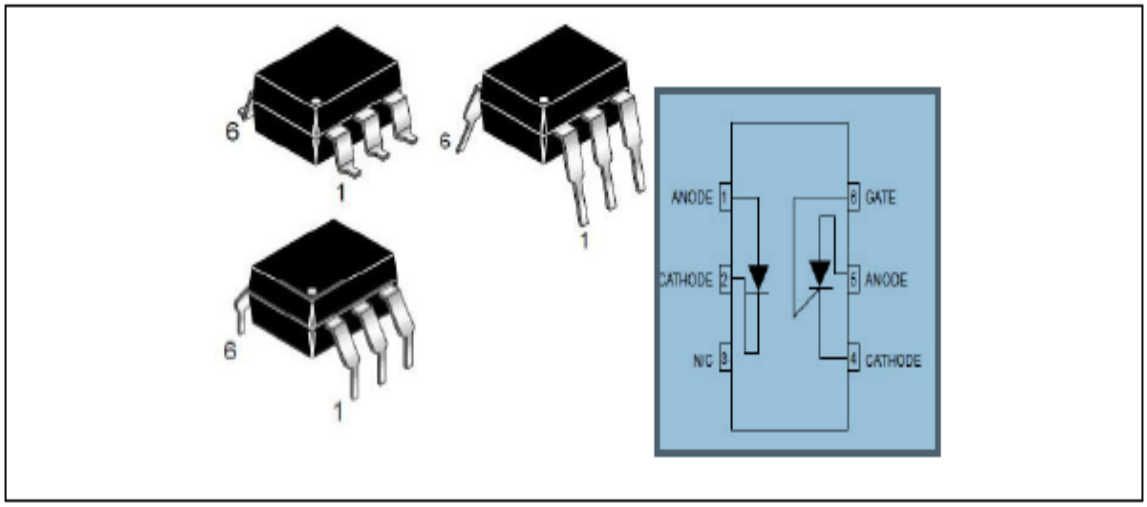


Figure (III.20) : Circuit intégrés 4N40.

III.8.4.4 Pont de hacheur TOSHIBA:

Pour les applications à faible tension nous sommes le choix d'utiliser des ponts intégrés en série sous des circuits de type**L293D** (fabriqué par **ST Microélectronique),** mais pour des applications à des tensions impotentes nous sommes dirigé vers l'utilisation des ponts supportent ces tensions, sur notre mémoire nous utilisons un pont des **IGBT** fabriqué par **TOSHIBA** à norme **MG50J2YS50** caractérisé par une tension 600V et 50A, le temps réponse à la fermeture 0.4µs et à la ouverture 0.5µs.

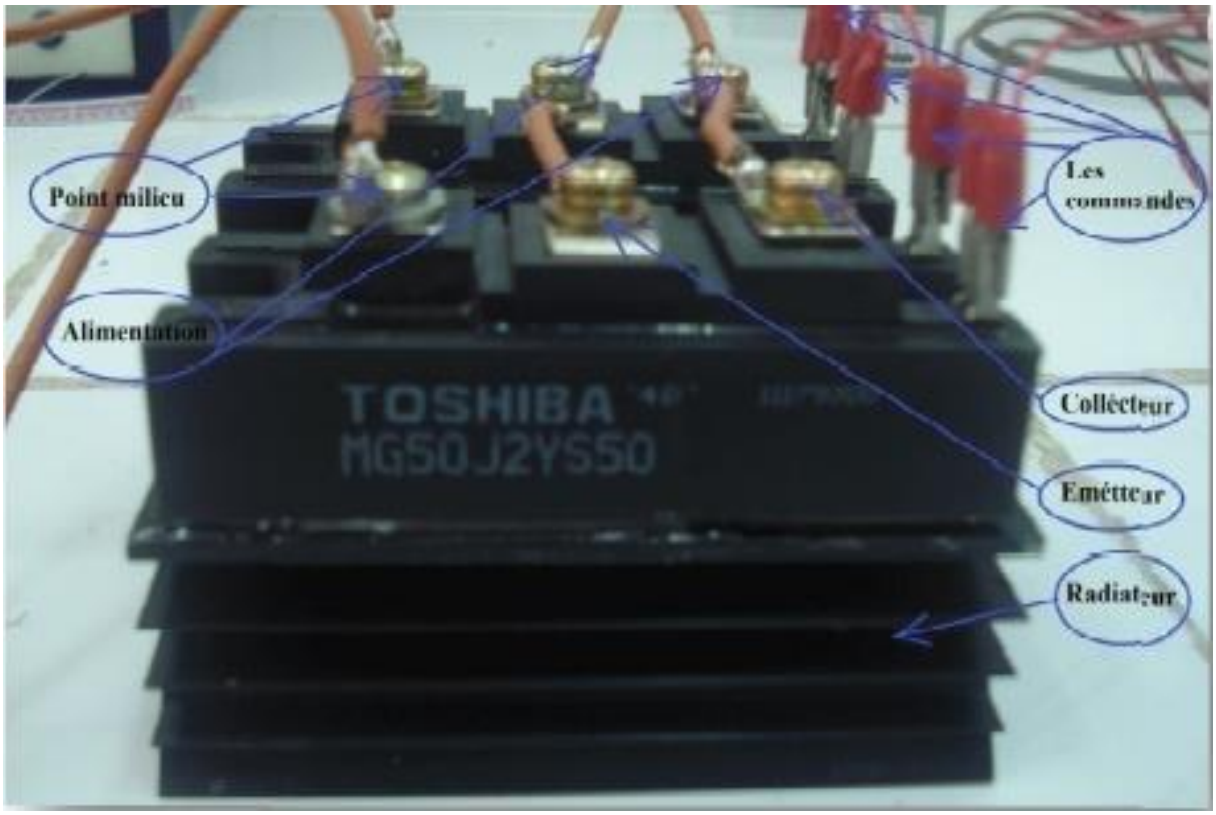


Figure (III.21): Pont de hacheur fabriqué par TOSHIBA.

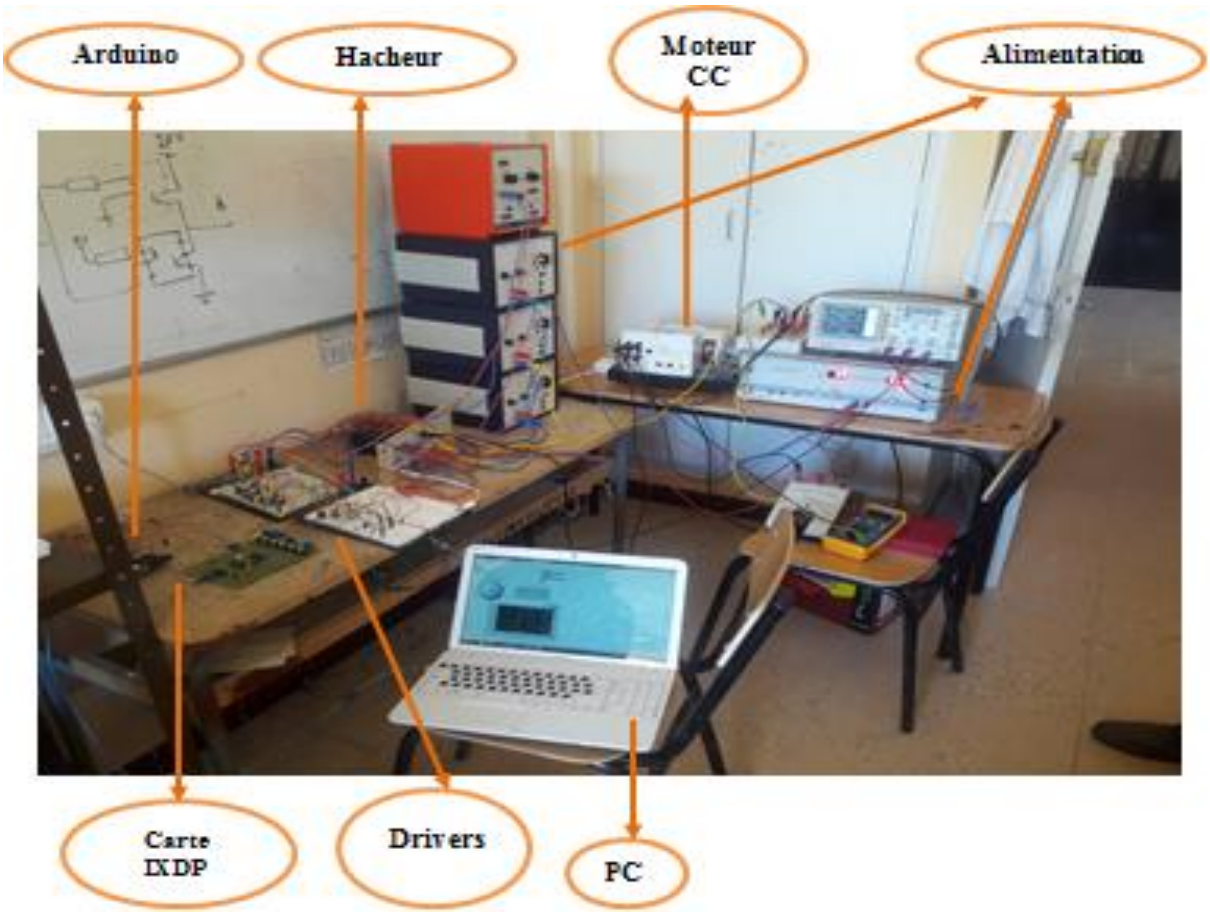


Figure (III.22) : Réalisation finale du système a commandé

III.9 Les résultat obtenu :

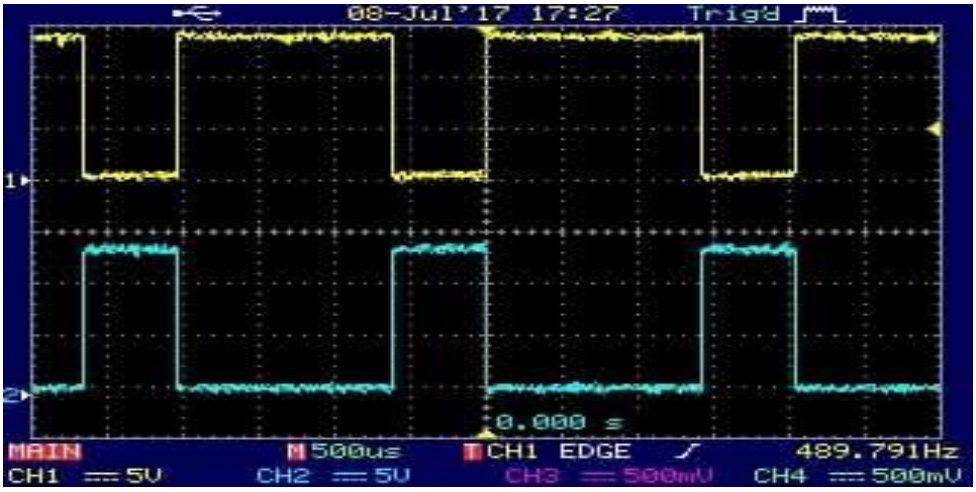


Figure (III.23): Signaux PWM de la commande et on mode complémentaire.

III.9.1 Signaux PWM de la commande a et la tension et le courant de moteur :

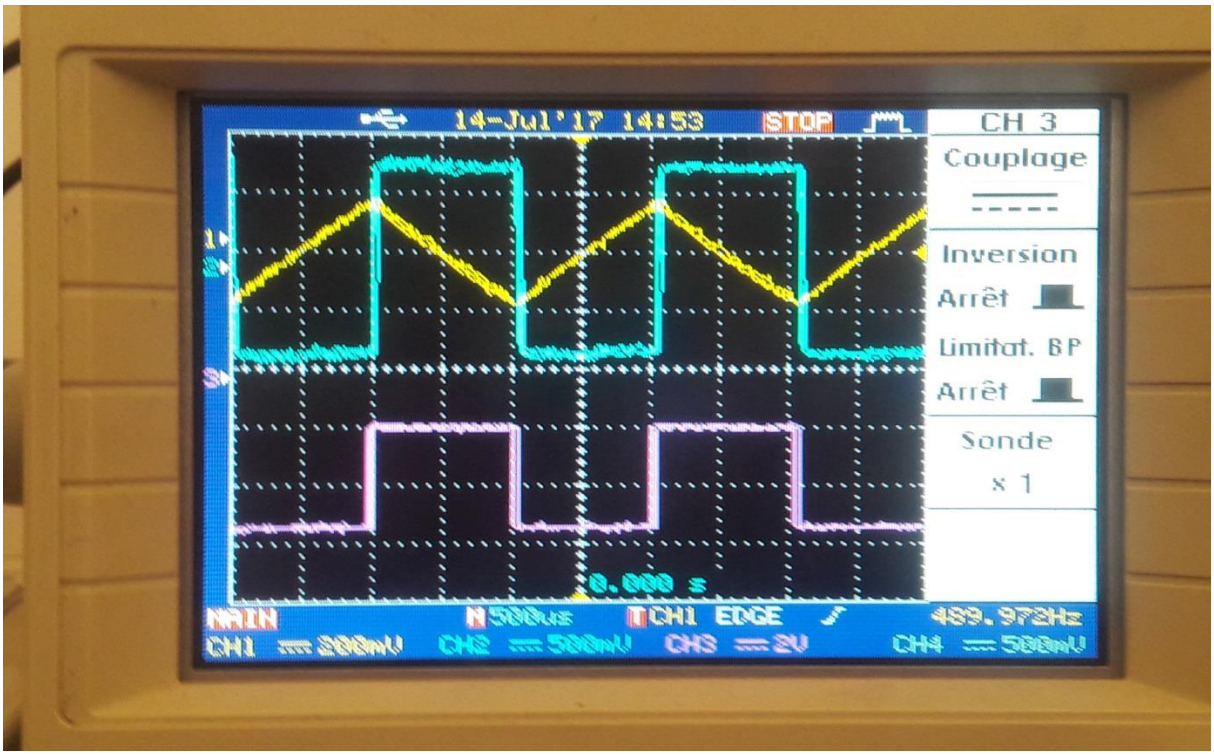


Figure (III.24): α = 50%

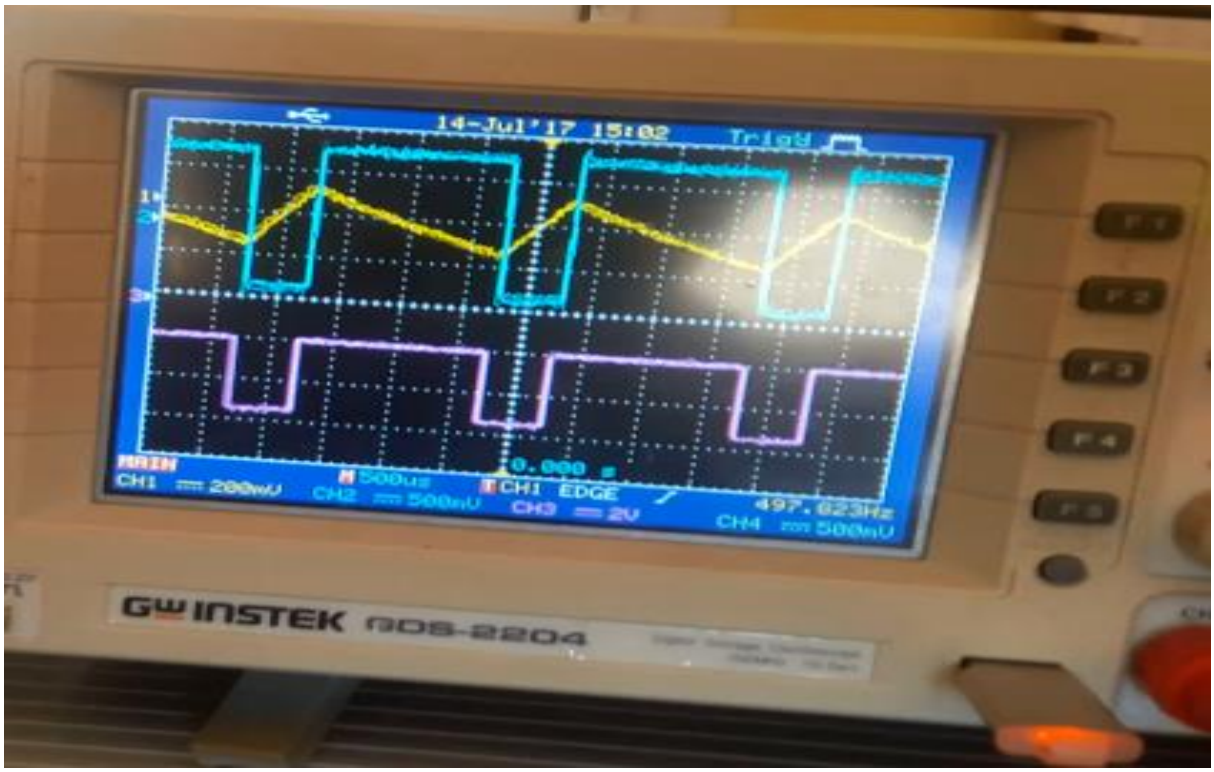


Figure (III.25): α = 75%

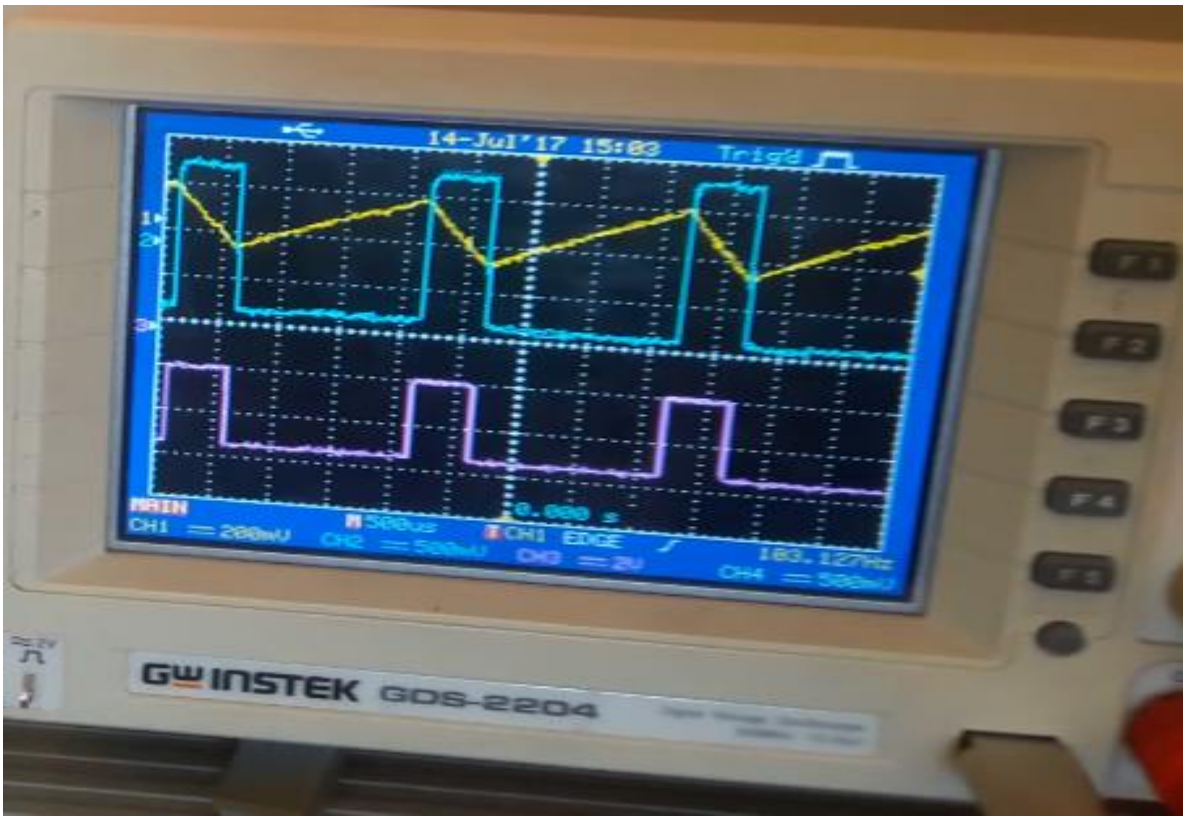


Figure (III.26): α = 25%

La tension du moteur

Le courant du moteur

Le signal PWM

III.9.2 Explication :

D'âpres l'étude qu'on a fait on trouve des bons résultats pratique représenter par trois test obtenu par la variation de rapport cyclique l'un le rapport cyclique égale à 0.5 on trouve la moyenne de la tension est nulle alors le moteur s'arrête et pour le deuxième test le rapport cyclique est égale à 0.25 on trouve la tension moyenne aux bornes du moteur à courant continu est égale à 45v qui implique le moteur va tourner dans le sens direct, et pour le troisième test on fixe le rapport cyclique a une valeur de 0.75 on trouve la tension moyenne est égale a -45v qui signifie le signe moins le sens de rotation inverse du moteur avec la même vitesse de rotation du 2 Emme test qui implique la commande MLI fonctionne correctement dans notre réalisation.

III.9.3 Analyse des résultats obtenus :

Dans ce travail, les différentes caractéristique nous montre que :

- La variation de rapport cyclique varie la vitesse de moteur à courant continu.
- La vitesse de rotation d'un moteur à courant continu dépend sa tension d'alimentation.
- On peut inverse la sens de rotation du moteur avec un hacheur quatre quadrant.

III.10 Conclusion :

Dans ce chapitre, nous avons présenté les résultats de simulation avec logiciel Proteus et **LabVIEW**. Une application a été créée sous **LABVIEW** pour faire la liaison entre L'Arduino et un PC afin de valider des signaux de la commande "PWM" vers le moteur à courant continu. Cette dernière montre que la vitesse de rotation d'un moteur à courant continu dépend de sa tension d'alimentation, la variation du rapport cyclique fait varier la tension ou borne de moteur à courant continu et on peut aussi inverser le sens de rotation du moteur avec un hacheur à quatre quadrants. Le résultat obtenu par simulation est validé par des essais expérimentalement. On peut conclure que les résultats obtenus sont satisfaisants compte tenu des limitations du matériel et des moyens dont nous disposons ainsi que les contraintes liées au temps.