République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de L'**E**nseignement **S**upérieur et de la A **R**echerche Scientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DE **G**ENIE **E**LECTRIQUE ET D'**I**NFORMATIQUE
DEPARTEMENT D'ELECTROTECHNIQUE

Mémoire de Fin d'Etudes
de MASTER PROFESSIONNEL

Domaine : Sciences et Technologies

Filière : Génie Electrique

Spécialité : ELECTROTECHNIQUE INDUSTRIELLE

Présenté par
Mohamed RILI
Nacer REZZOUK

Thème

Utilisation des Cartes Arduinos pour la Commande en Electronique de Puissance

Mémoire soutenu publiquement le 09 Juillet 2017 devant le jury composé de :

M Salah HADDAD
Professeur, UMMTO, **Président**

M^{elle} Rahma KACHENOURA
Maître de conférence-B, UMMTO, **Rapporteur**

M Ahmed NAHI
Maître Assistant-A, UMMTO, **Examinateur**

M Said AISSOU
Maître Assistant-B, UMMTO, **Examinateur**

Remerciements

Nous remercions d'abord avant tous, le bon Dieu qui nous a donné le courage, la patience, la santé et la volonté pour venir à bout de ce travail, et qui nous a permis d'acquérir ce savoir et d'enrichir nos connaissances.

Nos remerciements les plus vifs à toutes les personnes ayant pris part de près ou de loin à notre formation, a tous ceux qui nous ont un jour donné un conseil et à tous ceux qui nous ont guidé sur la voie du savoir. Particulièrement, notre promotrice Melle : R. KACHENOURA pour son aide, ses conseils judicieux, sa disponibilité sans faille, son amour du travail, le suivi et l'intérêt qu'elle nous a apportés tout au long de notre travail.

Nous tenons aussi à remercier Mr : A. NAHI, Mr : S. AISSOU,

Mr : A. CHALLAL et Mme :F. MEGHERBI-BITAM pour leurs précieux conseils.

Nous adressons de chaleureux remerciements aux membres de jury Mr : S. HADDAD, A. NAHI, S. AISSOU qui ont aimablement accepté de juger notre travail et de lui accorder l'attention nécessaire.

Dédicaces

Je dédie ce modeste travail a ceux qui me pérennent la dignité, l'honneur et la joie de ma vie mes très chers Aomar et Malika et je dis merci à leurs soutien, patience et confiance.

A ma chère sœur Sabrina, ma tante Zhour, mon oncle Rabah et son épouse et ses deux filles Sarah et Farah, mon oncle Ali et son épouse et leurs deux filles Lydia et Dalia, et a toute la famille chacun par son nom.

A tous les amis Hidouche, Said, Lahcen, Idir, Hocine, Zouhir, Amar, Krimo, Nora, à mon binôme Nacer et toute sa famille ainsi qu'à tous les amis qui m'ont aidé de près et de loin.

M.RILI

Je dédie ce modeste travail a mes très chers parents Meziane et Ourdia et je dis merci à leurs soutien, patience et confiance.

A mes frères et sœurs Mohamed, Tassadit, Lyes.

A tous mes amis Nabil, Hakim, Taous, Zouhir, Krimo, Amar, Hocine et tous ceux qui m'ont donné leurs aides de près ou de loin.

N. REZZOUK

A toute la promotion Master II professionnel (2016/2017)

Résumé

Les convertisseurs statiques sont à la pointe de la technologie durent ces dernières années, ils se sont incrustés dans presque tous les domaines, du coup il faut concevoir un convertisseur qui répond parfaitement au besoin de l'utilisateur coté performance et fiabilité, est surtout avec le moindre cout possible.

Du cout les convertisseurs statiques se composent généralement d'une partie commande et d'une partie puissance, dans notre cas on a fait notre travail sur la partie commande ou bien traitement de l'information, ce qui nous a amenés à utiliser les cartes Arduinos comme organe de control, et de faire une étude sur ces cartes, afin de voir les différentes performances et caractéristiques ainsi de constaté les avantages et les inconvénients de ces cartes, car pour bien dimensionner la partie puissance de n'importe quel convertisseur il faut tout d'abord, bien dimensionner la partie commande, car toute erreur sur cette dernière partie va être amplifié dans la partie puissance engendrent ainsi des perturbations dans notre installation au pire encore la détérioration de matériels.

Donc pour tester ces cartes Arduinos, on a réussi à faire des applications en temps réel afin de tester leurs fiabilités, et voir aussi leurs fréquences de commutation maximale.

Car un convertisseur qui monte très bien en frequence nos permet :

- De réduire les volumes des filtres à utiliser donc un volume plus réduit du convertisseur ;
- De plus on gagne en terme de poids ;
- Et le cout et automatiquement réduit car dans les convertisseurs c'est les filtres qui sont chers.

Mots clés

- Arduino.
- Matlab.
- Fritzing.
- Modulation de Largeur d'Impulsion (MLI ou PWM).
- Machine à courant continu.
- Le hacheur quatre quadrants.
- Les convertisseurs statiques.

Sommaire

Sommaire

Chapitre 3 : Matlab Simulink et MLI avec les cartes Arduinos

Chapitre 4 : Acquisition et restitution des données sur Arduino

Sommaire

Listes des figures :

Introduction Générale

Introduction générale

L'électronique de puissance a pris beaucoup d'importance avec le développement industriel et la diversification ainsi que les exigences techniques de son utilisation. Elle touche tous les domaines où un traitement d'énergie est nécessaire. Les puissances traitées s'étalent de quelques mW jusqu'au GW. Ceci est d'autant plus vrai que tout circuit électronique a besoin d'une alimentation en énergie qui doit être conforme aux conditions et aux restrictions du futur. Ainsi, une utilisation efficace de la puissance est non seulement avantageuse mais aussi indispensable.

L'électronique de puissance est une branche de l'électrotechnique qui évolue extrêmement vite, cela est dû principalement aux performances des composants qui ne cessent de s'améliorer, en raison notamment de l'application aux courants forts des procédés de fabrication de la micro-électronique cela nous permet d'avoir des convertisseurs avec de très bon indicateurs des performances, à savoir :

- La puissance du convertisseur ;
- La fréquence du travail ;
- De très bons rendements ;
- Des volumes les plus réduits possible ;
- Un poids le plus faible possible ;
- Un coût le plus bas possible ;
- Une grande fiabilité.

Cette évolution est aussi une conséquence de l'emploi de la micro-informatique qui permet d'élaborer des commandes de plus en plus sophistiquées.

D'un point de vue d'un automaticien, un convertisseur est considéré comme une simple fonction de transfert avec une constante de temps imposée par la charge. Pour un électrotechnicien le problème est plus complexe à cause des différentes bases de temps qui se côtoient, souvent, on parle de deux bases de temps, liées à la fréquence de la porteuse et à celle de la référence. Les fréquences de la porteuse sont très exigeantes. Si la commande est réalisée par un microcontrôleur, c'est cette fréquence qui doit être prise comme référence de travail [4-5].

La carte Arduino est un microcontrôleur qui draine des passions dans différents domaines d'utilisation. Depuis son développement en 2005 à ce jour elle a réussi à s'incrusté dans différents milieux en commençant pas des utilisations de loisirs jusqu'à atteindre des domaines de recherche.

Dans ce travail, l'idée est de voir quelles sont les possibilités offertes par ces cartes en électronique de puissance, c'est la raison pour laquelle on va s'intéresser à ses performances en fréquence que cela soit d'un point de vue acquisition ou restitution.

Pour mener à bien notre travail nous l'avons organisé comme suit :

Après une introduction générale, le premier chapitre introduit succinctement les problèmes liés à la conception en électronique de puissance.

Dans le deuxième chapitre une présentation des différentes cartes Arduinos sera faite.

Le troisième chapitre sera consacré à l'analyse du l'utilisation des cartes Arduinos avec le logiciel Matlab.

Le quatrième chapitre sera consacré aux performances de ces cartes en terme d'acquisition et de restitution dans le cas où elles sont programmées en langage machine.

Dans le cinquième chapitre une application qui consiste à alimenter un moteur à courant continu via un pont hacheur quatre quadrants sera faite.

Nous terminons notre travail par une conclusion générale.

Chapitre 1

Généralités

Sur

L'électronique de puissance

1.1. Introduction

La puissance de gestion est un aspect essentiel de conception des systèmes dans tous les domaines de l'électrotechnique, c'est le souci principal dans les systèmes de l'électronique de puissance ainsi que dans la production et la distribution de l'énergie. On se concentre dans l'électronique de puissance beaucoup plus sur la conversion, l'efficacité de la conversion, et le contrôle du transfert de l'énergie électrique, qui doit être réalisé par des convertisseurs statiques.

L'importance de l'électronique de puissance augmente avec la croissance de la complexité d'utilisation et l'augmentation des équipements électroniques dans la société moderne. Ceci est d'autant plus valable parce que n'importe quel circuit électronique a besoin d'une alimentation d'énergie qui doit être conforme aux conditions et aux restrictions du futur. Ainsi, une utilisation efficace de la puissance est non seulement avantageuse mais aussi essentielle.

Le domaine d'application des convertisseurs de puissance s'étend des alimentations pour des équipements, tel que l'alimentation pour équipement utilisé au niveau des télécommunications où les alimentations d'énergie pour l'électronique grand public (comme les ordinateurs portatifs, les téléphones mobiles, etc…), jusqu'au transport et à la production de l'énergie, surtout les énergies renouvelables. Ce type de source exige des convertisseurs de puissance qui doivent s'adapter à leur nature qu'elle soit de nature électromécanique, électrochimique ou autre, avec la charge, qui est dans de nombreux cas, le réseau de distribution.

Chaque champ spécifique d'application de l'électronique de puissance exige des topologies différentes de convertisseurs, différents composants semi-conducteurs et des contraintes imposées par les normes exigées par le domaine d'utilisation. Cependant, le concept général d'un circuit de commutation « convertisseur statique » avec des composants passifs comme éléments de stockage d'énergie reste essentiellement le même. Une importante restriction est que, un convertisseur de puissance devrait être conçu de telle manière que les ajustements maximaux de puissance à l'intérieur correspondent à l'espace disponible pour mettre en application le convertisseur. Seulement, juger le convertisseur uniquement sur la base de sa densité de puissance est quelque peu insuffisant. Par exemple, un convertisseur pour l'électronique grand public devrait être le moins coûteux possible pour résister à la concurrence sur le marché. La conception d'un convertisseur avec la densité de puissance la plus élevée, signifie, une utilisation des meilleurs composants disponibles. Un impact direct sur le prix est ressenti, avec inévitablement un produit plus onéreux. Cet exemple simple, montre que l'objectif de garder la densité de puissance la plus élevée possible et maintenir un

coût acceptable est un vrai dilemme. Ce choix est finalement résolu par le concepteur qui juge l'importance des exigences. Dans la pratique, un concepteur habile explore plusieurs scénarios et choisit celui qui, à son avis, fortement subjectif est le meilleur. Autrement dit, quand on crée une nouvelle conception, des conditions contradictoires multiples doivent être satisfaites et des solutions possibles multiples doivent être explorées. Formellement, ce processus s'appelle la synthèse [4],[5].

La synthèse peut être faite de plusieurs manières différentes. Par exemple, une approche directe vers l'avant est de concevoir et d'établir de nombreux prototypes, et alors avec différentes mesures et tests pour améliorer les circuits. La situation où la simulation remplace la conception de prototypes est légèrement meilleure, puisque moins de temps et de matériaux sont consommés. Cependant, beaucoup de paramètres du convertisseur doivent encore être ajustés par le concepteur. Le choix de ces ajustements dépend subjectivement et fortement de l'expérience antérieure du concepteur.

1.2. Définition d'un cahier des charges en électronique de puissance

La définition rigoureuse d'un cahier des charges est la chose la plus difficile. Cela ne peut être vraiment établi que par approches successives. On aboutit à la notion de projet et de prototype de projet si le produit n'existe pas dans le commerce. Il va sans dire que, si c'est possible, il est toujours préférable de se procurer un convertisseur du commerce et de le modifier, qu'en cré un de toutes pièces. La difficulté essentielle est non seulement la conception d'un prototype, mais aussi de prévoir le fait qu'il doit fonctionner en milieu industriel. En définitive, on s'efforce de définir [4],[5] :

- La puissance apparente mise en jeu dans l'ensemble du système.
- Les tensions nominales prévues pour la source d'alimentation et d'utilisation.
- Les courants nominaux.
- Les composants de puissance normalement prévus pour le convertisseur.
- Les critères aboutissant au choix entre une commande analogique et une commande numérique.
- La procédure normale de mise en route ou l'arrêt d'urgence du système convertisseur associé à une charge.
- Les sécurités de fonctionnement de l'ensemble.

- l'action des divers modes de commande et de contrôle, en particulier le rôle des commandes par ordinateur ou par automate.
- L'importance des harmoniques de courant sur le réseau alternatif : on parle de « pollution » du réseau.
- L'action ou l'influence des signaux parasites : la CEM, la compatibilité électromagnétique de l'ensemble.
- Le coût de l'étude préalable et de la mise au point du prototype.
- Le coût financier de fabrication selon le nombre d'unités produites.
- Le coût d'entretien par unité.

Enfin, et surtout, il faut déterminer :

- S'il est possible d'utiliser un convertisseur déjà existant, et définir alors les réglages, voire les modifications à y apporter.
- S'il est indispensable de concevoir complètement un prototype à 100 %, avec tous les moyens d'étude, de conception (appareils, maquettes, logiciels…) correspondants.
- Si une solution de compromis entre les deux précédentes est envisageable.

1.3. Méthode de synthèse

Paradoxalement, ce qui semble le moins important à priori se révèle être le plus délicat. On pourrait penser que seule compte la puissance commandée et transférée, et le fonctionnement au niveau des signaux de commande. On ne s'intéresserait alors qu'aux composants de puissance et aux drivers qui permettent leur commande en commutation [4], [5], [6].

Les outils logiciels commerciaux existants sont seulement capables d'exécuter l'analyse, laissant la synthèse entièrement au concepteur. Par exemple, le " prototype virtuel " dans trois domaines temporels de conception est hors de portée du logiciel de Berkeley PSPICE [22], qui est capable d'effectuer des simulations électriques seulement. On peut citer également le logiciel Matlab qui peut effectuer des simulations couplées, toutefois vu les bases de temps utilisées en électronique de puissance cela devient impossible. En outre PLECS [21] et PSIM [23] n'exécutent que l'analyse des circuits de conversion de puissance. Si l'on veut effectuer des simulations dans le but d'une réalisation, l'ensemble des éléments constituant le

convertisseur (partie puissance, partie commande éloignée, partie commande rapprochée et l'ensemble des capteurs), sa source d'alimentation, le récepteur et l'environnement (problème de compatibilité électromagnétique) doivent être pris en considération, voire figure suivante.

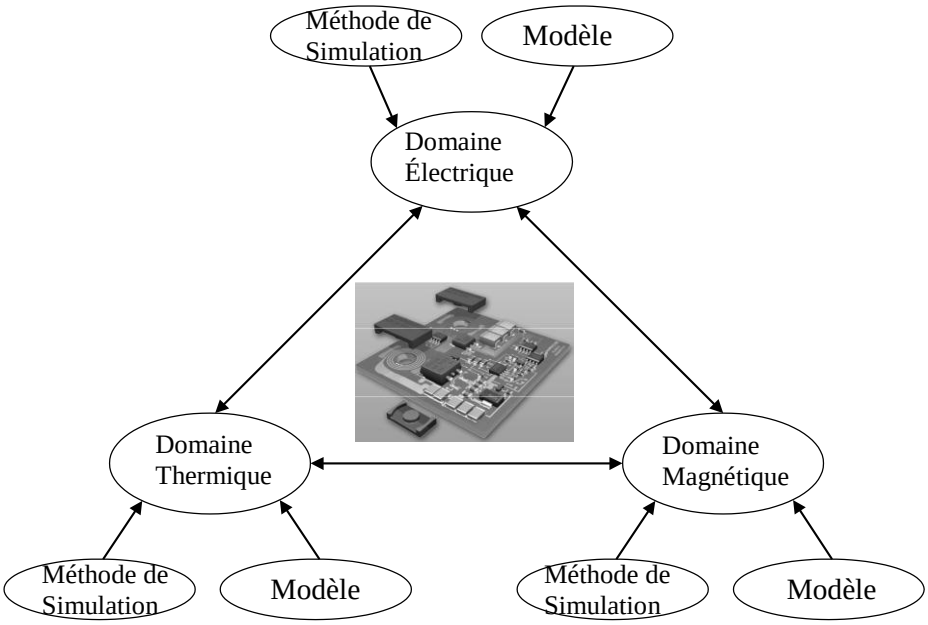

Figure 1.1 : Prototype virtuel « du convertisseur de puissance ».

A travers cette figure nous constatons que la commande éloignée où le microcontrôleur est le cerveau du convertisseur.

1.4. Les convertisseurs en fonctionnement réel

Pour la conception d'un convertisseur dédie à l'industrie, les contraintes sont encore plus exigeantes, la figure 1.2 schématise l'ensemble des protections à rajouter par rapport à l'exemple précédent [6].

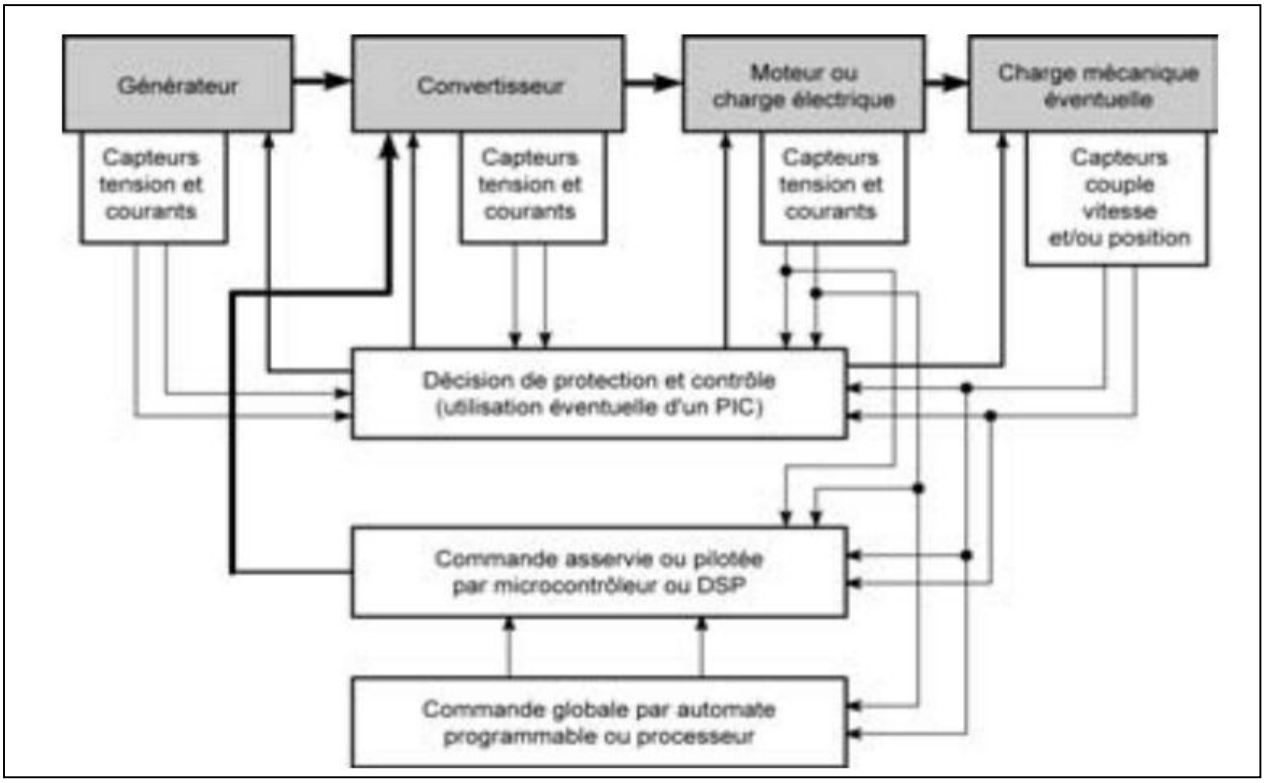

Figure 1.2 : Système utilisant un convertisseur en milieu industriel.

En électronique de puissance, un convertisseur fonctionne toujours en association avec d'autres systèmes électriques et mécaniques. On cherche à faire réaliser au convertisseur une fonction bien précise qui consiste à :
Commander soit analogiquement, soit numériquement par un processeur ou un automate, fonction qu'il s'agit de définir et de caractériser.

Les « sous-systèmes » d'un ensemble destinés à entraîner une charge mécanique présentent certains risques de défaillance en raison des fragilités dues à leur fonctionnement même.
Un sous-système électrique est fragile :

– Electriquement, dans le cas d'un mauvais isolement des bobinages, ce qui impose un contrôle au moment de la fabrication et lors de vérifications périodiques.

– Electriquement, dans le cas d'un mauvais isolement des câblages, ce qui impose un contrôle au moment de la fabrication et lors de vérifications périodiques.
De plus, si l'introduction de l'électronique dans les systèmes de commande et de contrôle des moteurs procure des avantages du point de vue souplesse et finesse de

réglage en vitesse ou en position, en contrepartie, les signaux utilisés dans les cartes électroniques sont souvent perturbés par les systèmes qu'ils sont censés commander (auto-parasitage). Parfois les cartes sont soumises à des signaux parasites externes qui vont rendre certaines commandes inopérantes. La compatibilité électromagnétique (CEM) est une approche de l'étude des systèmes qui permet de connaître « leur sensibilité » au parasitage.

– Mécaniquement, sur l'arbre, dans le cas d'un moteur : un dépassement de la limite d'élasticité de l'arbre de transmission provoque une déformation irréversible, qui rend le moteur inutilisable.

– Mécaniquement, dans le cas d'un moteur, sur le châssis et les pièces mobiles du stator (balais) : une commande par hacheur ou par onduleur provoquant une résonance peut être très nocive à long terme.

– Chimiquement dans le cas d'un fonctionnement d'une atmosphère explosible. Un convertisseur de puissance est fragile.

– Thermiquement, ce qui impose une surveillance permanente des pertes dans les composants électroniques et de l'effet Joule des conducteurs.

– Thermiquement, ce qui impose une surveillance permanente de l'effet Joule des conducteurs.

Enfin l'information concernant les grandeurs physiques importantes dans la connaissance d'un système est difficile à établir.

1.4.1. Détails du convertisseur

En toute rigueur, quelle que soit la conversion qui doit être assurée et la topologie du convertisseur, la cellule de commutation est suffisante pour décrire d'une manière rationnelle le fonctionnement des structures en électronique de puissance. C'est une approche puissante d'analyse permettant de dégager autour d'une commutation, les acteurs principaux. Initialement appliquée à la seule étude du fonctionnement global des convertisseurs, cette approche peut également trouver son application dans une étude de synthèse dans le but de réaliser un convertisseur respectant un cahier des charges bien défini, à condition de bien représenter tout l'environnement des semi-conducteurs. Ainsi, la figure 1.3 est complétée par le rajout de deux inductances parasites à savoir L_s et L_g, comme représenté par la figure 1.4.

Notre cellule se compose alors de deux boucles : la maille de puissance et le circuit de grille. Deux inductances couplées sont suffisantes pour se rendre compte de tous les phénomènes[6].

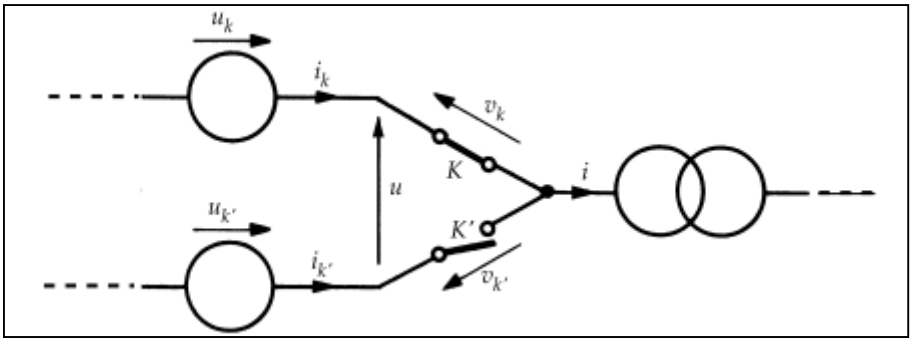

Figure 1.3 : Représentation généralisé de la cellule de commutation.

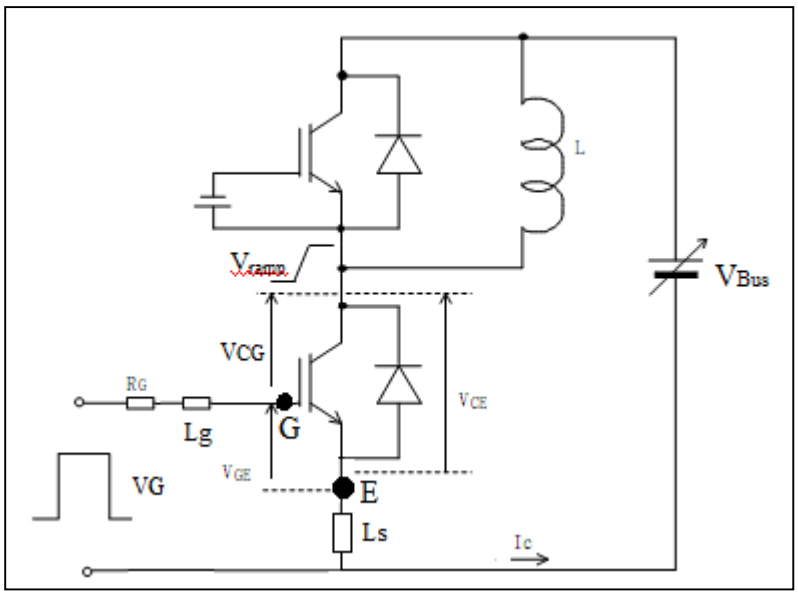

Figure 1.4 : Représentation particulière de la cellule de commutation.

Toutefois la partie puissance que représente la cellule de commutation n'est qu'un amplificateur de la partie traitement de l'information, c'est-à-dire commande éloignée, donc du microcontrôleur. Toute erreur provenant de ce dernier se verra amplifier. D'un point de vue pratique cela peut être extrêmement dangereux pour l'ensemble du processus schématisé sur la figue 1.2. La figure 1.4 doit être complétée pour tenir compte de l'ensemble des acteurs, de la partie convertisseur du schéma de la figure 1.2. La figure 1.5 représente le schéma global à considérer dans une approche de conception afin de prévoir tout l'ajustement à effectuer dans différentes parties constituant le convertisseur.

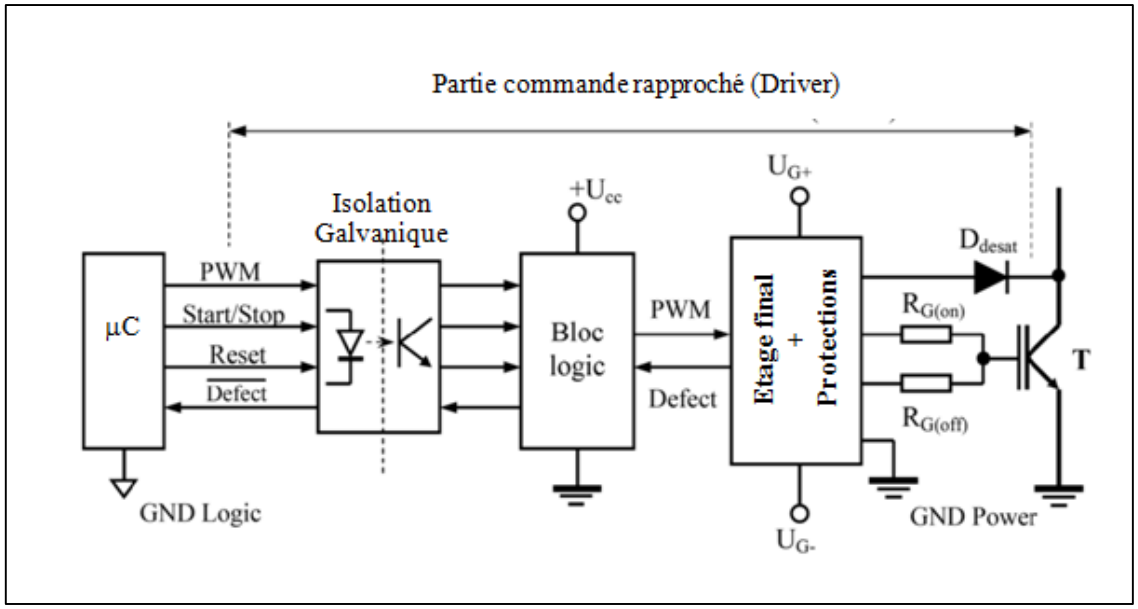

Figure 1.5 : Synoptique d'une commande d'une cellule de commutation [24].

1.4.2. Principe d'un convertisseur deux-niveaux

La majorité des convertisseurs statiques utilisent la Modulation de Largeur d'Impulsion (MLI ou PWM en anglais, pour Pulse Width Modulation). Le principe de fonctionnement consiste à connecter et à déconnecter deux sources de natures différentes, tension et courant, à un rythme rapide de période $T_{déc}$, afin d'avoir une valeur moyenne de tension ou courant souhaitée. Plus la proportion entre la durée de connexion et de non connexion des deux sources est grande, plus la puissance fournie à la charge sera importante. La phase de connexion est appelée dans ce manuscrit phase de transfert direct ou phase active, de durée $\alpha \cdot T_{déc}$; la phase de non connexion est appelée phase de roue libre, de durée $(1-\alpha)T_{déc}$. Les phases de transfert direct et de roue libre caractérisent les deux modes de fonctionnement des convertisseurs statiques. Pour la réalisation d'une telle cellule de commutation, nous n'avons besoin que d'interrupteurs fonctionnant en régime saturé/bloqué, imposant une tension quasi-nulle, ou un courant quasi nul. La caractéristique principale des convertisseurs statiques est donc de hacher le courant, côté haute tension, et la tension, côté basse tension.

1.5. Conclusion

Dans ce chapitre, une étude succincte d'une approche à des fins de réalisation d'un système de conversion d'énergie est faite. Une connaissance du matériel à utiliser est primordiale. La partie puissance des convertisseurs statiques est une amplification de leur partie commande. Ainsi, toute erreur provenant de cette dernière se verra donc amplifier, et éloigner ainsi le récepteur de sa tache fondamentale dans le meilleur des cas, ou provoquer des défaillances ou des désagréments allant jusqu'à la destruction de l'ensemble du dispositif.

Dans la suite de ce mémoire nous allons-nous intéresser à cette partie des convertisseurs. Le travail qui nous est confié est l'étude des possibilités d'utilisation des cartes Arduinos pour le contrôle des dispositifs de l'électronique de puissance.

Chapitre 2

Présentation

Des

Cartes Arduinos

2.1. Introduction

Avant d'entamer ce chapitre un petit historique concernant le développement des cartes Arduino nous semble nécessaire.

Historique

Le projet Arduino est issu d'une équipe d'enseignants et d'étudiants de l'école de Design d'Interaction d'Ivrée (Italie). Ils rencontraient un problème majeur à cette période (avant 2003 - 2004) : les outils nécessaires à la création de projets d'interactivité étaient complexes et couteux. Ces coûts souvent trop élevés rendaient difficiles le développement par les étudiants de nombreux projets et ceci ralentissait la mise en œuvre concrète de leur apprentissage.

En 2003, Hernando Barragan, pour sa thèse de fin d'études, avait entrepris le développement d'une carte électronique dénommée Wiring, accompagnée d'un environnement de programmation libre et ouvert. Pour ce travail, Hernando Barragan réutilisait les sources du projet Processing, basées sur un langage de programmation facile d'accès et adapté aux développements de projets de designers, la carte Wiring a donc inspiré le projet Arduino (2005).

L'objectif de toutes ces recherches était d'arriver à un dispositif simple à utiliser, dont les coûts seraient abordables, les codes et les plans « libres » (c'est-à-dire dont les sources sont ouvertes et peuvent être modifiées, améliorées, distribuées par les utilisateurs eux-mêmes) et, enfin, « multi-plates-formes » (indépendantes du système d'exploitation utilisé).

L'environnement Arduino est particulièrement adapté à la production artistique ainsi qu'au développement de conceptions qui peuvent trouver leurs réalisations dans la production industrielle.

Arduino est un projet qui a été fondé par une équipe de développeurs composé de six individus qui sont : *Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis et Nicholas Zambetti.*

Le nom Arduino trouve son origine dans le nom du bar '*di Re Arduino'* dans lequel l'équipe avait l'habitude de se retrouver. Arduino est aussi le nom d'un roi italien, personnage historique de la ville « Arduin d'Ivrée », ou encore un prénom italien masculin qui signifie « l'ami fort » [7], [8].

Sur la figure suivante, les différents cartes Arduinos que l'on trouve sur le marché sont représentées.

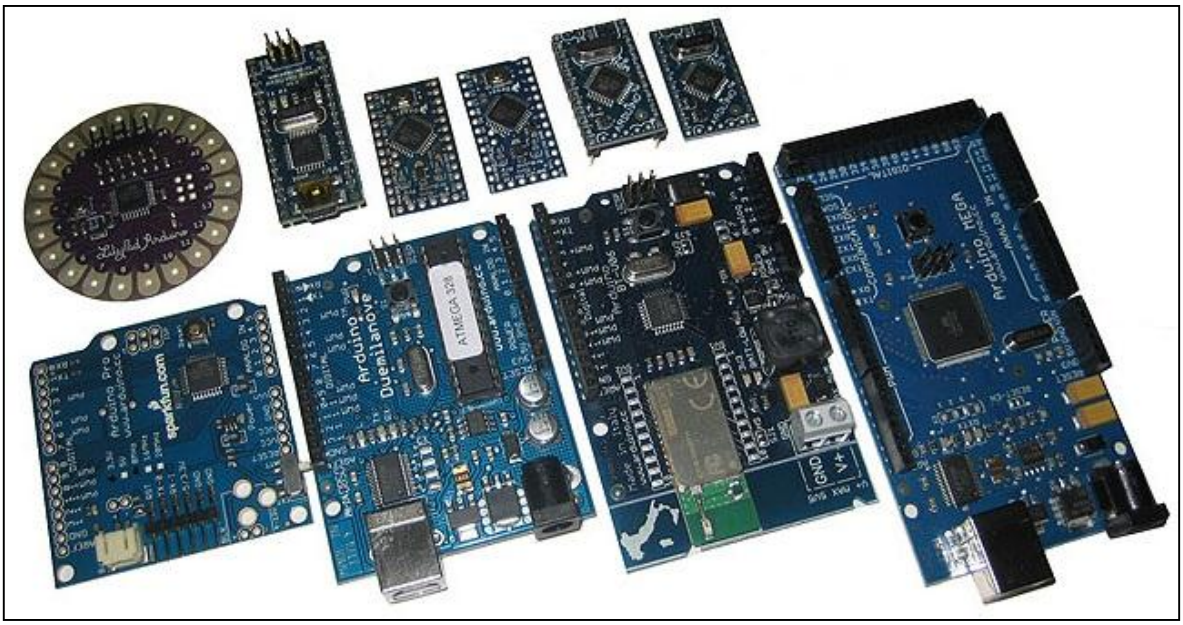

Figure 2.1 : Les différentes cartes Arduinos.

Parmi ces cartes nous allons nous intéresser uniquement à deux d'entre elles :

- Arduino UNO.
- Arduino DUE.

Le choix de ces deux cartes est dicté d'une part par la convivialité de la programmation concernant la UNO, et d'une autre part, par les performances concernant la DUE.

Le tableau comparatif des performances des cartes Arduino est donné en Annexe A.

Il est intéressant de mentionner aussi l'intérêt de la carte Arduino Mega 2560, celle-ci permet de travailler en temps réel sous Matlab/Simulink, concernant ces caractéristiques techniques elles sont identique à celle de la UNO.

De voire aussi les différents constituants de cette carte, et faire une étude succincte du microcontrôleur ATmel, car c'est la partie la plus importante de l'Arduino, aussi comment communiquer avec cette dernière. On va vous présenter le logiciel de simulation Fritzing pour Arduino.

2.2. Présentation de la carte Arduino

Arduino est l'association de deux branches, l'électronique et l'informatique (informatique embarquée) [8].

La carte Arduino est un circuit imprimé en matériel libre, les modules d'origine sont fabriqués par la société italienne SMART PROJECTS (dont les plans sont publiés en licence libre) sur lequel se trouve un microcontrôleur qui peut être programmé, afin d'analyser et de produire des signaux électriques, de manière à effectuer le contrôle de diverses tâches, dans différents domaines, comme la charge de batteries, la domotique (le contrôle des appareils domestiques comme, l'éclairage, le chauffage…), le pilotage d'un robot, la commande des actionneurs, etc.

C'est une plateforme basée sur une interface entrée/sortie simple. Pour la commande de charge à grande puissance « système d'électrotechnique » il faut un étage intermédiaire d'amplification de signal émis par la carte, car l'Arduino à une tension de sortie de 5V pour la UNO et au plus de 3,3V pour la DUE.

2.2.3. Différents éléments d'une carte

La figure suivante schématise les différentes parties essentielles constituant une carte Arduino UNO, néanmoins toutes les autres cartes sont construites avec la même architecture [9].

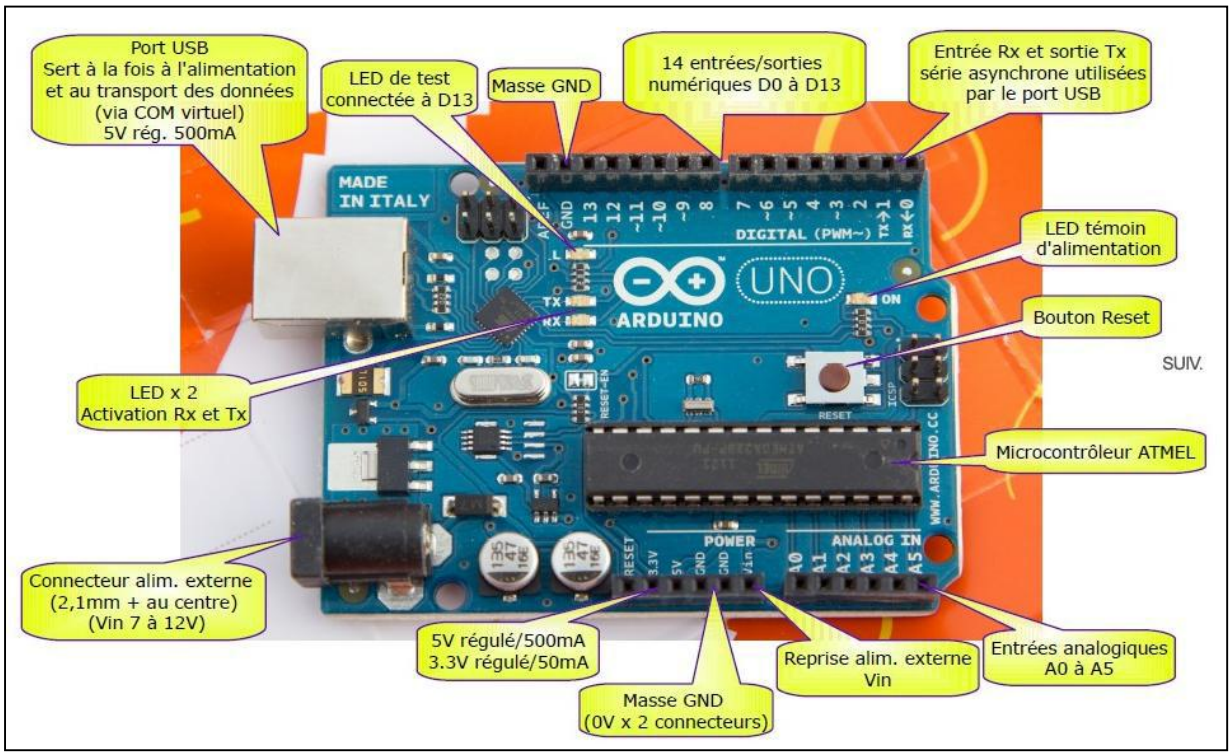

Figure 2.2 : Différentes parties de la carte Arduino UNO.

2.2.4. Arduino UNO

L'une des cartes que nous allons utiliser durant ce mémoire est la Arduino UNO. Elle est illustrée par la figure suivante [3], [9] :

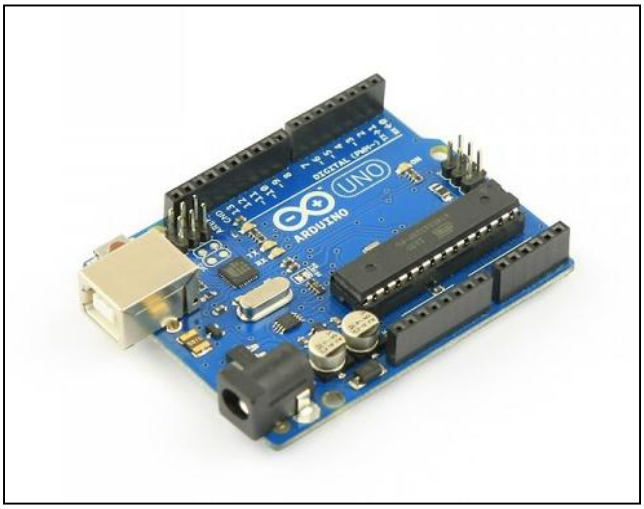

Figure 2.3 : Carte Arduino UNO.

2.2.4.1. Tableau de caractéristiques

Tableau 2.1 : Caractéristiques de l'Arduino UNO [3],[9].

Microcontrôleur	ATmega 328
Fréquence d'horloge	16 MHz
Tension de service	5 V
Tension d'entrée (recommandée)	7-12 V
Tension d'entrée (limite)	6-20 V
Ports numériques	14 E/S (6 sorties en MLI)
Ports analogiques	6 Entrées analogiques
Courant max par broche 3,3 V	50 mA
Courant max par broche 5 V	500 mA
Mémoire	➢ 32 Ko Flash ➢ 2 Ko SRAM ➢ 1Ko EEPROM
Dimensions	6,86 cm x 5,3 cm

2.2.4.2. Avantages

- Prix très abordables.
- Nombres de broches suffisants pour les projets élémentaires.
- Facile à utiliser, sa programmation est très accessible pour les débutants.

2.2.4.3. Inconvénients

- Manque de mémoire pour les gros projets.
- Insuffisance de broches pour les projets ambitieux.

2.2.5. Arduino DUE

La seconde carte utilisée dans notre travail est la Arduino DUE. A partir de la figure suivante on voit qu'elle est plus équipée que la UNO, qui est représenté sur la figure 2.3 [3],[10].

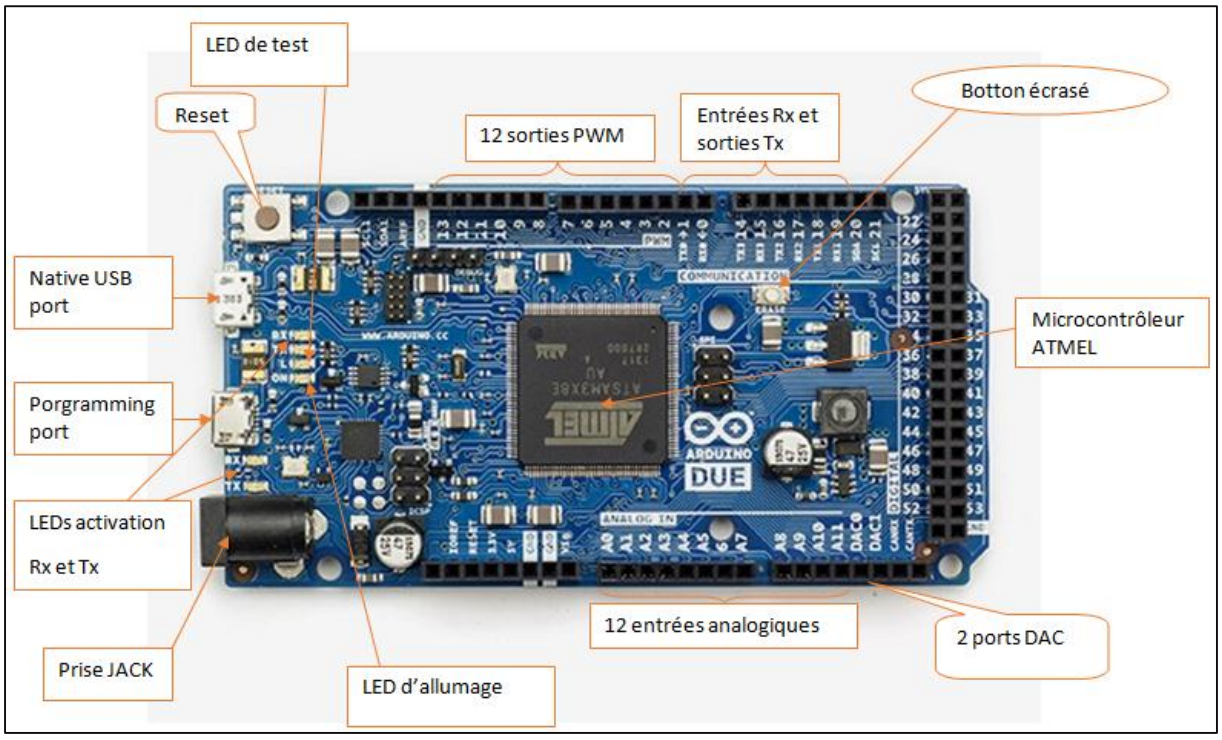

Figure 2.4 : Carte Arduino DUE.

2.2.5.1. Tableau de caractéristiques

Tableau 2.2 : Caractéristiques de l'Arduino DUE [3], [10].

Microcontrôleur	ATmega SAM3X8E
Fréquence d'horloge	84 MHz
Tension de service	3,3 V
Tension d'entrée (recommandée)	7-12 V
Tension d'entrée (limite)	6-20 V
Ports numériques	54 E/S (12 sorties en MLI)
Ports analogiques	12 E analogiques
Courant max par broche 3,3 V	800 mA
Courant max par broche 5 V	800 mA
Mémoire	➢ 512 Ko Flash ➢ 96 Ko SRAM ➢ 512 octets EEPROM
Dimensions	10,2 cm x 5,3 cm

2.2.5.2. Avantages

- De nombreuses entrées et sorties.
- Capacité mémoire suffisante pour les gros projets.
- Plus de broches MLI.
- Possède une PLL (boucle verrouillage de phase) pour callée les phases.
- Une meilleure résolution elle, possède un convertisseur analogique numérique à 12 bits.

2.2.5.3. Inconvénients

- Son prix est plus élevé que celui de la carte UNO.
- Tension de fonctionnement de seulement 3,3 V.
- Un peu plus complexe à manipuler comparé à la UNO.

2.2.6. Le Microcontrôleur ATmel

C'est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur, c'est lui le cerveau qui va traiter les données et les informations provenant des capteurs (entrées), cela avec un temps de traitement très réduit, ensuite donner les consignes souhaitées en sortie. Donc, c'est l'élément principal de la carte Arduino. La figure ci-dessous nous montre le schéma simplifié de la carte Arduino UNO donnée par la fiche technique [11],[12].

Nous constatons effectivement que le microcontrôleur est la pièce centrale.

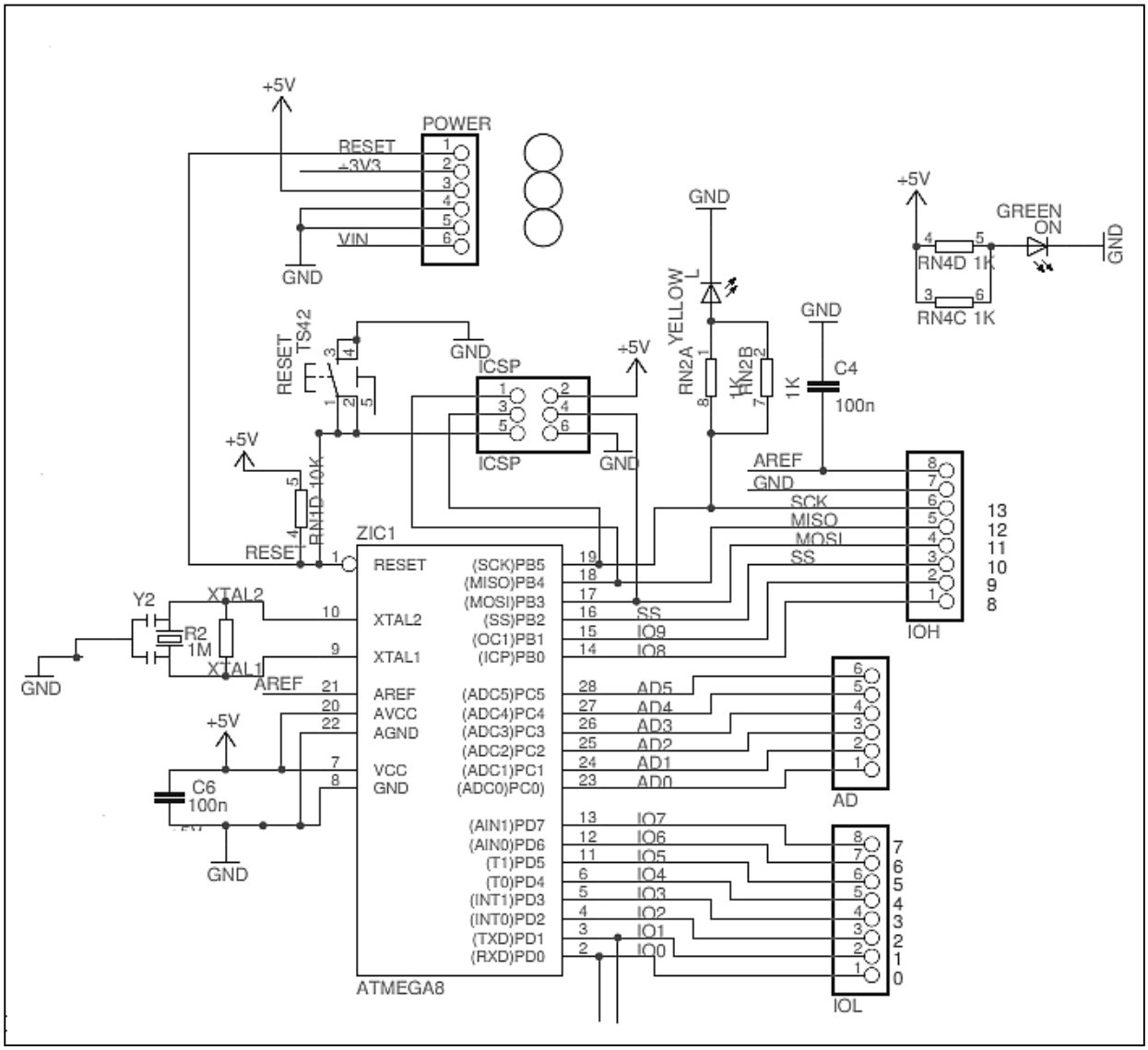

Figure 2.5 : Schéma simplifie de la carte Arduino UNO

Les éléments essentiels constituant un microcontrôleur sont :

- Unité centrale (CPU).
- ROM.
- RAM.
- Horloge interne.
- Ports d'entrée et de sortie.

2.2.6.1. Définitions des éléments d'un Microcontrôleur

- Mémoire Flash : c'est une mémoire de masse, donc même si on met l'appareil hors tension, les données ne sont pas perdues, car c'est un espace de stockage.
- Mémoire SRAM : c'est une mémoire volatile (qui perd ses données lorsqu'on coupe son alimentation électrique)
- Mémoire EEPROM : une mémoire non volatile (mémoire rémanente, c'est-à-dire qui conserve ses données même lorsqu'on coupe son alimentation électrique).
- Horloge interne : Elle est utilisée pour cadencer ou bien régler le microcontrôleur.
- Oscillateur : sa fonction est de produire un signal périodique, de forme sinusoïdale, carrée ou quelconque. L'oscillateur peut avoir une fréquence fixe ou variable.

Figure 2.6: Quartz (Horloge) de l'Arduino UNO.

Aujourd'hui, Le microcontrôleur est utilisé presque dans tous les domaines électriques, quel que soit le niveau de puissance et de fréquence. Nous allons citer quelques exemples où il est employé :

- La robotique.
- Les téléphones mobiles.
- L'électroménager.
- L'Automobile.

Avec la réduction des sources d'énergies fossiles, d'où la problématique des économies d'énergies, le microcontrôleur s'est imposé même dans le domaine des réseaux électriques via la gestion dans les « Smart grid ».

La figure 2.7 nous permet de mieux schématiser le principe de communication à l'intérieur d'un microcontrôleur et l'intérêt centrale du CPU dans le traitement de l'information.

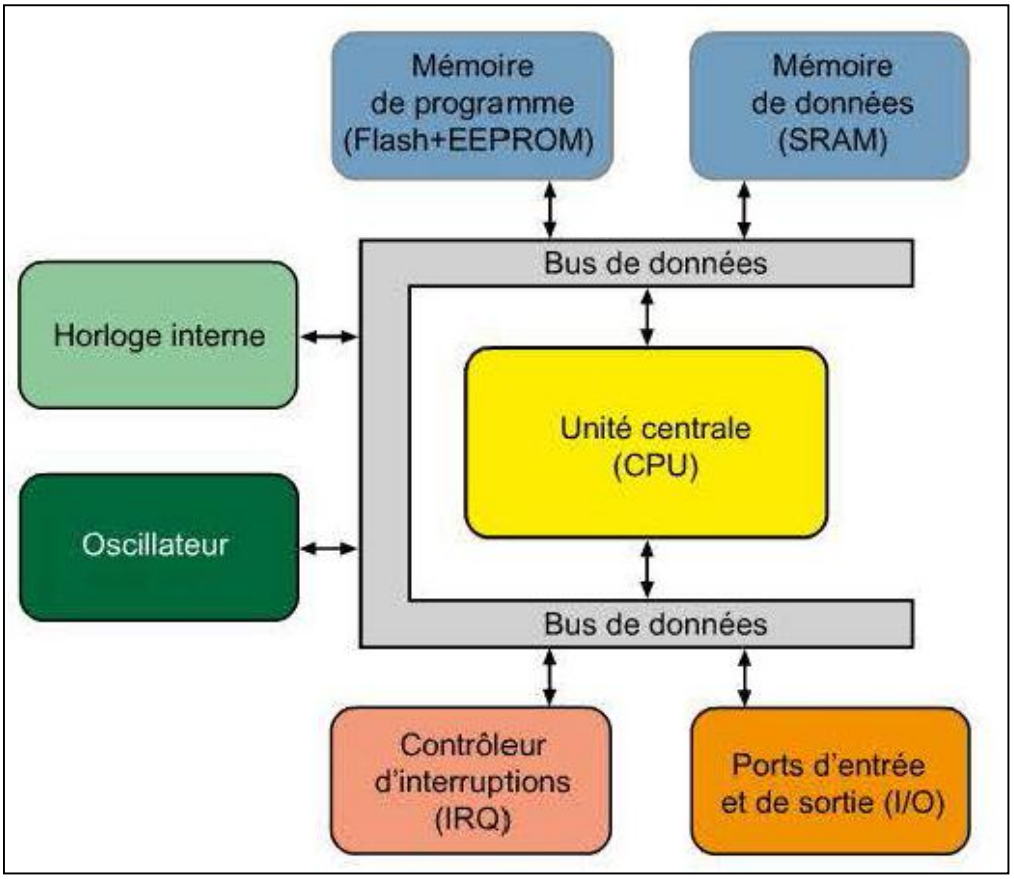

Figure 2.7 : Structure d'un microcontrôleur.

2.2.6.2. Le microcontrôleur de l'Arduino UNO

Le microcontrôleur de la Arduino UNO**,** est réalisé sur la base de ce qui a était dit précédemment. La figure ci-dessous représente la photo de l'ATmel ATMEGA328P le cerveau de la carte Arduino UNO, c'est grâce à ce microcontrôleur que toutes les tâches sont effectuées [3].

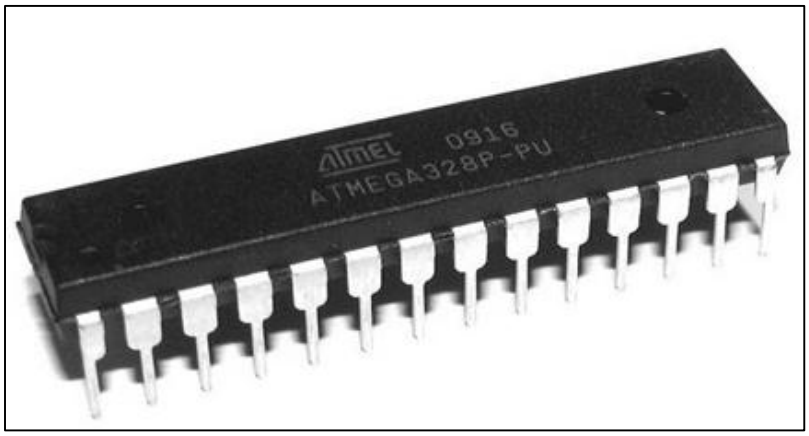

Figure 2.8 : Microcontrôleur de l'Arduino UNO.

2.2.6.3. Le microcontrôleur de l'Arduino DUE

Cette figure illustre également le microcontrôleur ATMEL ATSAM3X8E de la carte DUE, en comparaison avec la figure précédente, nous constatons qu'il est plus élaboré, par conséquent plus performant que celui de la UNO [3].

Figure 2.9 : Microcontrôleur de l'Arduino DUE.

2.2.7. Les entrées/sorties numériques ou Digitales

Les cartes Arduinos possèdent plusieurs broches numériques configurables en entrées et en soties selon les tâches assignées. Elles sont déclarées dans le programme avec le terme « digital ». Donc pour qu'une pin ou branche soit en sortie, dans le programme, il faut la déclarer avec l'instruction OUTPUT. Si elle est utilisée comme une entrée, il faut la déclarer avec l'instruction INPUT [2].

Le signal numérique ne peut prendre que deux états « 1 » correspond à l'état haut, en générale 5V pour la UNO et 3,3 pour la DUE, toute fois en fonction du montage à réaliser, il y a des chutes de tension, ou bien « 0 » correspond à l'état bas.

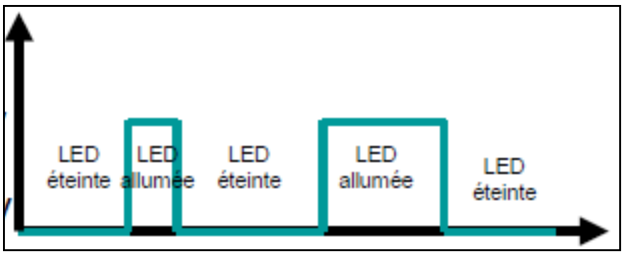

Figure 2.10 : Signal numérique.

2.2.8. Les entrées analogiques

Contrairement à un signal numérique qui ne peut prendre que deux états différents le signal analogique peut prendre une infinité de valeurs.

Sur la carte Arduino UNO, les broches A0 à A5 sont des entrées analogiques, elles sont dotées de convertisseurs analogique/numérique qui convertit la valeur de l'entrée en une suite de valeurs que la carte fait correspondre à un nombre variant entre 0 et1023, ce qui nous permet de récupérer les information d'un capteur [2].

Ces entrées analogiques sont déclarées dans le programme avec le l'instruction « analog ».

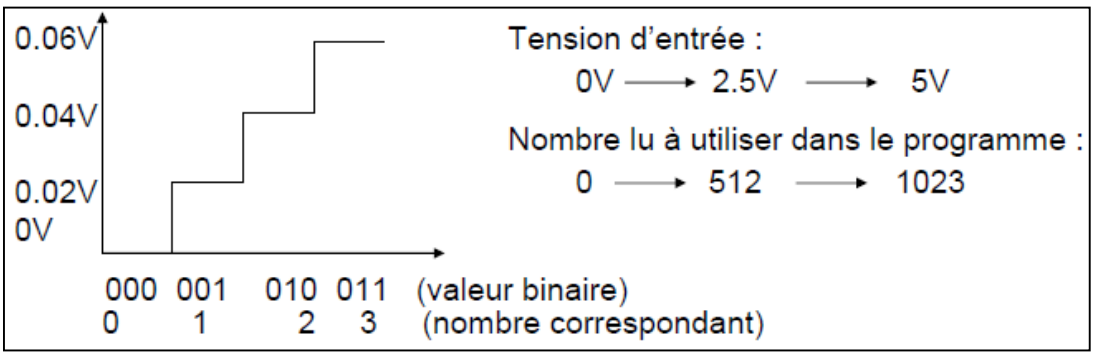

Figure 2.11 : Signal analogique.

2.3. Langage de programmation avec Arduino

Le langage de communication avec la carte Arduino est le logiciel Arduino, il est spécialement dédié à ces cartes, il est basé sur le langage évolué qui est le C++, car le microcontrôleur ne connait que le langage machine, composé de valeurs numériques [3].

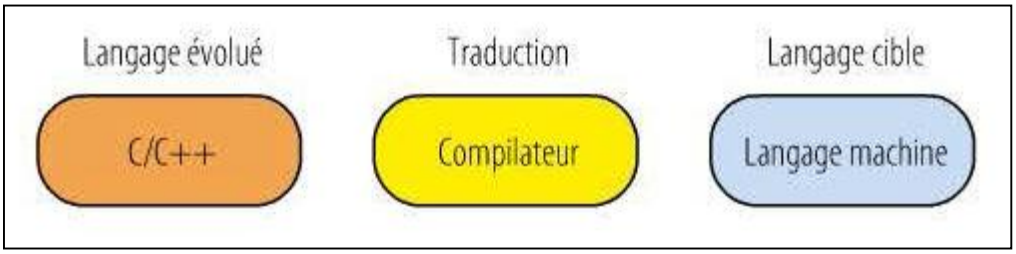

Figure 2.12 : Façons de communiquer avec Arduino.

2.3.1. La fenêtre de programmation

La programmation d'une carte Arduino sous le logiciel Arduino est toujours composée de trois parties [4], [10] :

1- Initialisation des variables.
2- Configuration des entrées/sorties, cette partie vient après l'instruction (void setup () {…..}).
3- Corps du programme ou on trouve les instructions et autres, cette partie vient après l'instruction (void loop () {……}).

La figure suivante présente une fenêtre du logiciel Arduino avec ses différents éléments.

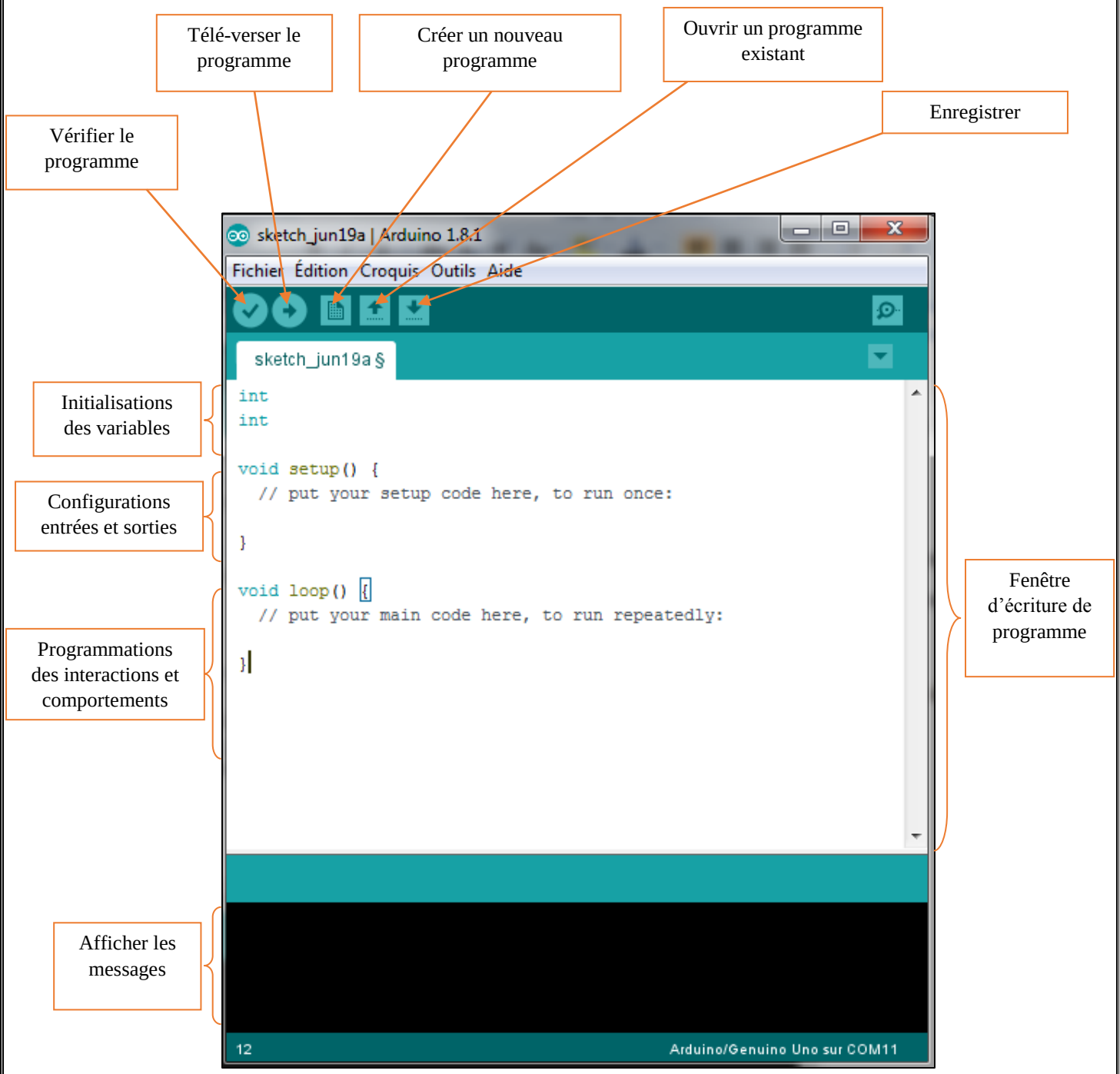

Figure 2.13 : L'interface de programmation de l'Arduino.

Alors, après création du programme il faut dans un premier temps, le vérifier ainsi corriger s'il y a des erreurs de programmation, puis le télé-verser dans la carte Arduino.

2.3.2. Exemple de programme des feux tricolore

La figure suivante illustre un programme Arduino des feux tricolores, avec les différentes opérations.

Figure 2.14 : Programme des feux tricolores.

2.3.3. Autres logiciels pour programmer les cartes Arduinos

D'autres logiciels nous permettent de communiquer avec les cartes Arduinos parmi lesquels, on peut citer le logiciel Fritzing et Matlab.

2.3.3.1. Logiciel Fritzing

C'est un logiciel qui a été développé par l'université des sciences appliquées de Potsdam en Allemagne, comme d'autres logiciels de simulation il nous apporte plusieurs avantages dans son contenu. Il facilite la création des circuits électriques en un temps impressionnant, car on dispose des différents types de cartes Arduinos et de tous les composants que nous pouvons utiliser pour nos applications [1], [3].

En plus il facilite la création des schémas électriques et la fabrication des circuits imprimés à partir des informations recueillies sur le circuit électrique, et la saisie des codes qui sert à commander l'Arduino.

Donc ce logiciel de simulation nous permet de voir les résultats finaux avant de passer en pratique, ce qui aide beaucoup dans le prototypage, la compréhension des phénomènes et dans l'industrie.

Dans les figures suivantes nous allons présenter l'interface de logiciel Fritzing.

2.3.3.1.1. Platine d'essai

La figure ci-dessous illustre la simulation des feux tricolores sous la platine d'essai avec le logiciel de simulation Fritzing, en utilisant les différents composants dont on aura besoin d'une façon tout à fait virtuelle.

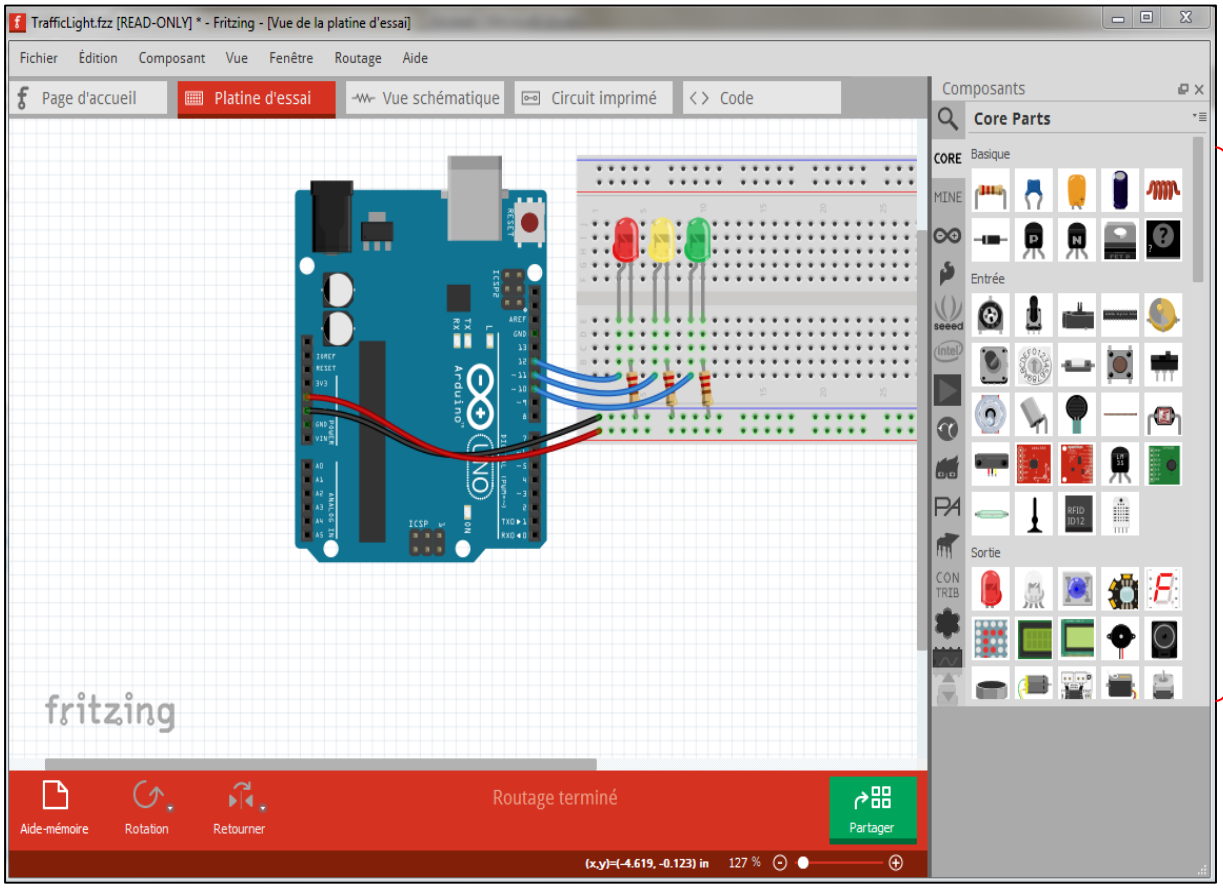

Figure 2.15 : Feux tricolores sous la platine d'essai de Fritzing.

2.3.3.1.2. Vue schématique

La figure suivante illustre le même exemple mais cette fois sous forme d'un schéma.

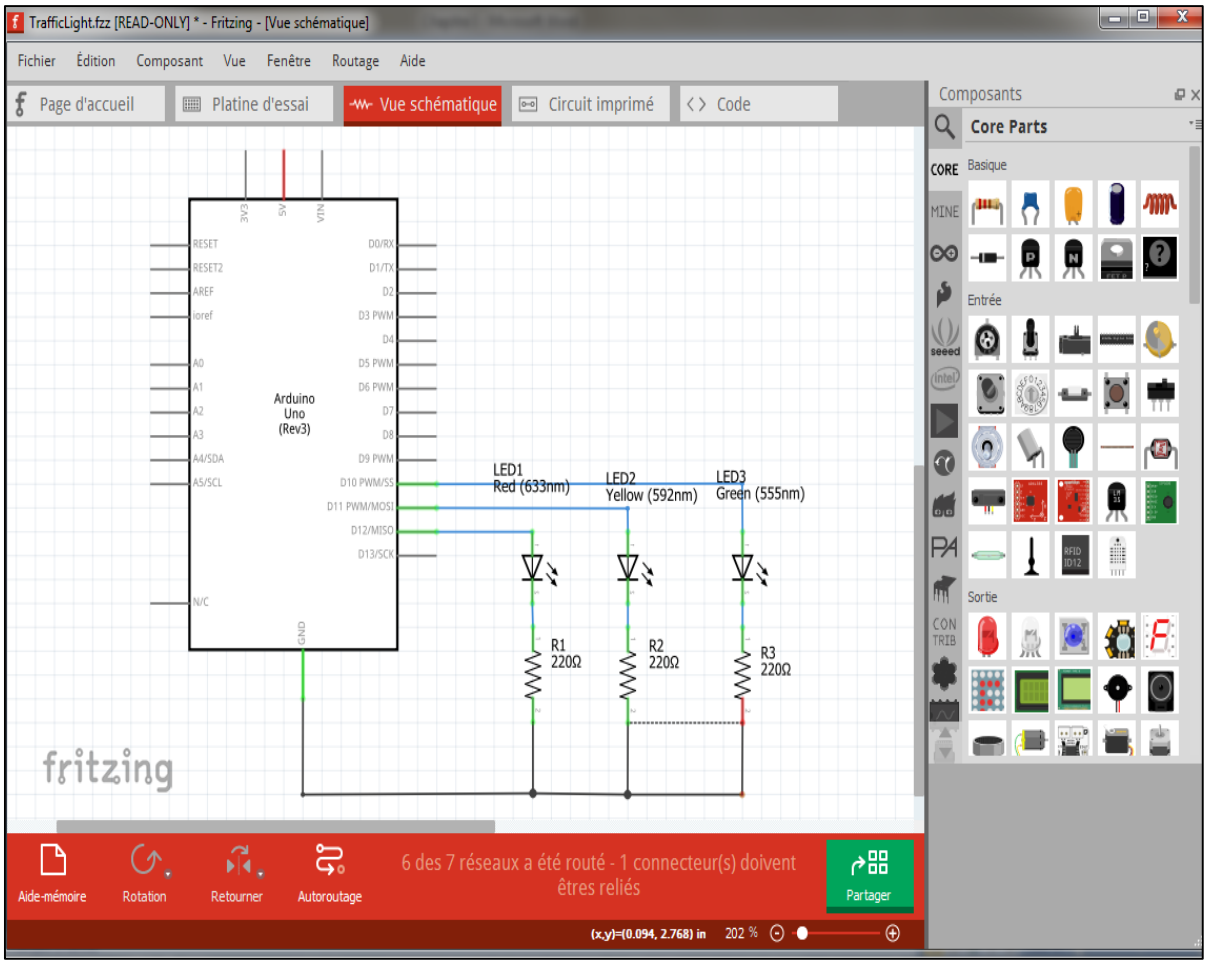

Figure 2.16 : Vue schématique de l'application précédente.

2.3.3.1.3. Circuit imprimé

La figure suivante illustre le même exemple en circuit imprimé.

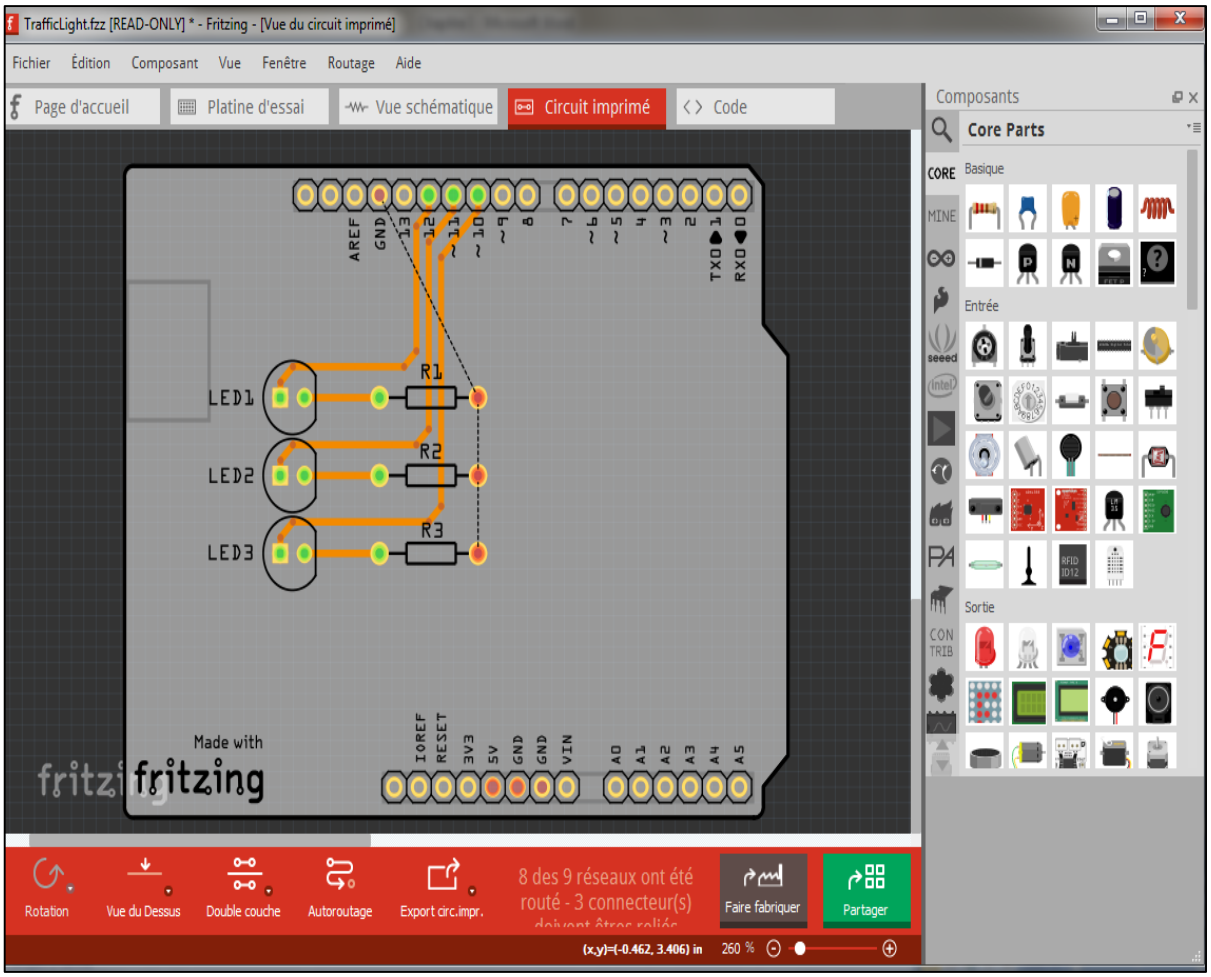

Figure 2.17 : Circuit imprimé de l'application précédente.

2.3.3.2. Logiciel Matlab

Comme la majorité des autres cartes, les cartes Arduinos communiquent avec Matlab et Matlab/Simulink. Une partie du travail de ce mémoire est de bien analyser les performances des cartes Arduinos en communication avec Matlab, en conséquence un développement plus détaillé fera l'objet du chapitre suivant.

2.4. Conclusion

Dans ce chapitre nous avons présenté succinctement l'ensemble des cartes Arduinos et légèrement plus en détails les deux cartes que nous allons utiliser dans les chapitres suivants.

On a également énuméré l'essentiel des logiciels avec lesquels on peut programmer ces cartes, dans ce qui suit nous allons nous intéresser aux performances et précisions de ces cartes. La problématique que nous chercherons à résoudre est la programmation des techniques de modulation de largeur d'impulsions.

La modulation de largeur d'impulsions constitue un problème crucial d'un point de vue programmation en temps réel à cause des constantes de temps mises en jeux, elles sont de l'ordre de 1e-5.

Les logiciels que nous utiliserons seront Matlab/Simulink et le logiciel Arduino.

Chapitre 3

Matlab Simulink

Et MLI

Avec

Les cartes arduinos

3.1. Introduction

Le logiciel Matlab est un outil de calcul numérique et de visualisation graphique, il est très utilisé dans les laboratoires de recherche. Il est utilisé par les ingénieurs, les techniciens et surtout par les scientifiques quel que soit leur domaine.

C'est un outil qui est employé soit par les établissements universitaires ou bien dans les unités de recherche et développement chez les industriels.

Dans les versions précédentes de Matlab antérieure à 2012, la bibliothèque Arduino n'est pas incluse dans Simulink, à partir de 2012, les éléments concernant la UNO et la Mega2560 ont commencé à être intègres, puis avec le temps toutes les cartes Arduinos ont été ajoutées, à l'aide de ce logiciel on peut établir des schémas de commandes et d'acquisitions de données.

Ce chapitre est consacré à la présentation des étapes à suivre pour exploiter la carte Arduino avec Matlab, ensuite, générer un signal MLI.

3.2. L'interface Arduino et Matlab/Simulink

On dispose de deux possibilités pour communiquer la carte Arduino avec Matlab [11], [14].

- Utilisation du la library Arduino IO ;
- Utilisation du package Arduino.

3.2.1. Library Arduino IO

Cette méthode consiste à utiliser la carte comme une interface d'entrées (analog Input) et de sorties (Analog/ Digital Output), Cette échange d'information se fait à travers un câble USB via le port série « Com » du PC.

3.2.1.1. Procédure de configuration du la carte

Avant toute manipulation sur Matlab on doit d'abord charger un programme dans notre Arduino afin qu'elle puisse fonctionner en serveur, en suivant les étapes suivantes :

- Télécharger Arduino IO ;
- Le Décompresser ;
- Ouvrir le dossier après décompression ;
- Aller vers Arduino IO\pde\adiosrv ;
- Charger le fichier 'adiosrv' dans le logiciel Arduino puis le télé-verser dans la carte.

Après ces étapes, la carte est prête pour être utilisée comme une interface.

3.2.1.2. Installation de la library Arduino IO :

Après avoir lancé Matlab on exécute 'Install-arduino' qui se trouve dans le fichier décompressé auparavant, pour la prise en charge de cette library sous Matlab, il faut relancer le PC, une exploitation de la base de données Arduino IO dans Simulink est ensuite possible. Tous les éléments nécessaires pour élaborer des programmes sous Matlab/Simulink sont disponibles. La figure suivante détaille les éléments de cette Library.

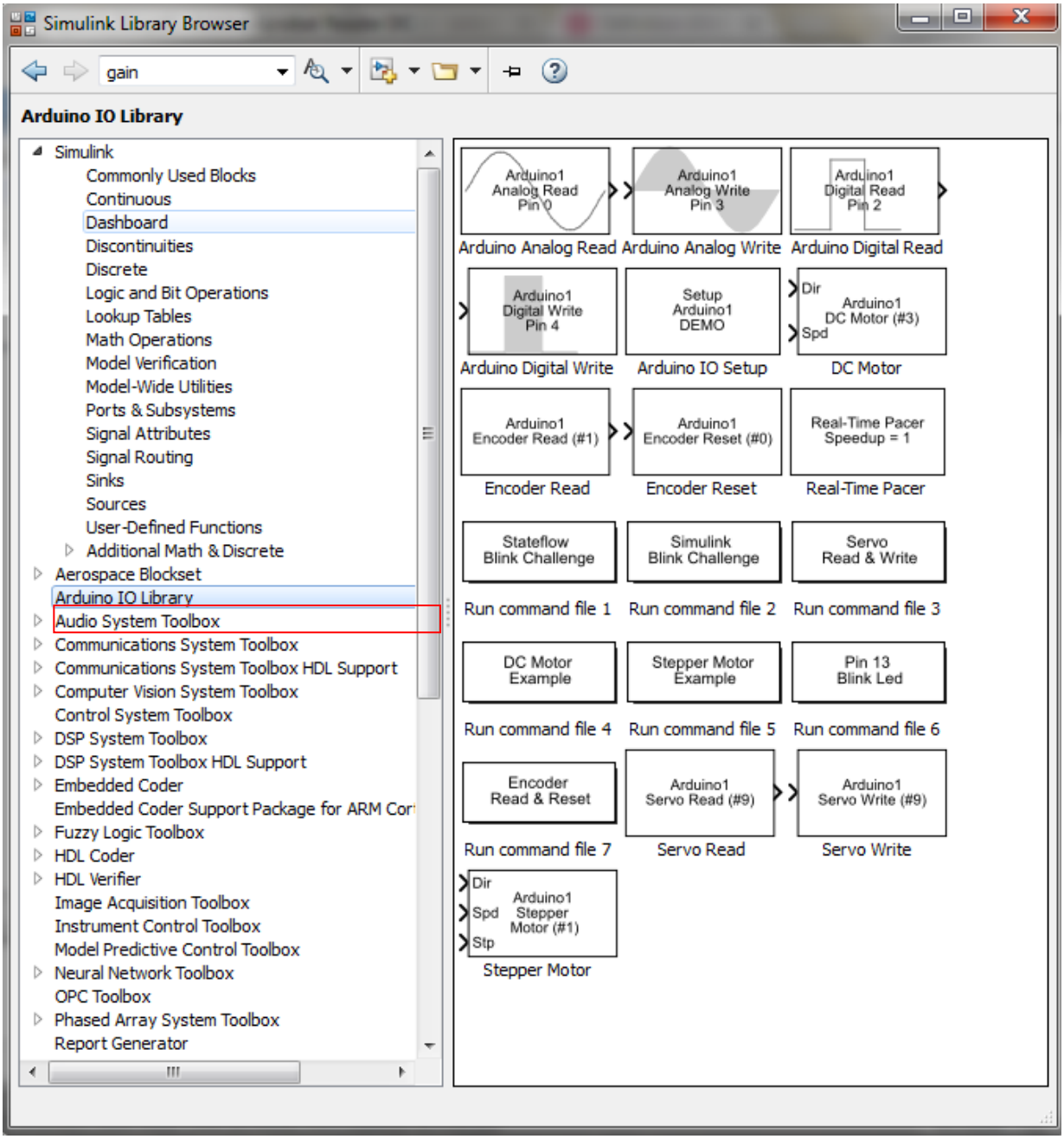

Figure 3.1: Arduino IO library.

3.2.2. Package Hardware for Arduino

La deuxième possibilité est l'utilisation du pack hardware for Arduino, ce pack est disponible uniquement via internet. Avant toute chose il faut disposer d'une bonne connexion internet, ensuite créer un compte sur le site Mathworks. Pour installer cette librairie il faut suivre la procédure suivante :

- Cliquer sur Add-Ons dans la fenêtre de Matlab Windows.
- Choisir Get Hardware Support Packages.
- Choisir Install from internet et en appuyant sur next cette fenêtre apparait.

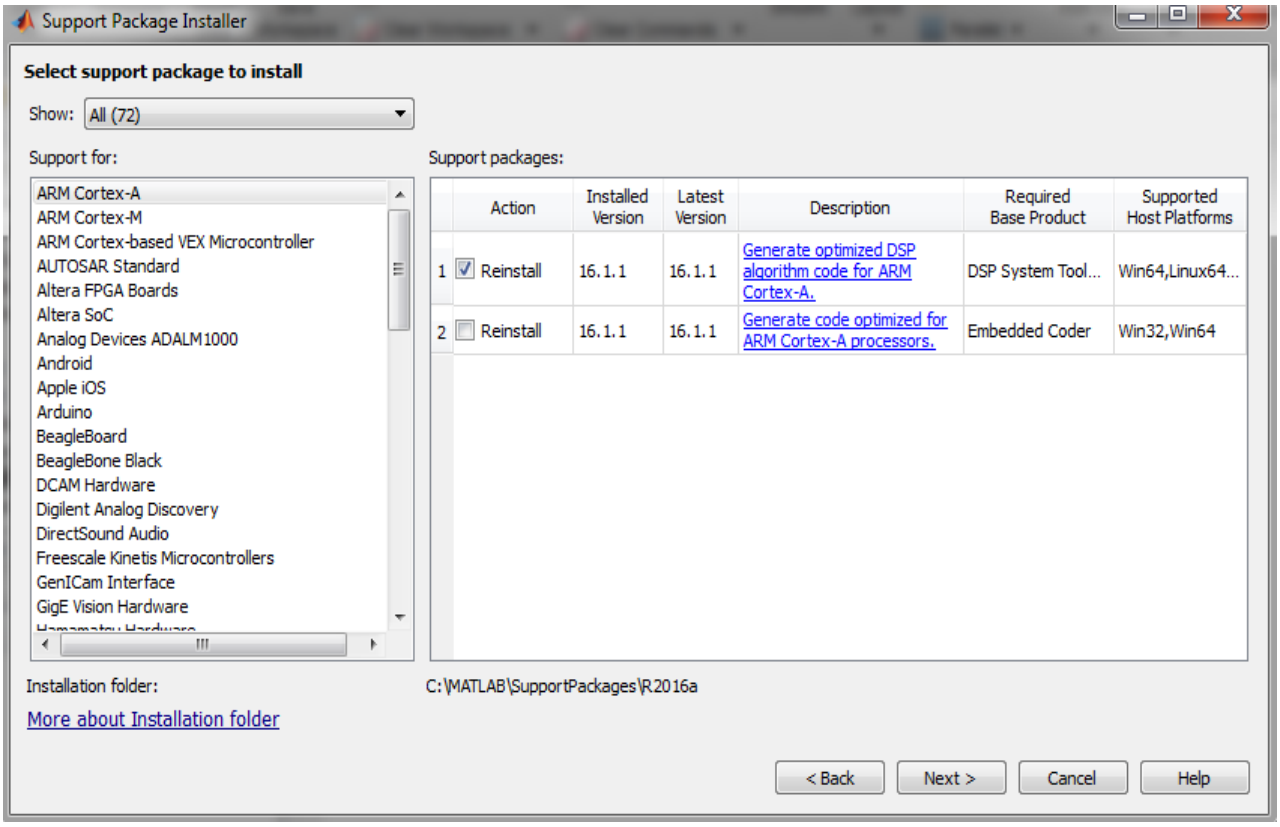

Figure 3.2: Différents packs qu'on peut installer.

- Choisir le pack Arduino à installer ;
- Ensuite entrer les cordonnées de notre compte Mathworks et appuyer sur 'Log in'.

A la fin de l'installation on trouve le Support Package de Arduino dans Simulink avec tous les éléments nécessaires. La figure suivante illustre les éléments de ce Pack sous Simulink.

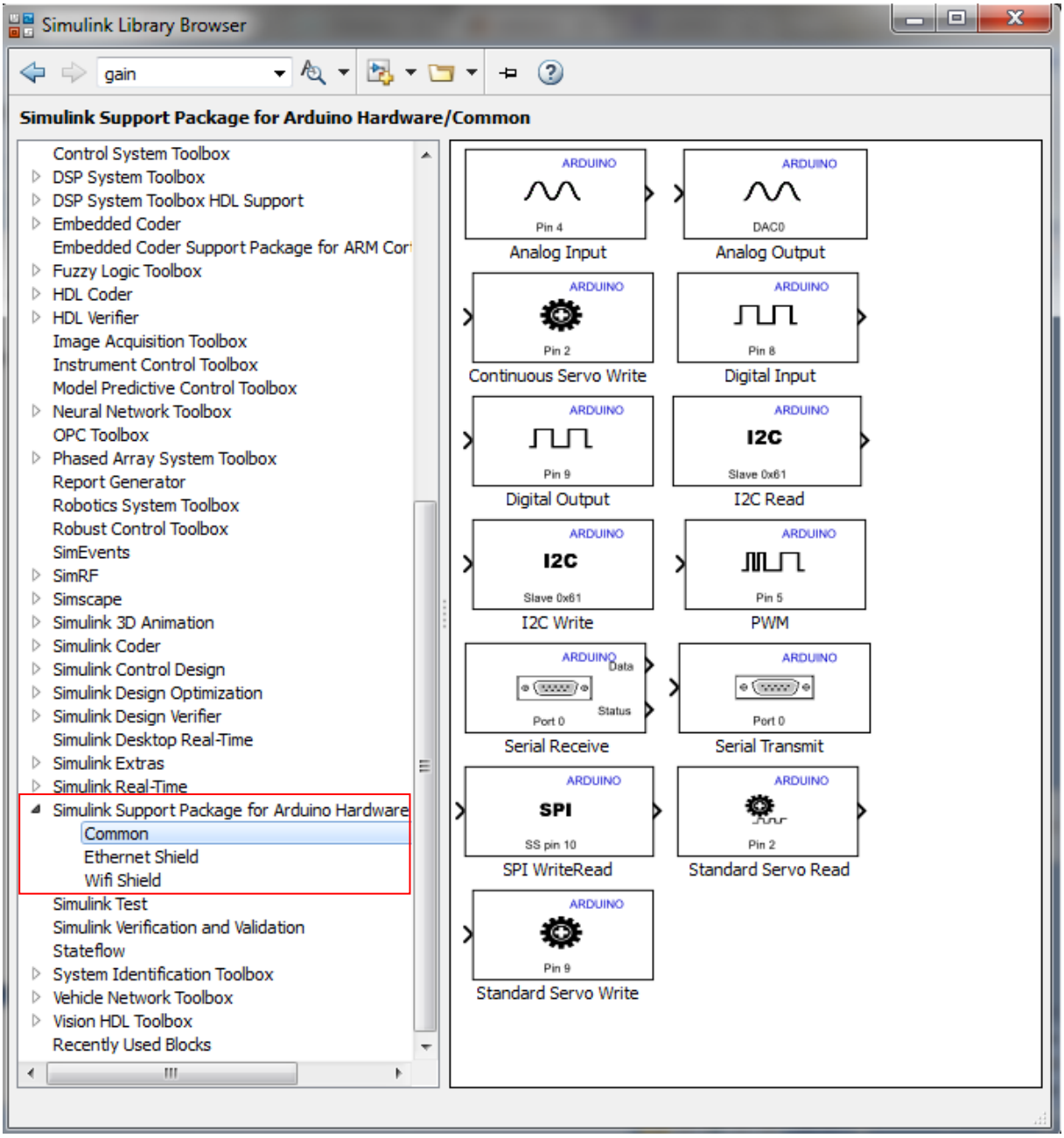

Figure 3.3 : Support Package des cartes Arduino.

3.3. Générer un signal MLI sous Matlab

La Modulation de Largeur d'Impulsion (Pulse Width Modulation) en anglais, consiste à générer un signal carré avec un rapport cyclique modulé en fonction du signal de référence souhaité. Les signaux résultant de cette comparaison sont utilisés pour commander les semi-conducteurs utilisés en électronique de puissance. Ils sont aussi dit circuits de puissance à découpage (exemple d'un pont H).

3.3.1. Définition de la MLI

Le principe de la Modulation de Largeur d'Impulsion tel qu'illustré par la figure ci-dessous, consiste en une comparaison d'une onde modulée basse fréquence, grandeur de référence ou bien consigne, elle est généralement de forme sinusoïdale ou constante, avec une onde porteuse haute fréquence de forme quelconque. Le but de cette technique de commande est d'envoyer les harmoniques de sorties vers les hautes fréquences afin de faciliter leurs filtrages et bien évidement de réduire la taille des filtres et leur coût. Par conséquent avoir un convertisseur avec un rapport qualité /prix intéressant [6],[7].

Eléments caractérisant une MLI :

f : Fréquence de la consigne.

f' : Fréquence de la porteuse ou de découpage.

Donc : $f < f'$

$m = \frac{f'}{f}$: L'indice de modulation, en fonction de la MLI utilisée, il nous renseigne sur le spectre d'harmoniques des grandeurs de sortie.

$$r = \frac{U_{Amp\_C}}{U_{Amp\_P}}$$

Et le coefficient de réglage « r », c'est le quotient de l'amplitude du signal de la porteuse sur celle de la consigne. Il nous renseigne sur l'amplitude du fondamental du signal de sortie, dans le cas des grandeurs alternatives. Si on a une conversion continue, il s'agit de la valeur moyenne, ce coefficient est aussi appelé rapport cyclique.

Avec :

U_{Amp_C} : Amplitude du fondamental de la consigne.

U_{Amp_P} : Amplitude du fondamental de la porteuse.

La figure suivante schématise le principe de cette technique de commande.

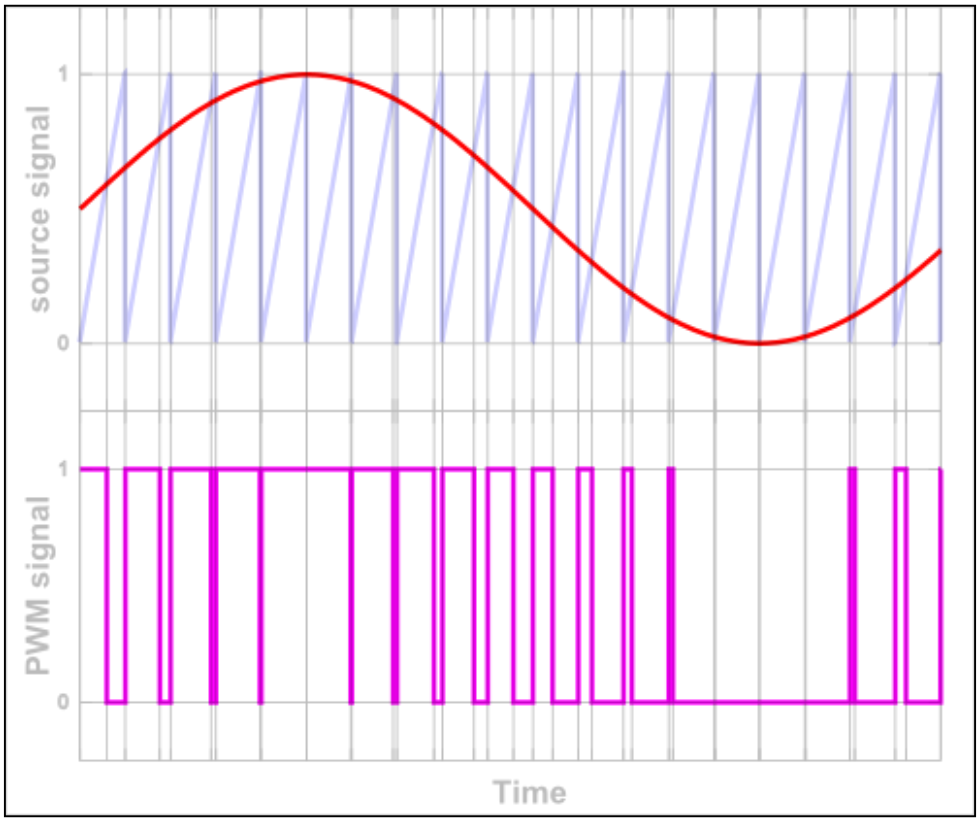

Figure 3.4 : Principe de la MLI.

3.3.2. Partie pratique

Dans cette partie nous allons implanter le programme MLI sous Simulink dans le but de le télécharger sur les cartes Arduinos UNO et la DUE. La référence sera de forme sinusoïdale, elle est importée de l'extérieur. Pour générer le signal de référence on a utilisé un synthétiseur de fréquence, qui est une source d'alimentation purement sinusoïdale de fréquence et d'amplitude variable. L'acquisition de cette référence directement avec les cartes Arduinos n'est pas possible, car elle ne tolère que des tensions allant de 0 à 5 V pour la UNO et de 0 à 3,3V pour la DUE. Une adaptation est donc nécessaire. La figure suivante montre le schéma

block sous Simulink pour générer une commande MLI avec une référence sinusoïdale, elle est importée par le bloc « Analog input ».

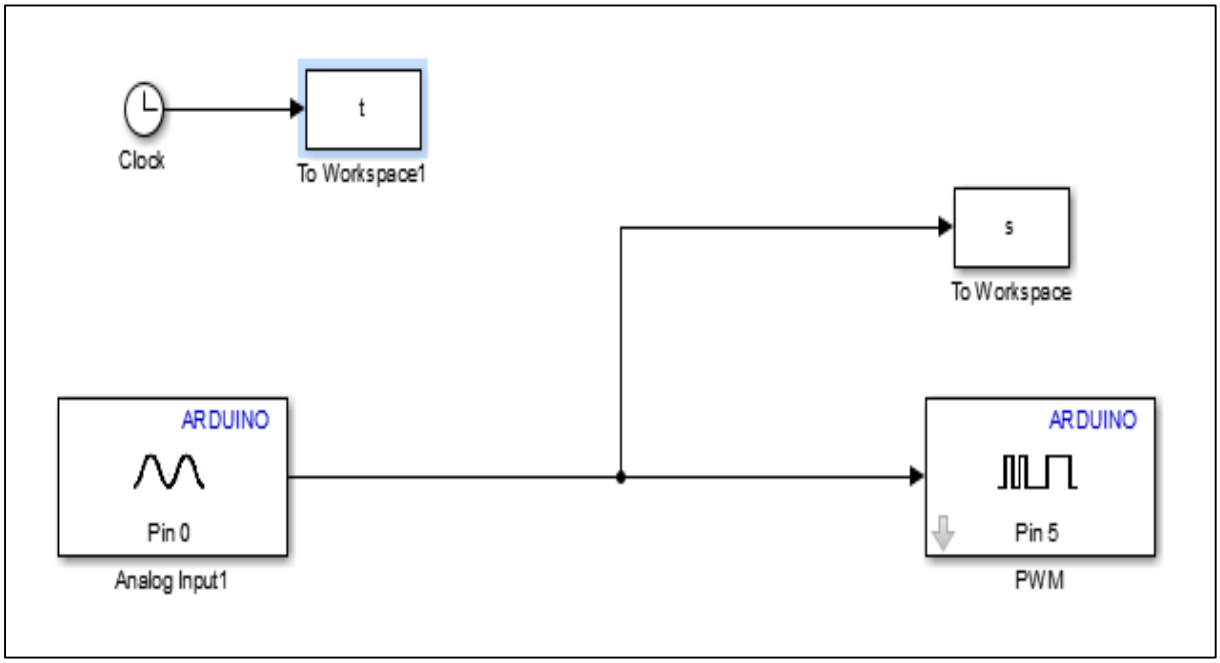

Figure 3.5 : Schéma Simulink pour générer une MLI avec Arduino.

3.3.2.1. Description du dispositif expérimental

Ce synthétiseur de fréquence nous permet de générer un signal purement sinusoïdal de fréquence et d'amplitude variable. La figure3.6.a est une photo de notre synthétiseur et la figure3.7.b schématise le signal délivré en sortie. Cette appareil nous a été très utile du fait de la qualité de son signal de sortie.

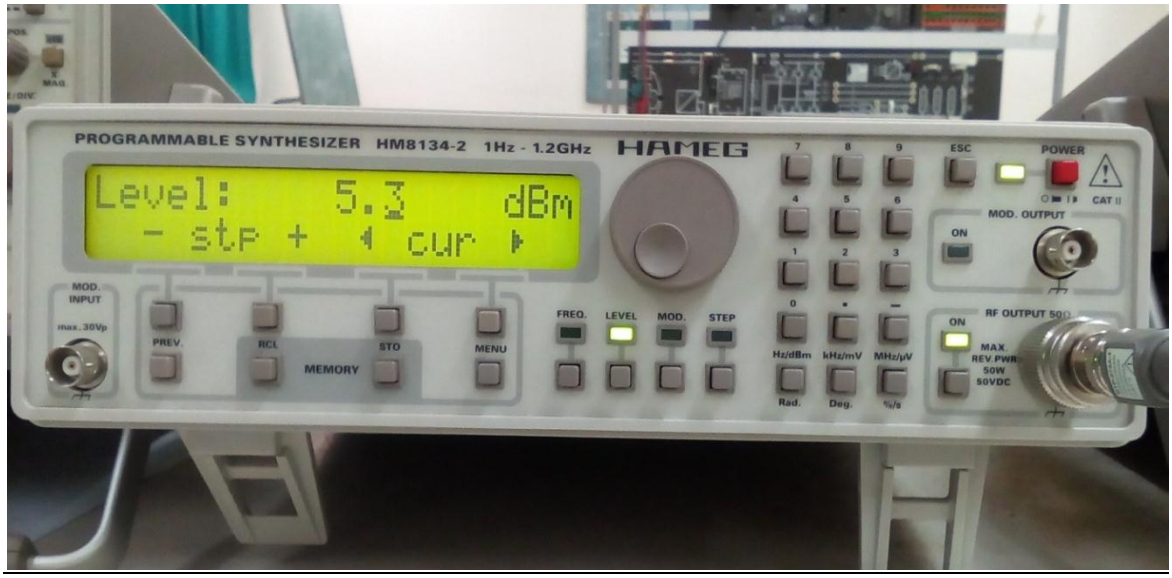

Figure 3.6.a : Synthétiseur de fréquence.

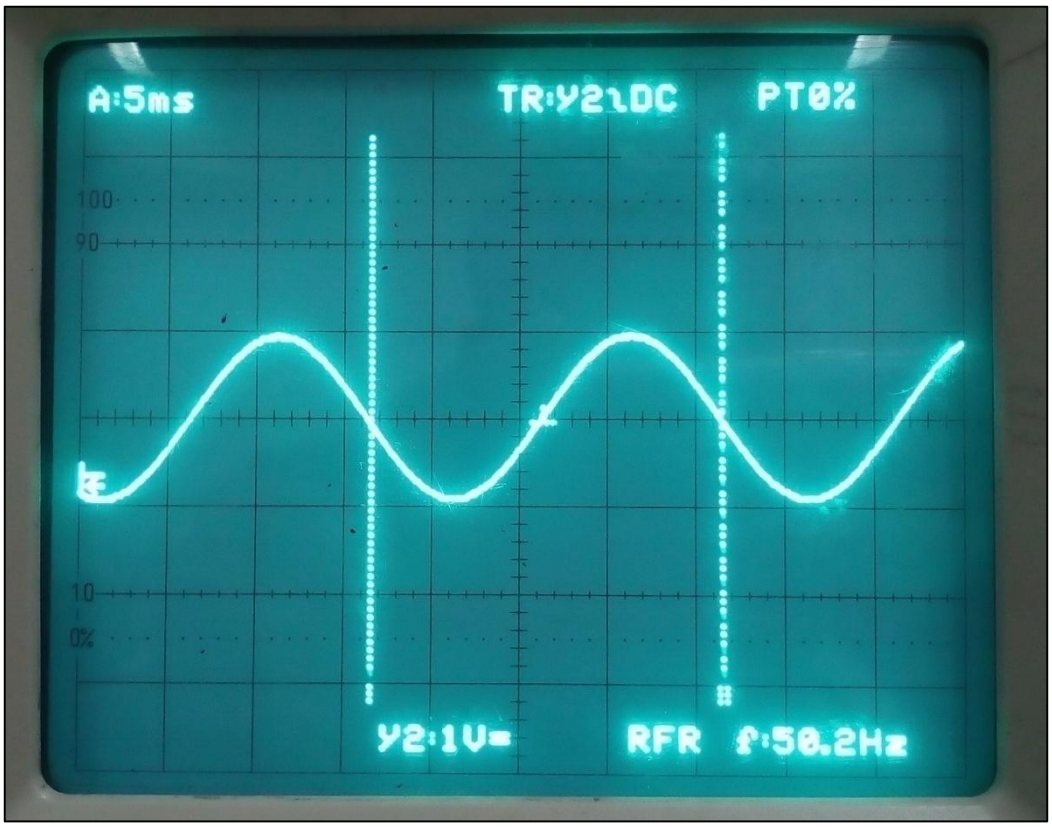

Figure 3.6.b : Signal délivré par notre synthétiseur.

A cause de la nature alternative du signal à la sortie du synthétiseur, il est nécessaire de le décaler pour pouvoir l'injecter dans les cartes Arduinos. Ce décalage devait être égal à (5/2) V pour la UNO et de (3,3/2) pour la DUE. Le montage réalisé est celui donnée sur la figure 3.7, il est alimenté par deux alimentations stabilisées, dont l'une permet d'alimenter le circuit

additionneur, et l'autre permet de régler l'offset souhaité de la référence, afin d'avoir en sortie du circuit additionneur une fonction de forme sinusoïdale purement positive de valeur moyenne égale à son amplitude.

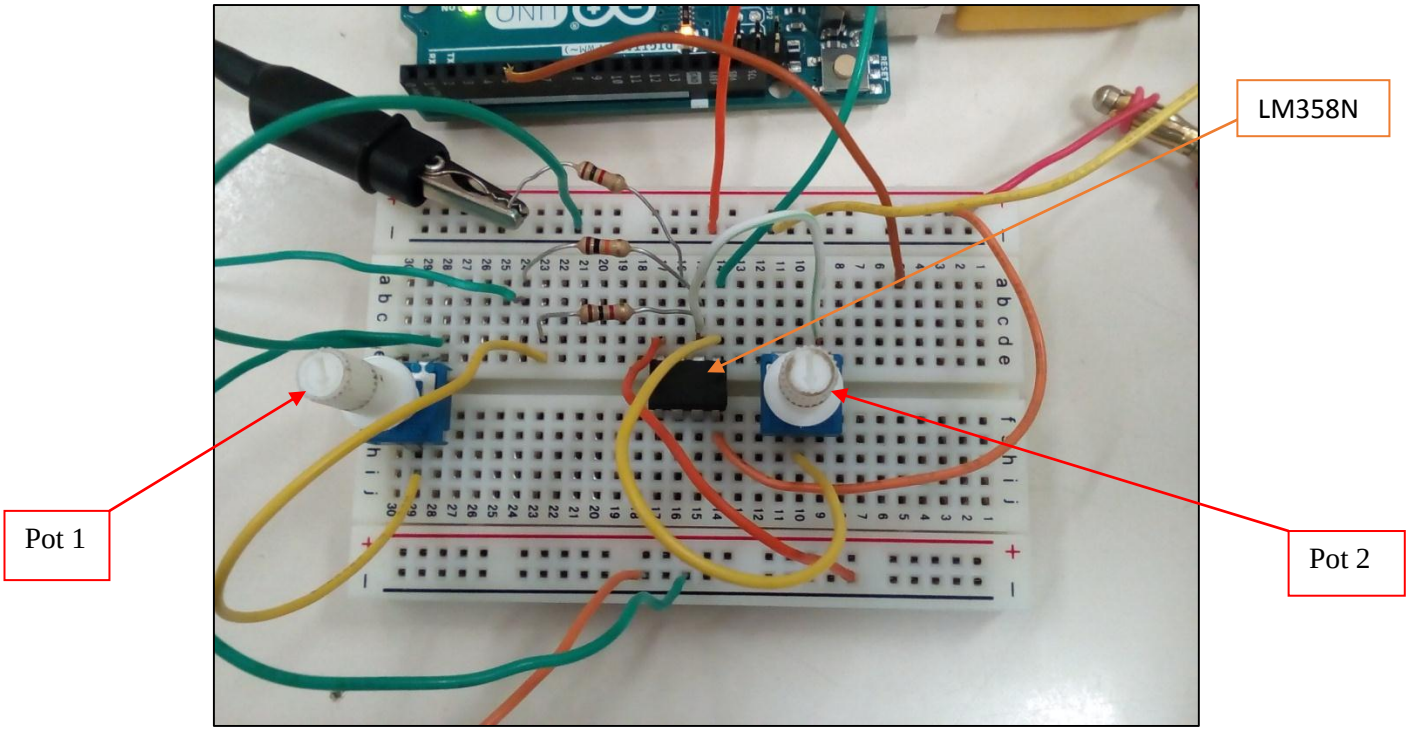

Figure 3.7: Circuit additionneur.

Le potentiomètre 1 (Pot 1) permet de régler la composante continue à additionner au signal de référence délivré par le synthétiseur.

Le potentiomètre 2 (Pot 2) permet de régler l'amplitude du signal de référence de sortie de circuit additionneur, en réalité c'est un amplificateur opérationnel, donc il nous permet le réglage sans passer par le synthétiseur.

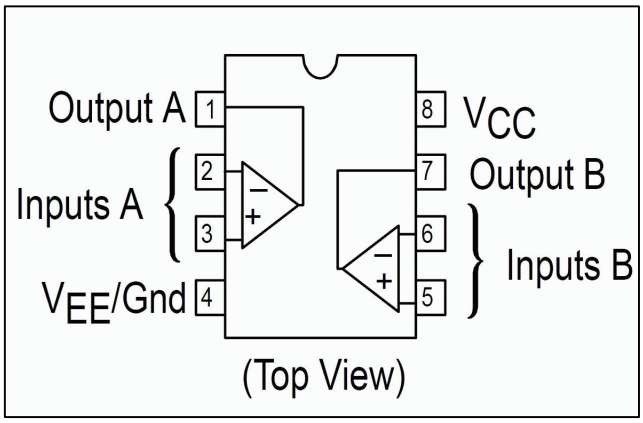

Figure 3.8 : Schéma interne de l'amplificateur LM358N [15].

A l'aide d'une des cartes on va faire l'acquisition du signal de forme sinusoïdal généré par le circuit additionneur (qui représente la référence) par l'entrée analogique A0, ensuite après traitement, se signal sera transféré à la sortie PWM, ensuite récupéré au niveau de la pin 5 afin d'être visualisé.

La dent de scie ou la porteuse, est auto générée par le bloc PWM des cartes Arduino. La fréquence de découpage ne peut prendre que deux valeurs. Dans le cas de la carte UNO, elle dépend des pins utilisés, elle se gère comme suit :

- Pins 5 et 6 : presque 1 KHz.
- Pins 3, 11, 9 et 10 : presque 500 Hz.

On déduit que la fréquence de la MLI, est soit de 1 KHz ou 500 Hz selon les pins.

3.3.2.2. Résultats expérimentaux

Deux essais on était effectués, les deux sont réalisés à une fréquence fixe égale à 50Hz, pour le signal de référence, et de 1KHz pour la dent de scie. L'amplitude du signal de référence en sortie de l'additionneur est quant à lui variable.

3.3.2.2.1. Premier essai

Dans cet essai la référence a une amplitude approximative de 0,8V. Sur la figure3.9, le relevé de la tension injectée sur la Pin A0 est donné, l'amplitude crête à crête du signal qui ne doit pas dépasser 5V, est de 1,61V, avec une fréquence de 50Hz. Cette référence est injectée dans le bloc PWM ce qui nous donne à la borne du pin 5, qui fonctionne à 1KHz, le résultat ainsi obtenu est représenté par la figure (3.11). Ensuite une sauvegarde a été faite grâce à l'oscilloscope numérique, afin de constater l'impact du pas de calcul choisi sous Matlab. Cela nous permet de constater que des commutations ont été ratées, pour le pas de 1e-3 par rapport au pas de 1e-4. Pour une meilleure vérification, le même test à été fait sur Matlab/Simulink, les résultats sont représentés sur la figure (3.12). Les résultats ainsi obtenus nous permettent une validation à vue d'œil pour le pas de calcul de 1e-4. Le test avec un pas de 1e-5 présente plus de commutations.

Le pas de calcul pour le microcontrôleur quand il fonctionne en temps réel sous Matlab/Simulink est un véritable problème, car c'est difficile de trouver des cartes pouvant fonctionner avec des pas inférieurs à 1e-4.

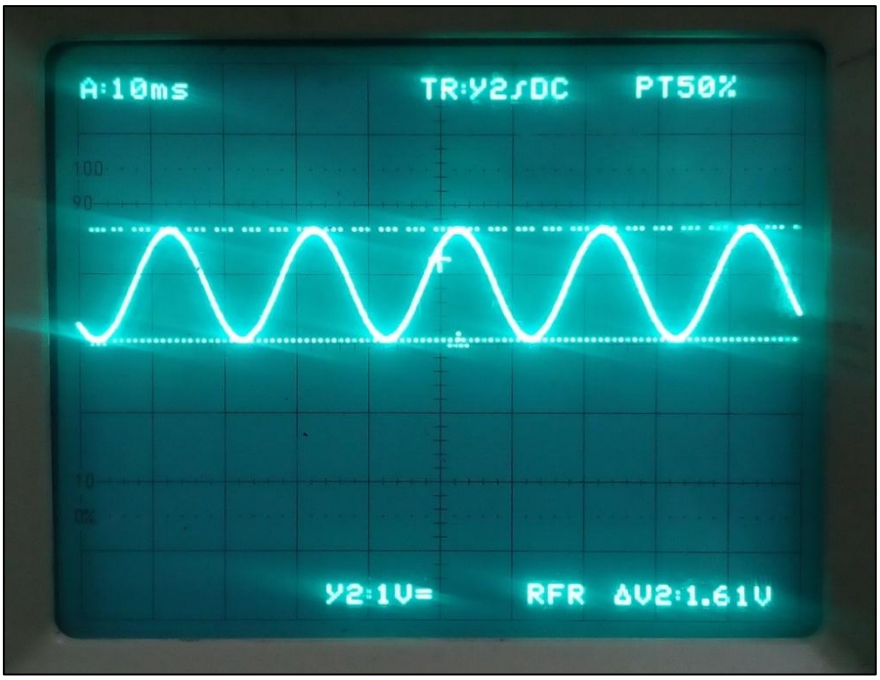

Figure 3.9 : Signal sinusoïdal à la sortie de circuit additionneur.

- **Le signal MLI**

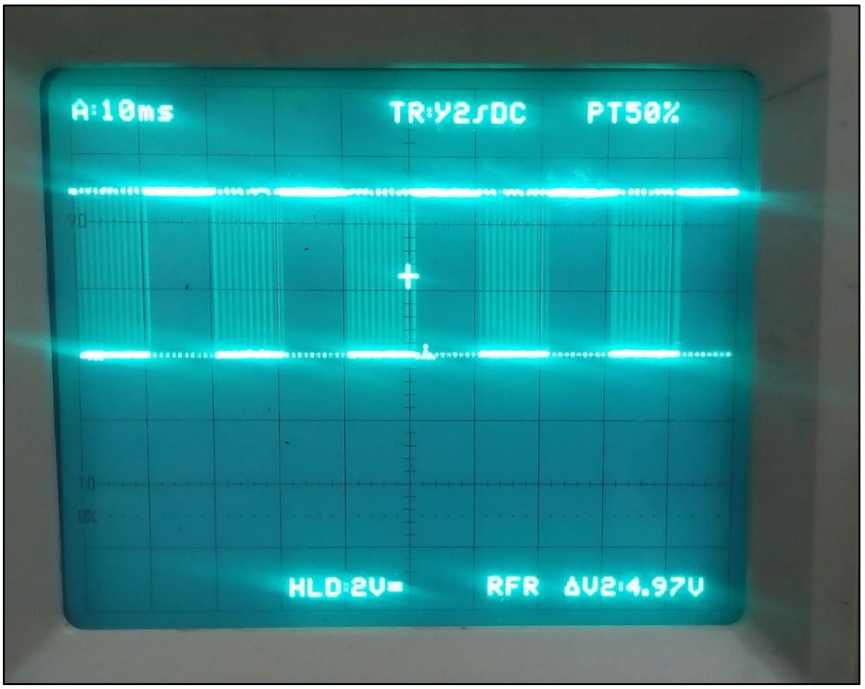

Figure 3.10 : Signal de sortie MLI sur la pin 5.

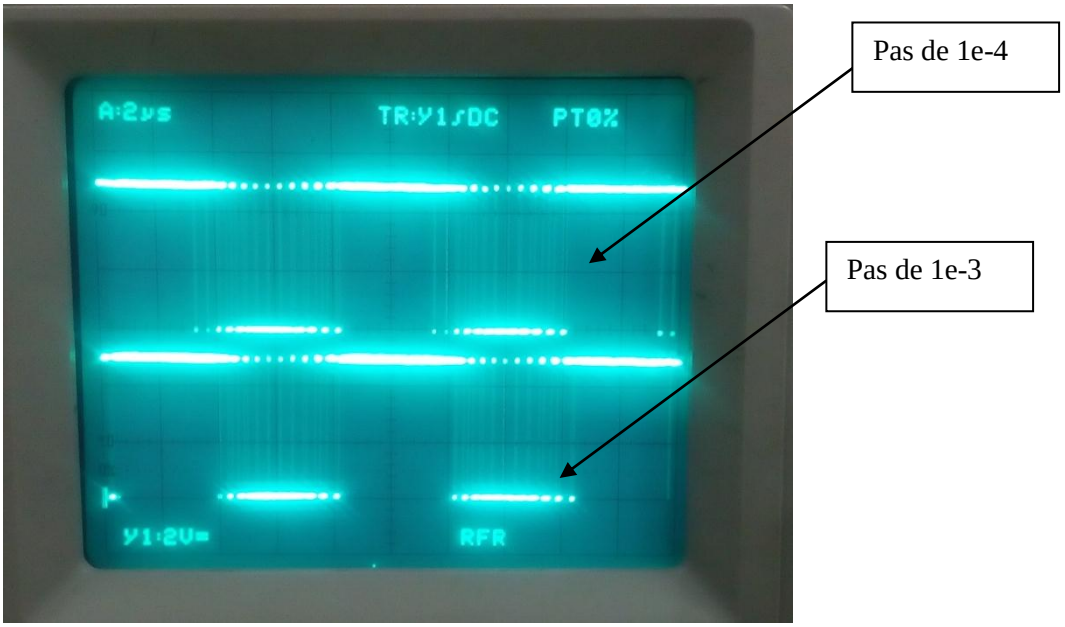

Figure 3.11 : Signal de sortie MLI sur la pin 5 pour des pas de calculs différents.

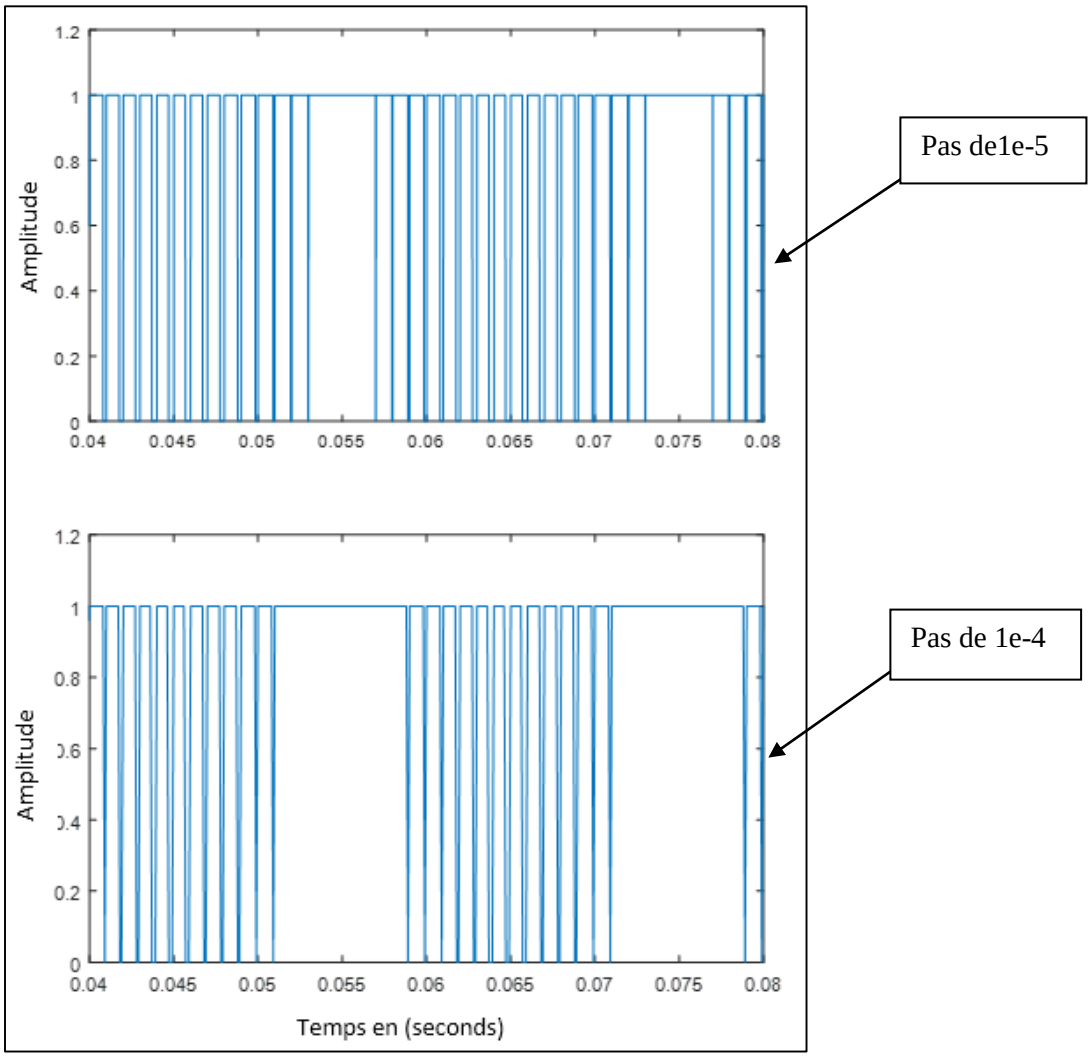

Figure 3.12 : Signal MLI pour des pas de calculs différents avec simulation Matlab.

3.3.2.2.2. Deuxième essai

Cette essai a était effectuer pour une référence d'amplitude U=0,6 V

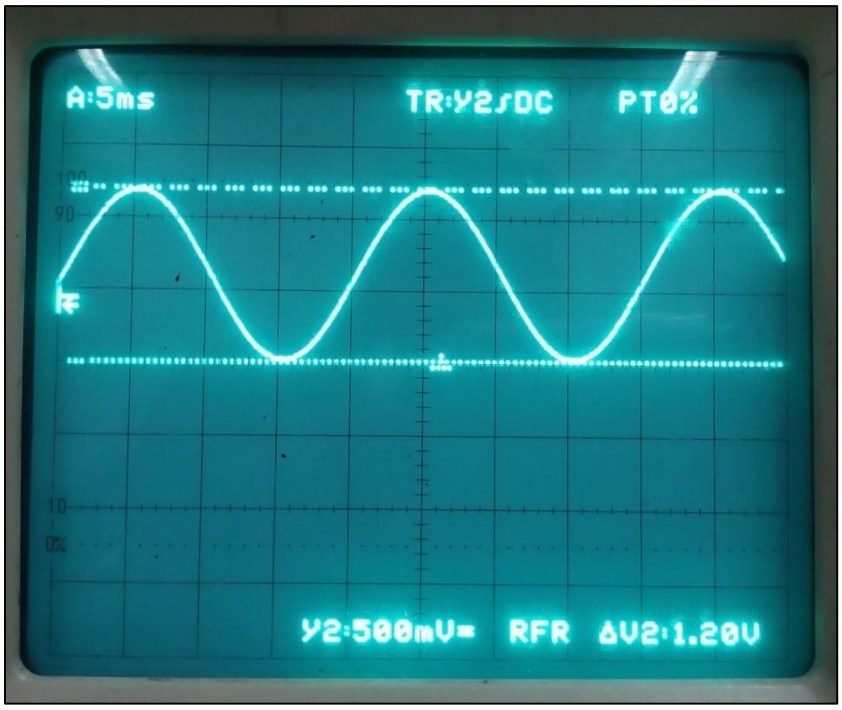

Figure 3.13 : Signal sinusoïdal à la sortie de circuit additionneur.

- **Résultat à la sortie de la carte Arduino**

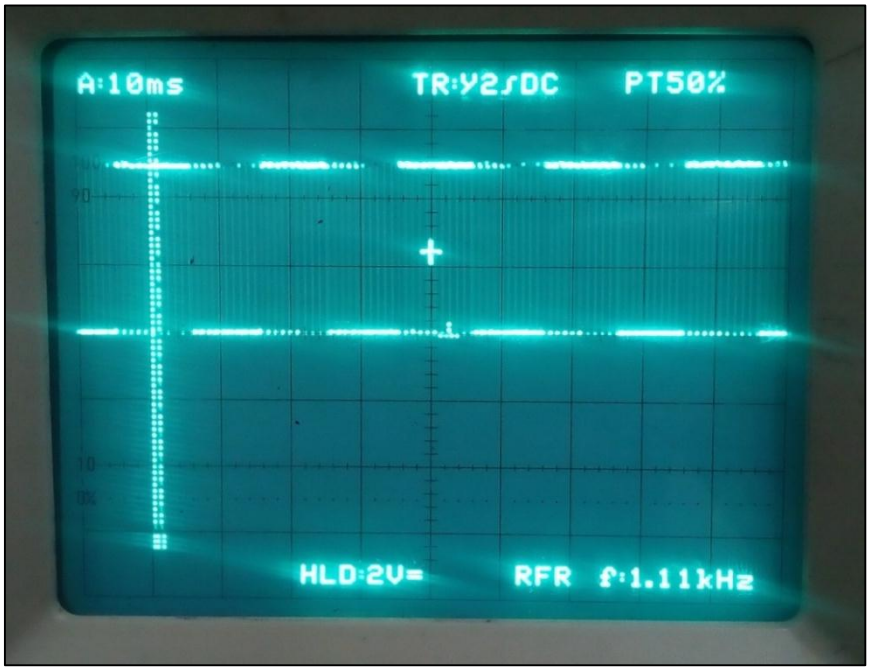

Figure 3.14 : Signal de sortie MLI sur la pin 5.

On observe que le signal à une fréquence au environ de 1 KHz avec une amplitude de 5 V.

3.4. Conclusion

Avec Matlab/Simulink la programmation est très conviviale pour des électrotechniciens, toutefois en analysant les résultats précédemment représentés, des limites existent pour l'utilisation des cartes Arduinos en électronique de puissance, ces limites sont dues principalement aux constantes de temps mises en jeux. Ainsi nous n'avons pas réussi à dépasser la fréquence de découpage de 1 KHz.

De plus, même si nous n'avons pas représenté les résultats de l'acquisition, qui restent de mauvaise qualité surtout avec la carte UNO, à cause du pas de calcul.

Cela nous conduit au chapitre suivant, où nous allons utiliser le logiciel Arduino pour tester les performances de ces cartes et les comparer à celles de leur fiche technique.

Chapitre 4

Acquisition et restitution

Des données

Sur

Arduino

4.1. Introduction

Dans ce chapitre nous nous intéressons au convertisseur analogique numérique des cartes Arduinos. La précision de ces cartes et la qualité d'acquisition et de restitution de la carte Arduino pour des signaux variables dans le temps dépendent fortement de la qualité des convertisseurs ADC de ces dernières. Pour tester les performances de ces cartes en acquisition, nous allons utiliser un signal de forme sinusoïdale de fréquence et d'amplitude variable. Le choix d'une fonction sinusoïdale est dicté par le fait qu'en électrotechnique cette forme est considérée comme une référence.

4.2. Acquisitions des données

Un microcontrôleur est un système qui fonctionne avec des grandeurs numériques. Ainsi pour exploiter des informations disponibles sur des ports d'entrées analogiques, une conversion de ces grandeurs analogiques en grandeurs numériques doit être faite.

Dans un premier temps, l'information est disponible au niveau des capteurs sous forme analogique. Dans le cas des cartes Arduinos, un convertisseur Analogique/Numérique est déjà intégré.

Le niveau maximal des tensions d'entrées des ports analogiques ne doit pas sortir de l'intervalle [0 : 5] V pour la UNO, et entre [0 : 3,3] V pour la DUE, sinon on peut facilement endommager ces entrées analogiques et même la carte [3], [17].

Il faut aussi être conscient que l'ADC n'est pas parfaitement linéaire, l'espace entre deux points n'est pas forcément le même à chaque fois. Le principe d'échantillonnage au niveau de ces convertisseurs doit toujours respecter le théorème de Shannon [17].

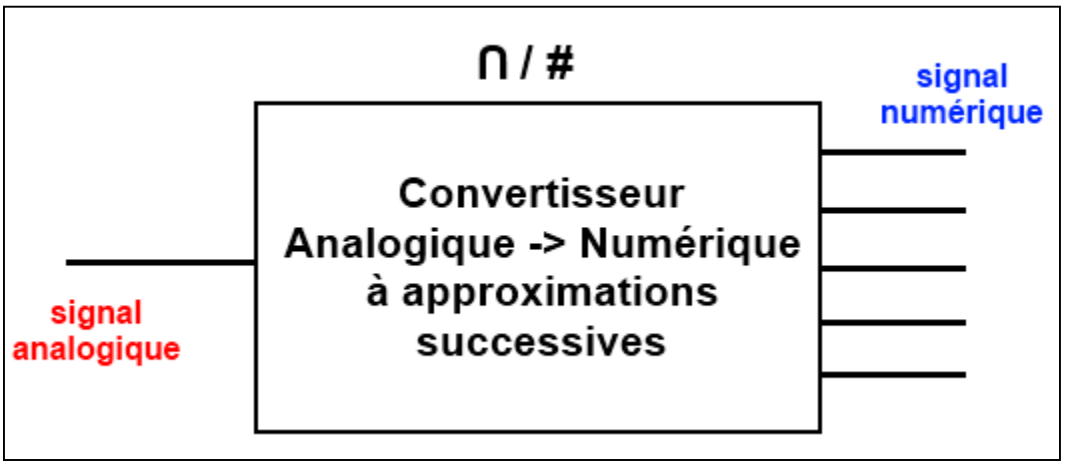

Figure 4.1 : Convertisseur analogique/numérique [11].

4.2.1. Arduino UNO

Sur l'Arduino UNO on trouve 6 entrées analogiques, elles se distinguent par la numérotation suivante A0, …, A5. A l'intérieur de la carte on trouve un seul convertisseur ADC, avec une précision de 10 bits (soit 1024 points), d'où la répartition des 5 volts maximale disponible à l'entrée en valeur discrète à la sortie de l'ADC de 1023 échantillons soit (2^{10}=1024). Pour la carte UNO la fréquence d'échantillonnage maximale est d'environ 10 KHz [17].

4.2.2. Arduino DUE

Sur ce type de carte on trouve 12 entrées analogiques, distinguées par la numérotation suivantes A0, …, A11, et un seul ADC avec une précision de 12 bits (soit 4096 points). Ainsi le nombre de valeur à la sortie de l'ADC attient au maximum 4096 échantillons, soit : (2^{12}=4096) [18].

Un des avantages de cette carte est la valeur de sa fréquence d'échantillonnage qui peut atteindre 1 MHz [18].

4.2.3. La précision de l'Arduino UNO

- Sur l'Arduino UNO la tension d'entrée est comprise dans l'intervalle [0 ;5V] V :

Donc chaque échantillon correspond à 0,0048V soit :

$$\left({}^{5}/_{1024}\right) = 0{,}0048\ V$$

En effet, entre un point est un autre on a $0{,}0048\ V = 4{,}8mV$[17]

4.2.4. La précision de l'Arduino DUE

- Sur l'Arduino DUE la tension d'entrée est comprise dans l'intervalle [0 ; 3.3] V :

$$\left({}^{3.3}/_{4096}\right) = 0{,}00080\ V$$

Ainsi, entre un point est un autre on a $0{,}0008\ V = 0.8mV$[18]

Donc les résultats de la DUE surpasse largement ceux de la UNO, pour des applications de forte précision, il est conseillé d'utiliser la DUE.

4.2.5. Acquisition d'un signal sinusoïdal sur la UNO

Dans cette partie nous avons fait une acquisition d'un signal sinusoïdal de fréquence variable, afin d'estimer la précision de l'acquisition de la carte UNO, un synthétiseur de fréquence est utilisé pour être sûr de l'exactitude du signal à l'entrée des pins analogiques. Différentes fréquences du signal d'entrée sont utilisées allant de 1Hz à 10KHz sur une carte Arduino UNO, afin de définir la limite de la fréquence des signaux d'entrées au-delà de laquelle il faut rester attentif à la forme sinusoïdale, et vérifier l'information de 10KHz de fréquence maximale d'acquisition.

4.2.5.1. Le montage réalisé

La figure 2.2 est une photo du montage réalisé au laboratoire, utilisé pour l'acquisition de nos signaux. L'acquisition des signaux avec Arduino semble simple, grâce à la fonction analogRead, néanmoins pour avoir accès à ces mesures, soit par graphisme, soit par fichier de données, souvent sous Excel, d'autre soubroutine de programmation sont nécessaires. Dans notre cas nous avons opté pour l'utilisation de la carte en oscilloscope, via une application qui exploite le logiciel XOscillo.

Après avoir téléchargé le logiciel XOscillo, et charger un programme Arduino, qui nous a permis une connexion avec le logiciel XOscillo, nous avons réussi à utiliser notre ordinateur comme oscilloscope grâce au logiciel XOscillo, et ainsi visualiser en temps réel nos grandeurs.

Le matériel utilisé :

- Un oscilloscope ;
- Un synthétiseur de fréquence ;
- Une source de tension continue ;
- Une carte Arduino UNO ;
- Un additionneur sur lab de test.

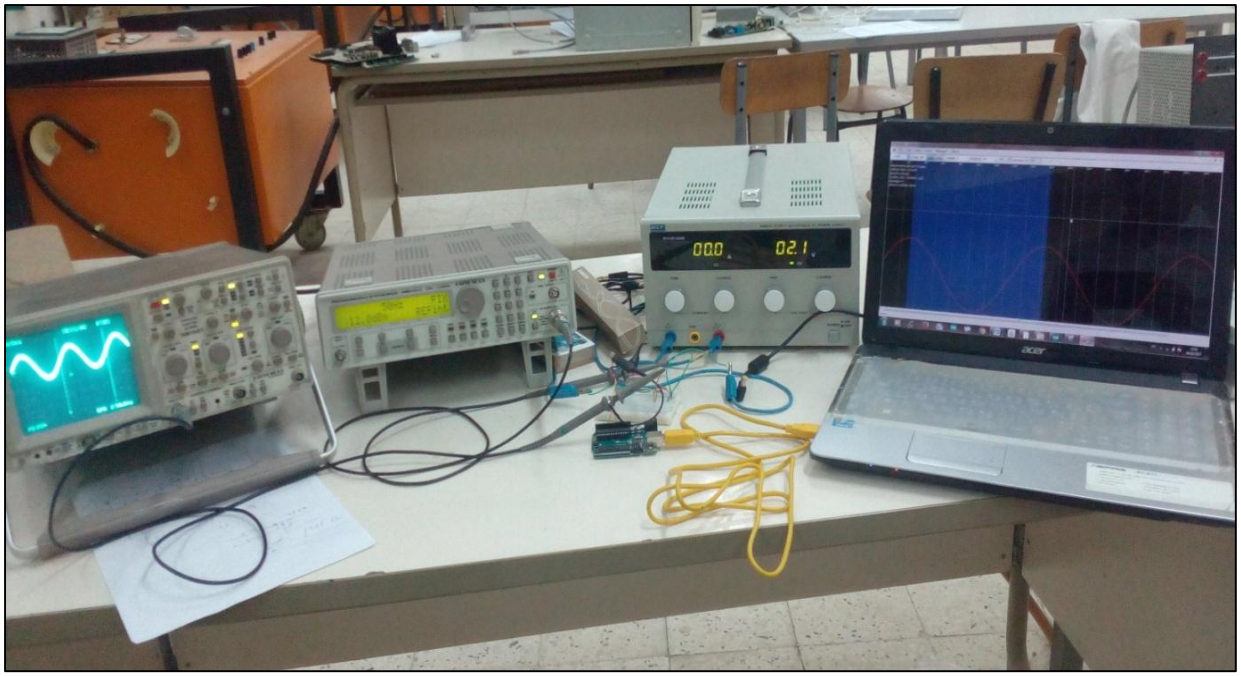

Figure 4.2: Montage réalisé pour l'acquisition d'un signal « sinusoïdal ».

4.2.5.2. Acquisition pour une fréquence de 50 Hz

La figure suivante montre le signal sinusoïdal à 50 Hz après acquisition « donnée téléchargée sur XOscillo », donc après le passage par l'ADC de la carte Arduino il est devenu à 65 Hz.

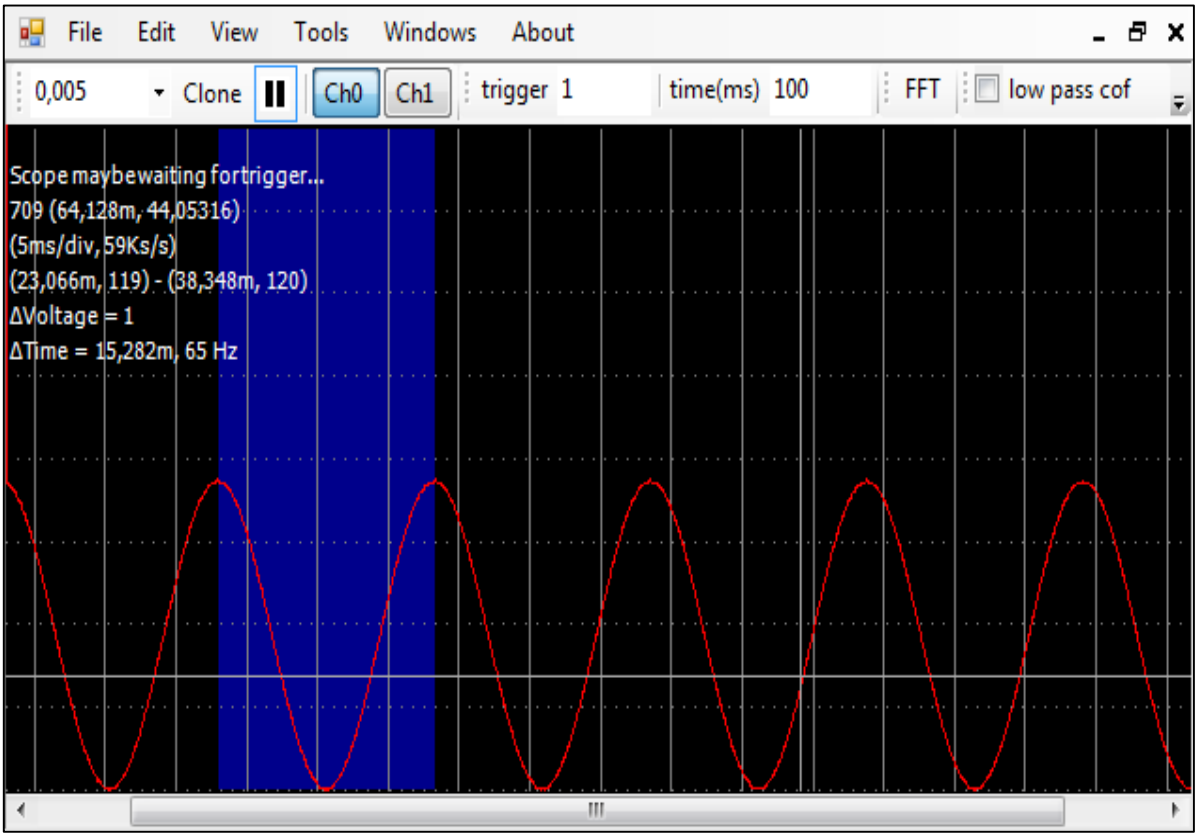

Figure 4.3: Acquisition en entrée d'un signal sinusoïdal à 50 Hz.

Le logiciel XOscillo nous permet de faire la FFT du signal après acquisition, ce qui nous offre l'opportunité d'une bonne comparaison avec le signal source fourni par le synthétiseur et visualiser sur oscilloscope après son décalage, c-à-dire à l'entrée de la carte UNO. Cela nous permet une appréciation de la qualité de l'échantillonnage, nous constatons un décalage de la fréquence, elle était à 50Hz avant l'échantillonnage, la restitution est à 65Hz. Donc avec un rapport d'environ égal à 1,3.

Concernant l'amplitude après échantillonnage, l'approche reste bonne.

Afin de vérifier le facteur de proportionnalité entre les fréquences plusieurs autres tests sont effectués, dans ce qui suit nous allons présenter certains d'entre eux.

4.2.5.3. Acquisition pour une fréquence de 1 kHz

La figure suivante illustre le signal après acquisition pour une fréquence de 1 KHz, qui passe à 1300 Hz à la sortie de l'ADC.

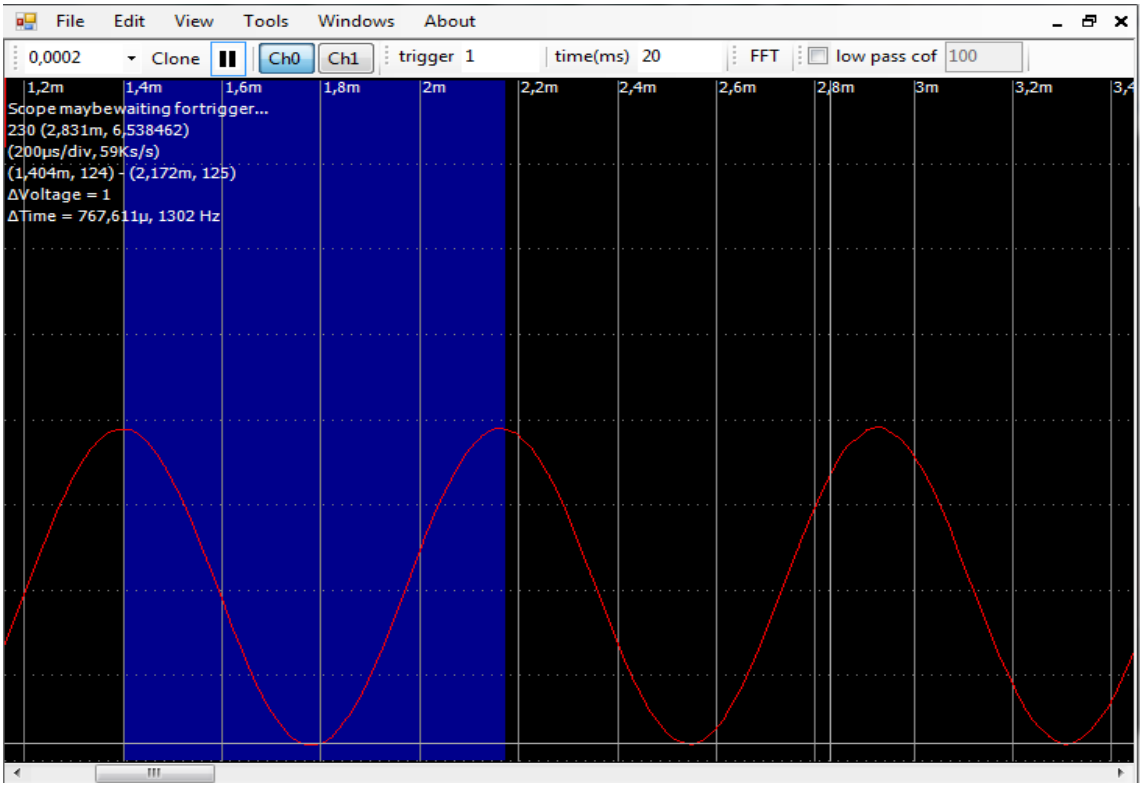

Figure 4.4: Acquisition en entrée d'un signal sinusoïdal à 1 KHz.

Nous observons le même phénomène concernant le rapport de proportionnalité en fréquence, avec un signal qui suit parfaitement une forme sinusoïdale.

4.2.5.4. Acquisition pour une fréquence de 10 kHz

La figure suivante illustre un signal sinusoïdal à 10 kHz après acquisition, qui passe a 13032 Hz a sa restitution à la sortie de l'ADC de la carte.

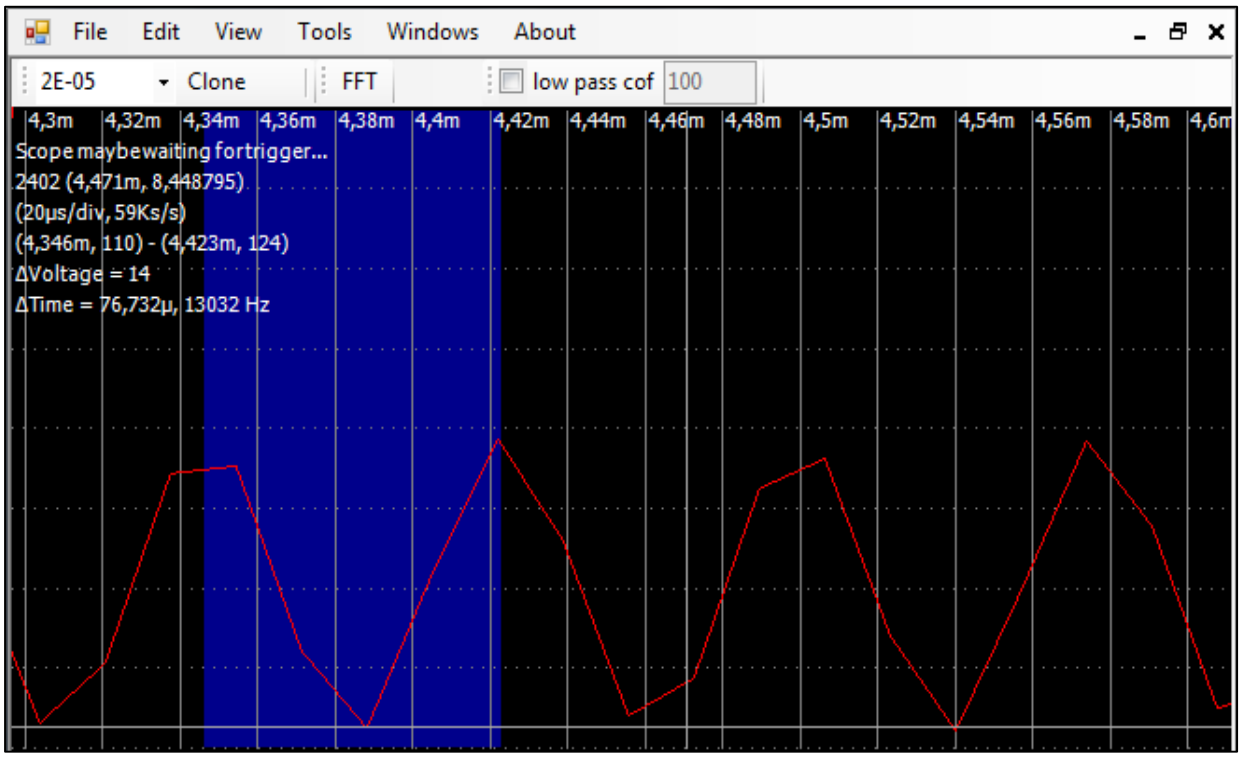

Figure 4.5 : Acquisition en entrée d'un signal sinusoïdal a 10KHz.

En regardant de plus près, nous constatons que la période d'échantillonnage est voisine de 0,02 ms c'est la projection des segments sur l'axe des abscisses. Ce qui nous permet de conclure concernant la fréquence de l'ADC.

A cette fréquence le signal après l'ADC n'est plus fiable. Nous constatons une forme pseudo sinusoïdale avec une fréquence toujours dans le même rapport de proportionnalité.

Remarque :

Le procédé d'acquisition de l'Arduino DUE est dans le principe, le même que la UNO, néanmoins le téléchargement ne se fait pas de la même façon. Les programmes pour l'envoi vers un fichier Excel sont plus complexes. Et l'exploitation du logiciel XOscillo n'est pas compatible avec la DUE, c'est le logiciel Matplotlib qu'il faut utiliser, et une programmation sous Python pour le lien et le téléchargement. La procédure est plus complexe, elle demande plus de temps pour la maitriser.

4.3. Restitution des signaux

Les cartes Arduinos dans notre projet, sont utilisées pour gérer la commande éloignée des convertisseurs statiques, en d'autres termes la partie traitement de l'information.

Pour des raisons de dimensionnement, de coût, de qualité d'énergie et de normes concernant les perturbations électromagnétiques, les convertisseurs d'électronique de puissance sont commandés avec des techniques dites par Modulation de Largeurs d'Impulsions.

Ces types de convertisseurs sont appelés système à deux bases de temps, en général la haute et la basse fréquence se côtoient.

A titre d'exemple, une machine asynchrone classique, doit être soumise à une fréquence de découpage « fréquence PWM » qui tourne autour de 4kHz. Il se trouve que la carte Arduino UNO comme elle utilise des broches PWM, les sorties sont à des fréquences standards bien déterminées :

- Broche 5 et 6 délivre une fréquence de sortie d'environ 960 Hz ;
- Broche 9 et 10 délivre une fréquence de 492 Hz ;
- Broche 3 et 11 délivre la même fréquence 492 Hz.

Utiliser ces fréquences à cet état n'est pas acceptable pour certaines applications, par exemple :

- Onduleurs pour les filtres actifs ;
- Les algorithmes MPPT pour les systèmes photovoltaïques qui exige des fréquences souvent supérieures à 50KHz ;
- Onduleur alimentant des machines Asynchrones.

4.3.1. Booster la carte Arduino UNO en fréquence

Une possibilité d'augmenter en fréquence existe, il faut insérer certains codes dans le programme arduino dans la partie (void setup () {etc...}). Cela nous permet d'atteindre des fréquences assez élevée pour diverses applications. Nous pouvant aller jusqu'à 62 kHz.

Les codes à rajouter sont comme suit :

4.3.1.1. Les Pins 5 et 6

Setting	Divisor	Frequency
0x01	1	62500
0x02	8	7812.5
0x03	64	976.5625 <--DEFAULT
0x04	256	244.140625
0x05	1024	61.03515625

TCCR0B = (TCCR0B & 0b11111000) | <setting> ; c'est le code à insérer dans le programme en changeant le setting selon la fréquence qu'on veut obtenir [4].

4.3.1.2. Les Pins 9 et 10

Setting	Divisor	Frequency
0x01	1	31372.55
0x02	8	3921.16
0x03	64	490.20 <--DEFAULT
0x04	256	122.55
0x05	1024	30.64

TCCR1B = (TCCR1B & 0b11111000) | <setting> ; c'est le code à insérer dans le programme en changeant le setting selon la fréquence qu'on veut obtenir [4].

4.3.1.3. Les Pins 3 et 11

Setting	Divisor	Frequency
0x01	1	31372.55
0x02	8	3921.16
0x03	64	490.20 <--DEFAULT
0x04	256	122.55
0x05	1024	30.64

TCCR1B = (TCCR1B & 0b11111000) | <setting> ; c'est le code à insérer dans le programme en changeant le setting selon la fréquence qu'on veut obtenir [4].

4.3.2. La restitution d'un signal à 62 kHz

Les figures suivantes illustrent un programme avec le code pour augmenter en fréquence, et un signal de sortie PWM boosté à 62 kHz.

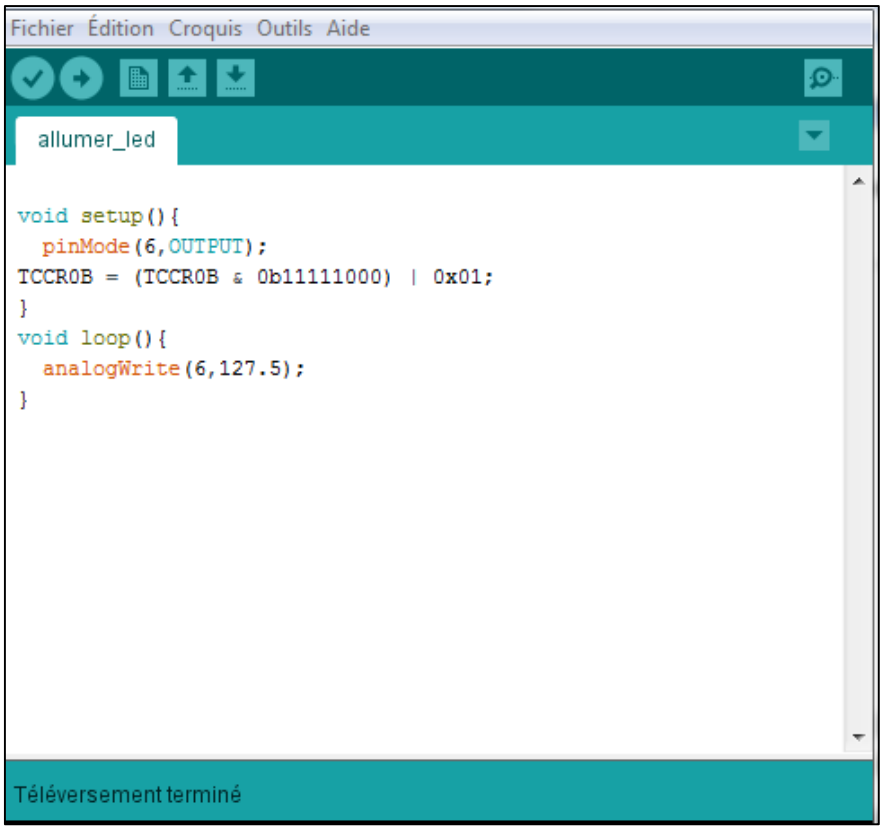

Figure 4.6 : Le programme avec le code pour booster la frequence.

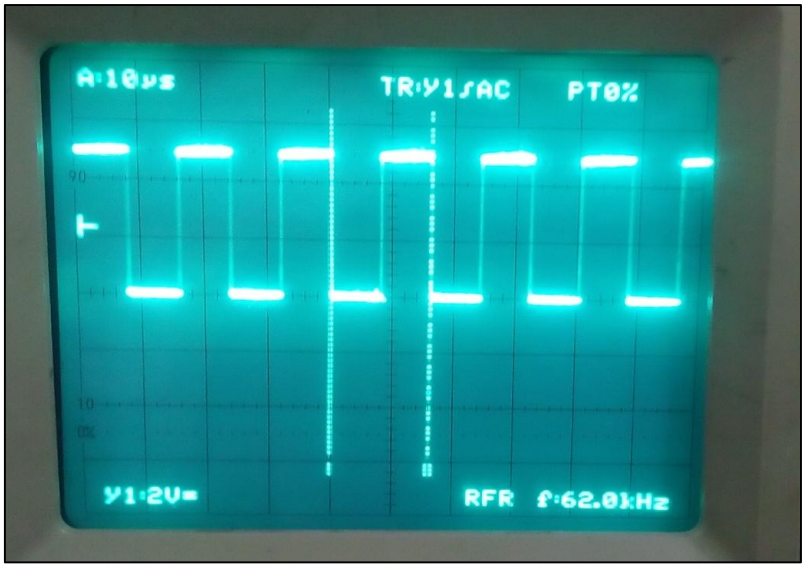

Figure 4.7 : Signal de restitution en sortie.

Nous constatons que la carte Arduino UNO en terme de restitution peut aller jusqu'à une fréquence maximale de 62 kHz.

4.3.3. Gestion d'un signal MLI avec la DUE

Dans cette partie, nous avons implanté un programme de modulation de largeurs d'impulsions avec la carte Arduino DUE, la référence est à 50 Hz, elle est de forme sinusoïdale et générée par programme, la modulante est à 200 kHz. Le programme sous Arduino est donné en Annexe B [12].

Dans la figure suivante on voit bien notre signal MLI à 200 kHz avec la carte Arduino DUE.

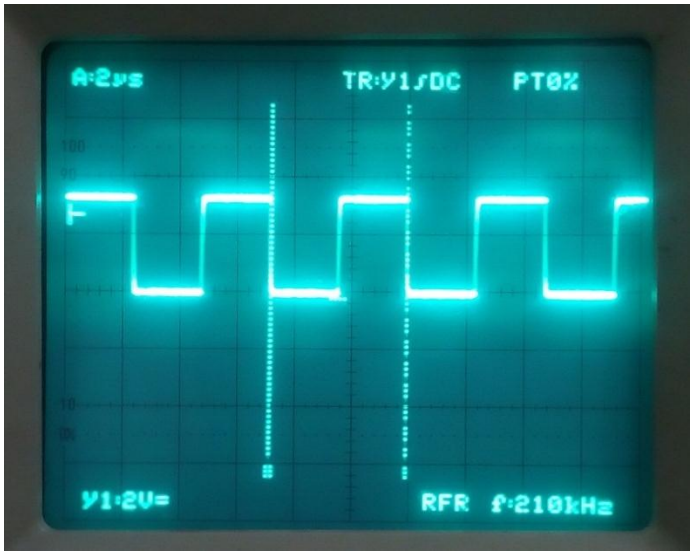

Figure 4.8 : Zoom du signal de la MLI en sortie de la DUE.

4.4. Conclusion

Dans cette partie de notre travail nous avons testé les deux cartes Arduinos concernant leur fiabilité.

En restitution, nous nous sommes plus focalisés sur le problème de fréquence, car en amplitude, on utilise des grandeurs digitales. Des chutes de tension peuvent survenir en charge, mais cela ne dégrade en rien l'information que l'on désir transmettre. Concernant la base de temps, dans un premier temps on a cru être limités avec la carte UNO. Toute fois après des recherches nous sommes arrivés à l'information que la carte UNO pouvait être boostée en temps en utilisant des codes. Cela nous a permis de faire passer la fréquence de découpage de 1 kHz à 62 kHz, un gain très appréciable qui élargi l'utilisation de la carte en électronique de puissance.

Pour la carte DUE une performance en fréquence de 200KHz est atteinte sans aucun code.

En acquisition, nous avons constaté que la carte Arduino UNO est fiable concernant les amplitudes, mais d'un point de vue base de temps, un problème s'est présenter. Il s'agit d'une compression du temps. Cette erreur est due au convertisseur analogique numérique incorporé avec les cartes Arduinos UNO. Pour mettre en évidence cette compression, plusieurs tests ont été effectués. Dans notre étude, plutôt que d'utiliser la variable temps, nous avons raisonné en fréquence. Nous avons trouvé cela plus judicieux, à cause de la conclusion à laquelle nous sommes arrivés, à savoir, si on injecte un signal d'une fréquence « f » nous obtenons en sortie le même signal de fréquence différentes, mais toujours proportionnel avec un rapport de 1,3.

Plusieurs cartes non clonées ont été utilisées, ces cartes nous ont permis de généraliser notre conclusion concernant le rapport de proportionnalité.

Pour la carte DUE, le travail n'a pas été effectué par manque de temps.

Chapitre 5

Commandes

Des

Moteurs

A

Courant Continu

5.1. Introduction

Le but principal de ce chapitre est de faire une application. Elle consiste à utiliser le signal MLI précédemment généré par les cartes Arduinos pour la commande d'un hacheur quatre quadrants, ce dernier alimente une machine à courant continu a vitesse variable avec deux sens de rotation.

Le pont H dont on dispose est le L293D, il peut être commandé de différentes façons, à titre d'exemple soit en convertisseur réversible en tension uniquement, soit en hacheur quatre quadrants.

Nous avons opté pour un fonctionnement en hacheur réversible en tension, deux essais pour deux fréquences différentes on était réalisés.

Le but de cette partie est de vérifier notre commande, c'est-à-dire le fonctionnement de la carte Arduino en charge. Cela peut être fait quelle que soit la puissance de la charge, d'où l'utilisation d'une Micro machine à courant continu (M.C.C) de faible puissance.

5.2. Commande de la machine DC

5.2.1. Description de la machine à courant continu

Toutes les machines électriques sont des convertisseurs électromécaniques, donc elles permettent une conversion bidirectionnelle de l'énergie. Dans une installation électrique selon le besoin, l'énergie peut changer de sens de transfert, d'où deux types de fonctionnements :

- Fonctionnement moteur : énergie électrique convertie en énergie mécanique.
- Fonctionnement générateur : énergie mécanique convertie en énergie électrique.

Ainsi la machine à courant continu permet d'avoir un fonctionnement dans les quatre quadrants du plan couple-vitesse (quand la réversibilité en courant et en tension est possible[8].

Figure 5.1 : Micro machine à courant continu.

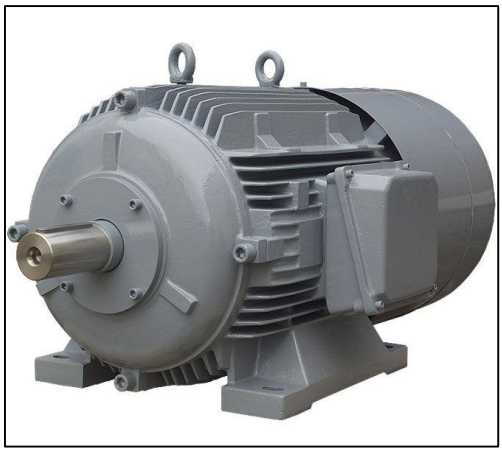

Figure 5.2 : Moteur à courant continu de faible puissance.

5.2.1.1. Principe de fonctionnement

Son principe de fonctionnement est très simple, en fonctionnement moteur l'inducteur doit être excité avec un courant I_s, d'où la création d'un champ magnétique (induction magnétique) dans l'entrefer (stator-rotor). En alimentant le circuit du rotor avec un courant I_r on aura une interaction entre l'induit et l'inducteur cela crée une force de Laplace, entrainant ainsi l'arbre du moteur [8].

La M.C.C est composée principalement des éléments suivants :

- Un stator (inducteur).
- Un rotor (induit).
- Armature.
- Les balais.
- Le collecteur.

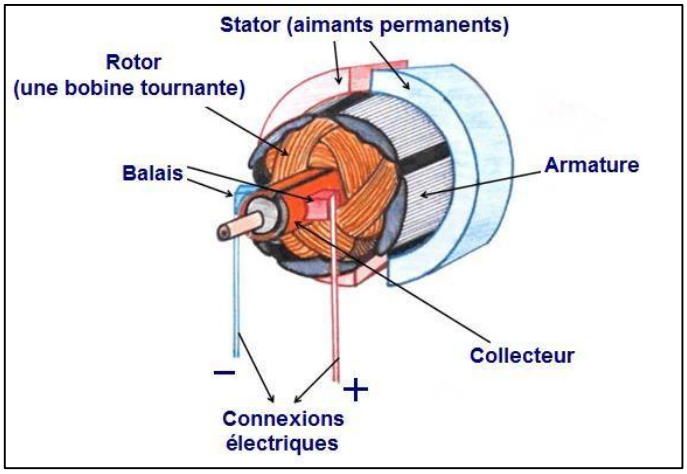

Figure 5.3 : Différents éléments de la machine à courant continu.

5.2.1.2. Les différents types de M.C.C

Les moteurs à courant continu fonctionnent tous avec le même principe, la différence réside dans la manière dont on excite le stator, à partir de là on distingue 4 types de moteurs qui sont les suivants [8], [13] :

- Moteur à excitation indépendante ;
- Moteur série ;
- Moteur à excitation shunt ;
- Moteur à excitation composé.

Le choix d'une excitation donnée dépend de l'application.

5.2.2. Le pont H L293D

Pour les applications de faible puissance nous disposons de la structure intégrée de pont en H. A titre d'exemple le L293D, qui est illustré sur la figure 5.4, et que nous allons utiliser dans la suite de notre application.

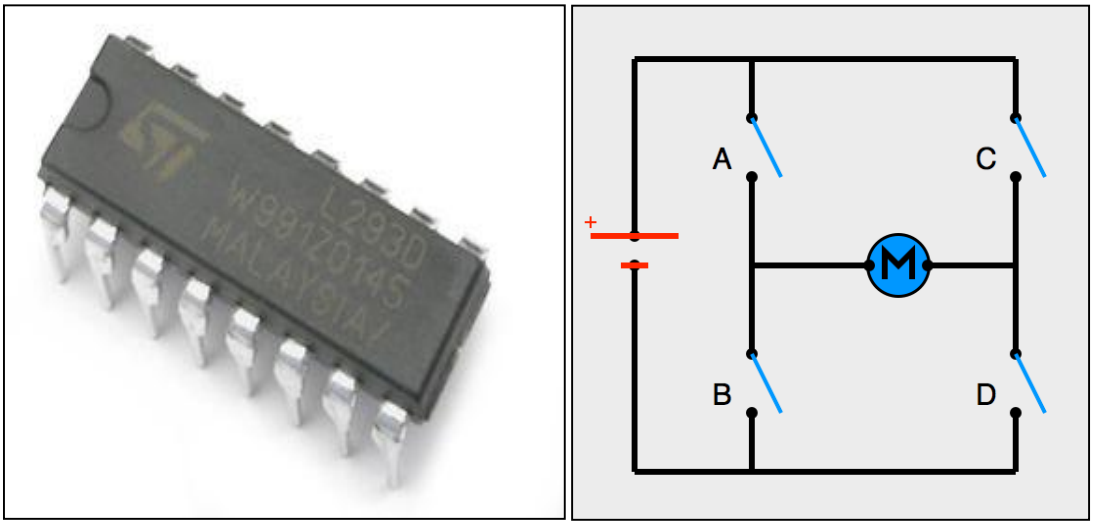

Figure 5.4 : Circuit d'un pont H

Le pont en H est une structure électronique qui sert à contrôler la polarité et la tension fournie aux bornes d'un dipôle. Il est composé de quatre éléments de commutation qui peuvent être des transistors bipolaires ou à effet de champ, avec des diodes en antiparallèle. Ce circuit est très employé dans les applications de l'électronique de puissance pour le contrôle des moteurs de faible puissance, en fonction de l'application, il peut fonctionner soit en hacheurs ou bien en onduleurs d'une façon indifférente cela dépend de la nature de la charge qu'il alimente [8].

5.2.2.1. Structure du L293D

De plus avec le L293D on peut commander deux moteurs à la fois, car sa structure interne en réalité nous offre deux ponts en H réversible en courant.

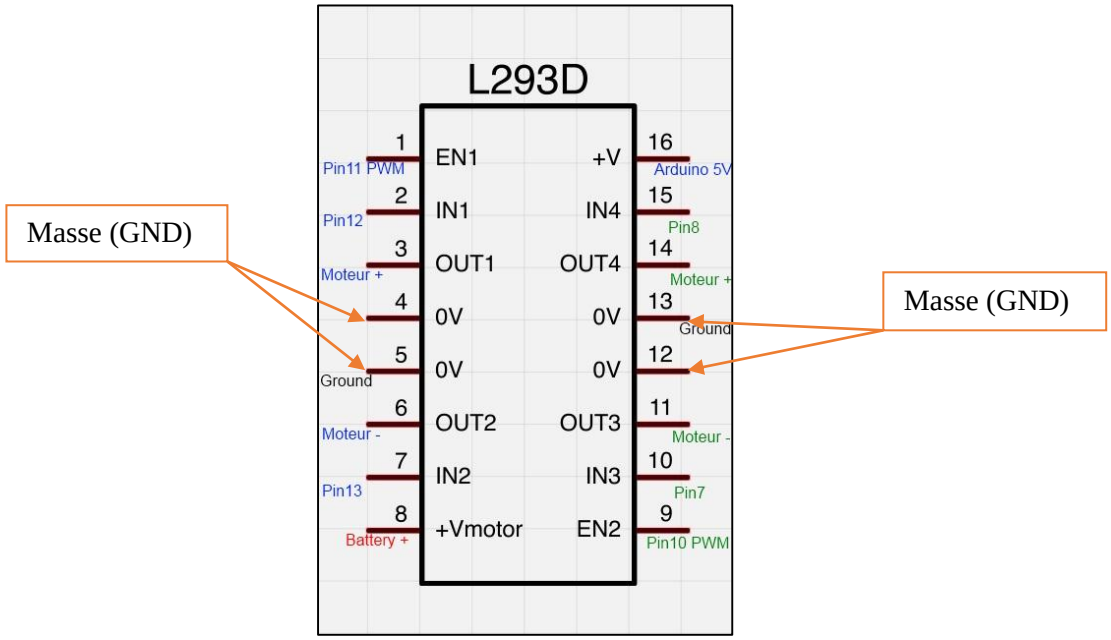

Figure 5.5 : Structure de L293D

De plus, d'après cette figure on voit bien qu'il dispose de plusieurs branches GND, La raison de cette multitude de GND est le refroidissement du composant. [2]

5.2.2.2. Utilisation du pont H avec M.C.C

Le montage que nous allons réaliser avec la Carte Arduino et le pont en H qui assure deux fonctions :

- ✓ Inverser le sens de rotation ;
- ✓ Variation de la vitesse.

En fonction de la commande utilisée, le pont en H qui est le L293D, permet d'effectuer un freinage régénératif, ce qui veut dire que dans la phase de freinage en récupère de l'énergie électrique (principe de véhicule électrique).

Dans notre cas on utilise la partie réversible en tension uniquement, ce qui nous permet d'obtenir un fonctionnement dans les deux quadrants I et III :

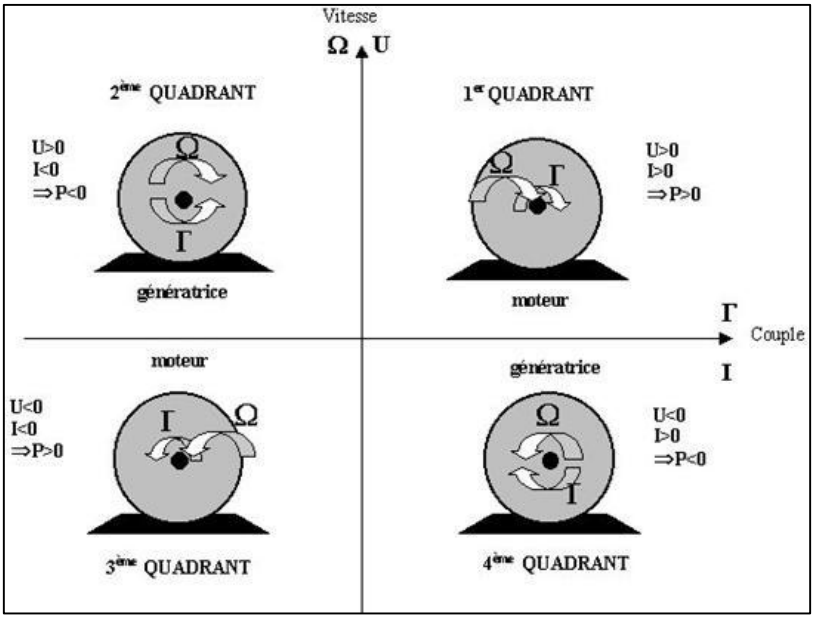

Figure 5.6 : Les quadrants de fonctionnement de la M.C.C.

5.2.3. Le hacheur quatre quadrants

Un hacheur quatre quadrants est un convertisseur qui assure une réversibilité en courant et en tension, en fonction de la manière dont on commande les quatre transistors, il assure différents types de fonctionnement.

Dans notre cas, pour assurer deux sens de rotation différents avec une variation de vitesse, le pont sera commandé comme indiqué par la figure 5.8. Il est possible de faire varier la vitesse de rotation du moteur en limitant plus ou moins la puissance fournie au moteur et son contrôle se fait en agissant sur le rapport cyclique [8].

Le test est fait pour deux valeurs de la fréquence de commutation des transistors.

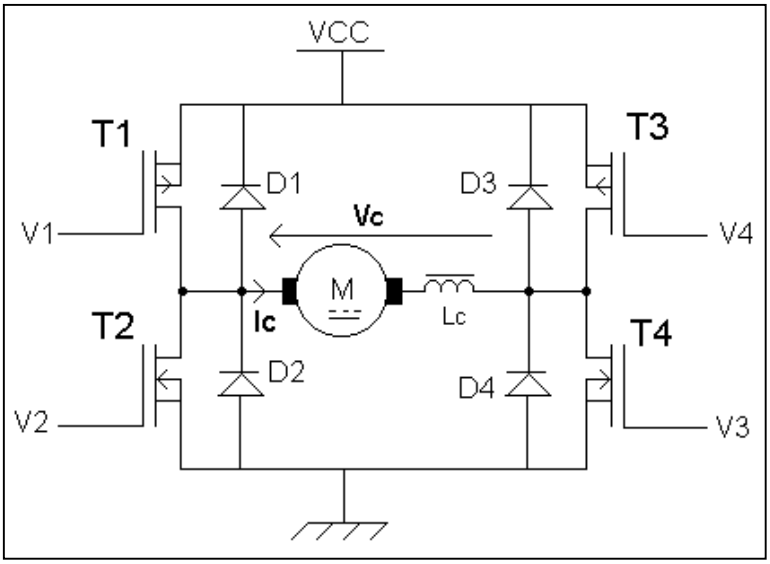

Figure 5.7 : Structure d'un hacheur quatre quadrants [19].

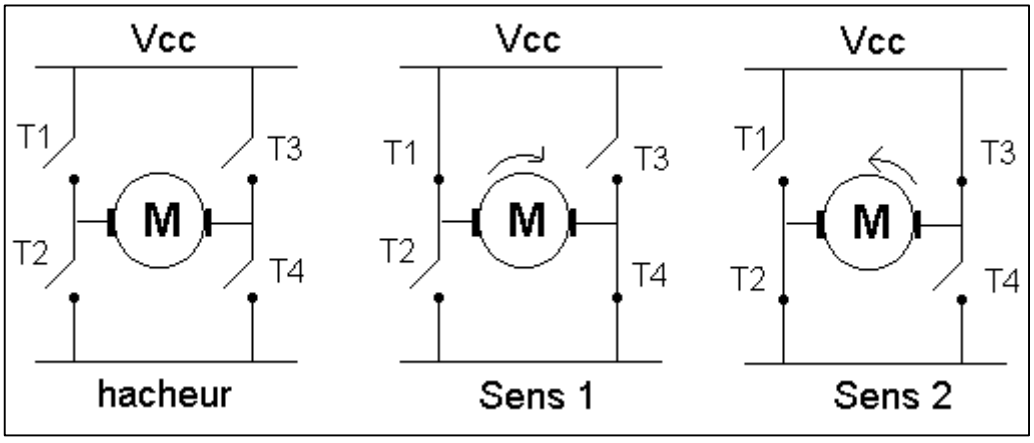

Figure 5.8 : Etats des interrupteurs pour les deux sens de rotation [19].

La structure du hacheur quatre quadrants est constituée de :

- Quatre transistors T1, T2, T3, T4.
- Quatre diodes de roues libres D1, D2, D3, D4 montées en antiparallèles avec les quatre transistors.

5.2.3.1. Principe de fonctionnement

Pour avoir le sens de rotation « 1 » les transistors T1et T4 sont commandés d'une façon complémentaire et haché, ces interrupteurs reçoivent des signaux de commandes identiques. De l'autre côté, les transistors T2 et T3 seront continuellement bloqués [20].

Donc le pont va fonctionner comme suit :

- ✓ Dans l'intervalle [0, αT], les transistors T1et T4 sont commandés.
- ✓ Dans l'intervalle [αT, T], les transistors T1 et T2, T3 et T4 sont bloqués donc les diodes D2 et D3 prennent le relai.
- ✓ Si le courant de charge s'annule à un instant t1 inferieur à T, alors on observe une tension différente de celle du bus continu du pont, c'est la f.c.e.m du moteur.

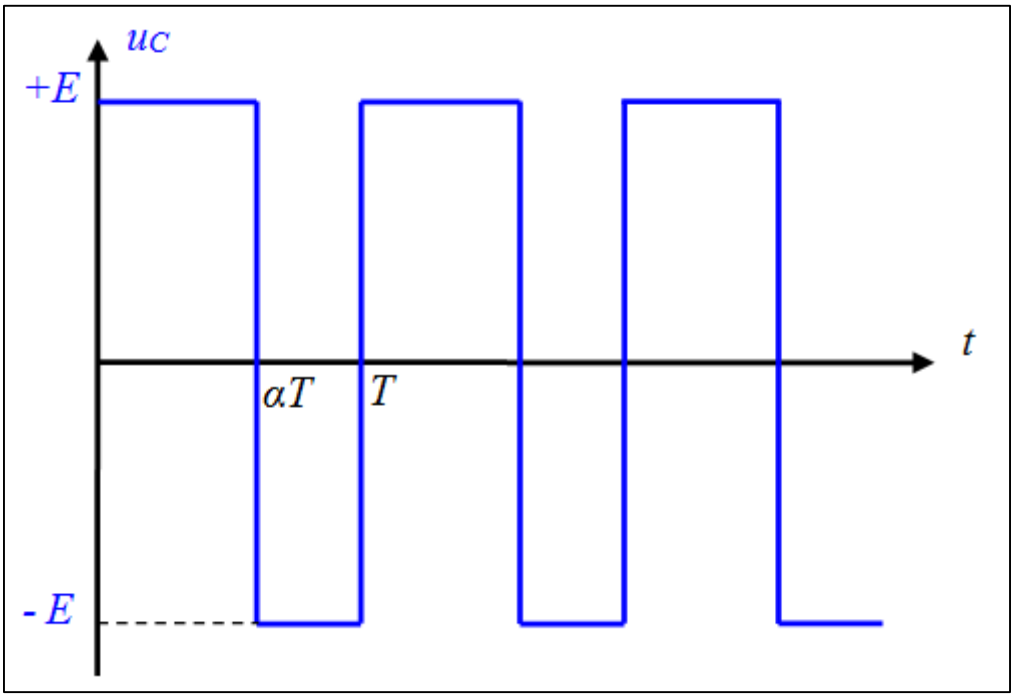

Figure 5.9 : Tension de charge au bornes du moteur si le courant de charge est non nul.

D'où l'expression de la tension de sortie moyenne :

$$Uc_{moy} = \frac{1}{T}\left[\int_0^{\alpha T} E\,dt + \int_{\alpha T}^{T} (-E)\,dt\right]$$

Après intégration, on aura :

$$Uc_{moy} = E(2\alpha - 1)$$

5.3. Partie expérimentation

Dans la suite de cette partie on va procéder à l'implantation des composants nécessaires pour commander la machine à courant continu à l'aide d'une carte Arduino.

5.3.1. Le matériel à utiliser

- Carte Arduino UNO.
- Moteur à courant continu.
- Un pont en H (utiliser en hacheur réversible en tension), d'où on aura que deux quadrants de fonctionnement.
- Un potentiomètre (pour varier la vitesse).
- Deux boutons poussoirs (S1 et S2).
- Pile 9V.
- Des résistances.

La figure suivante est une photo du montage réalisé :

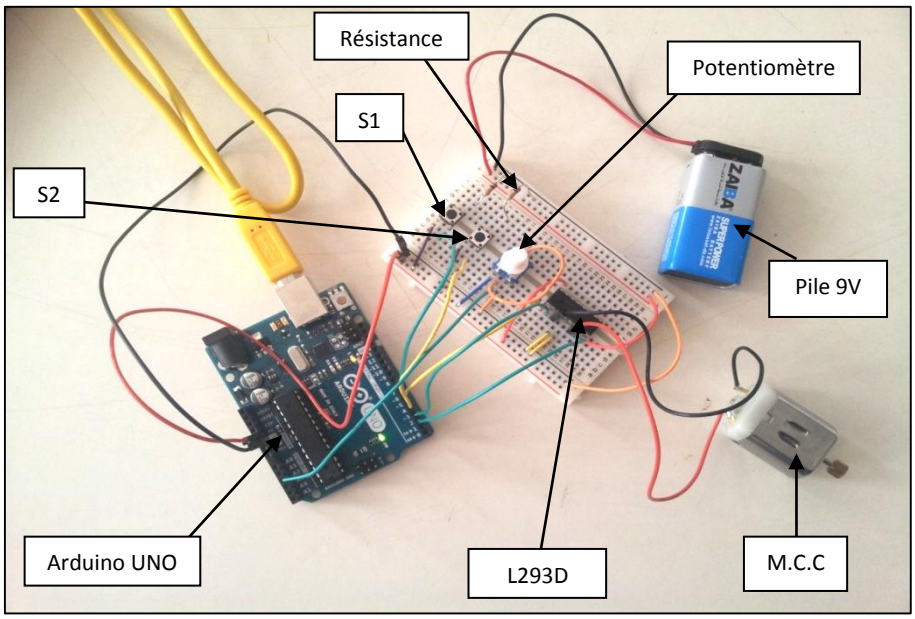

Figure 5.10 : Montage réalisé [1].

5.3.2. Explication du dispositif expérimental

Après avoir écrit le programme sous Arduino, l'avoir vérifié, ensuite téléchargé vers la carte on aura [1] :

- En appuyant sur S1 le moteur est alimenté dans un sens, et en appuyant sur S2 on a l'inversion du sens de rotation et ça par l'intermédiaire du pont en H ;
- Le potentiomètre branché sur la broche PWM nous permet de faire varier la tension de référence afin de contrôlé la vitesse de rotation.

5.3.3. Acquisitions des données

Après plusieurs essais nous avons vu que le moteur fonctionne parfaitement sans problème que ce soit au niveau de ce dernier ou bien au niveau de la carte.

De plus, on à réaliser deux essais :

- Le premier à basse fréquence ;
- Le deuxième à haute fréquence.

5.3.3.1. Les courbes obtenues

5.3.3.1.1. Essai à basse fréquence

Ce test a été réalisé à une fréquence de presque 1 kHz.

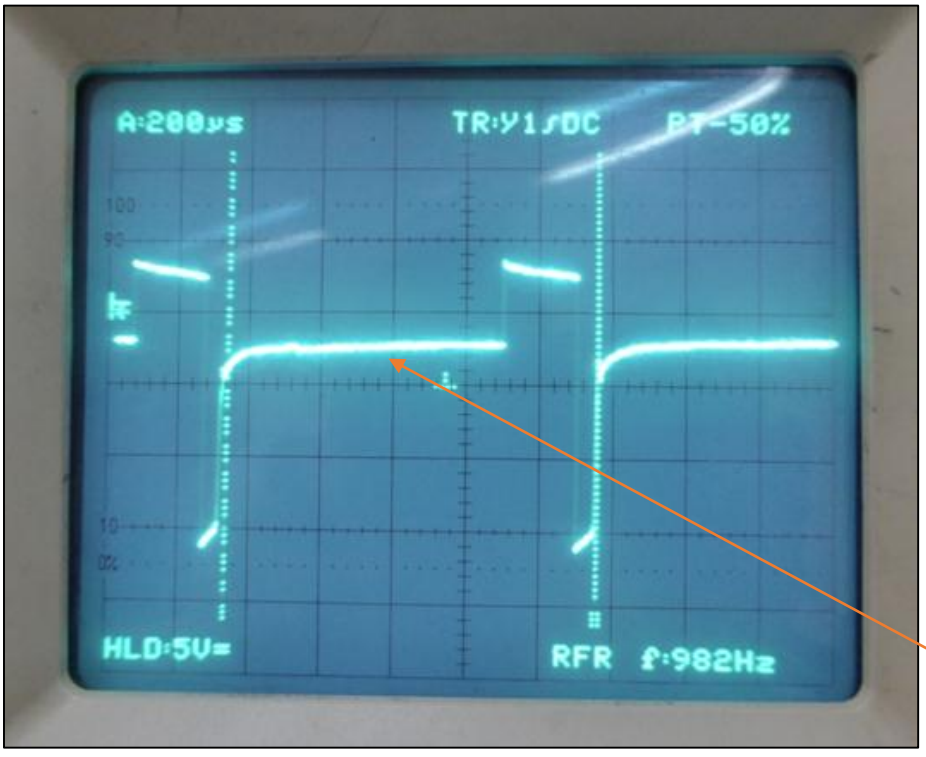

Figure 5.11 : Tension aux bornes du M.C.C pour f=982 Hz.

5.3.3.1.1.1. Explication du graphe

A partir de cette figure on voit bien qu'on est en conduction discontinue (le courant s'annule dans les composants sur une fraction de la période). Cela est due à la constante de temps de notre circuit τ , qui est inférieure à la période de notre signal T, d'où l'apparition de la f.c.é.m du moteur sur une fraction de la période.

5.3.3.1.2. Essai à haute fréquence

Ce test a été réalisé à une fréquence de 62 kHz.

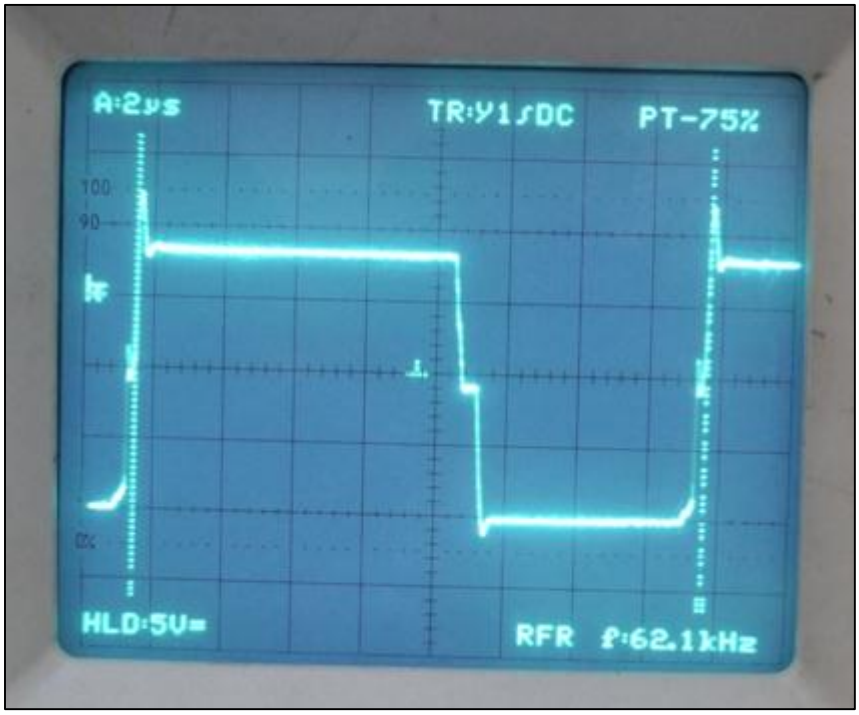

Figure 5.12 : Tension aux bornes du M.C.C pour f=62 kHz.

5.3.3.1.2.1. Explication du graphe

Durant cet essai, on constate qu'on est en conduction continue, car Il n'y a pas d'apparition de la f.c.é.m au niveau de la tension aux bornes du moteur. Les phénomènes constatés s'expliquent comme suit :

Si en augmente la fréquence f, cela entraine une diminution de la période T, ($f = \frac{1}{T}$). Donc dans ce cas on a $\tau > T$, ce qui permet le passage du courant à travers le M.C.C durant toute la période.

5.4. Conclusion

Dans ce chapitre, nous avons utilisé la carte Arduino UNO en acquisition et en restitution, cela nous a permis de vérifier sa réponse en charge concernant surtout les fréquences de découpages.

Durant notre expérience on a constaté qu'en basse fréquence le moteur est bruyant surtout durant le démarrage, en régime permanent le bruit est légèrement atténué. Par contre pour la haute fréquence le moteur est silencieux. Cela est dû au fait que l'ondulation du courant est inversement proportionnelle à la fréquence de découpage, et qu'au niveau des moteurs à courant continu la qualité du couple est toujours dictée par celle de son courant d'alimentation. D'où l'intérêt des fonctionnements des convertisseurs à haut fréquence. Cela est dû au fait que dans les filtres électriques ce sont les réactances qui s'opposent au variations des grandeurs d'états plutôt que les inductances ou les capacités.

Conclusion Générale

Conclusion générale

Les convertisseurs statiques en électronique de puissance, depuis leur apparition, sont en perpétuelle évolution surtout ces dix dernières années. L'une des raisons est l'intérêt apporté par les concepteurs et les formateurs à l'évolution de l'informatique industrielle, introduisant ainsi les microcontrôleurs dans tous les domaines utilisant l'électricité comme support pour transmettre l'information. Cela nous ramène à nous intéresser à la carte Arduino. Depuis son premier prototype en 2005 la carte Arduino n'a cessé d'attirer l'attention sur elle, qu'il s'agisse des néophytes ou des professionnels, elle tend à occuper l'esprit de tout utilisateur de microcontrôleur.

Pour nous électrotechniciens, la question posée est : Comment utiliser les cartes Arduinos dans la gestion d'énergie ? Cette gestion étant réalisée par des convertisseurs statiques donc cela revient à se reposer la question autrement : Comment utiliser les cartes Arduinos comme organe de contrôle des convertisseurs statiques ?

L'électronique de puissance est une branche multidisciplinaire, le but principal est le contrôle du flux énergétique entre une source est une charge. La question qu'on se pose est la suivante qu'elle est la fonction de conversion d'un convertisseur, sachant que cette fonction est dictée principalement par les fonctions de connexions c'est-à-dire des semi-conducteurs, ces états proviennent de l'organe de contrôle qui doit synthétiser les besoins de la charge en fonction de ce que peut fournir la source.

Donc une analyse fine doit être faite en fonction des informations que l'ensemble des capteurs fournissent, le microcontrôleur qui est l'organe de synthèse de l'information doit fonctionner avec beaucoup de précision il faut bien connaitre toutes ces caractéristiques afin de ramener les ajustements nécessaires.

Ces raisons nous ont amenés à structurer notre travail comme suit, après un premier chapitre d'introduction, nous avons consacré le deuxième chapitre à la présentation des cartes Arduinos principalement la UNO qui est la plus conviviale et la DUE la plus performante, nous avons présenté leurs principales caractéristiques, le troisième chapitre est consacré à l'utilisation d'un langage évolué en l'occurrence Matlab/Simulink pour la programmation de nos cartes, ce langage est limité car la transmission des codes n'est pas directe. C'est cela qui nous a conduit à utiliser dans le quatrième chapitre un langage machine en l'occurrence, le langage Arduino spécialement conçu pour ces cartes. Des acquisitions et des restitutions des signaux ont était réalisées afin d'apprécier la précision de ces cartes, au quatrième chapitre nous avons aussi implanté un algorithme de Modulation de Largeur d'Impulsion, cet

algorithme est très exigent concernant les périodes d'échantillonnages, dans le pire des cas elles doivent être de l'ordre du dixième de la microseconde (1e-5), dans nos applications on a réussi à générer avec la carte DUE des signaux allons jusqu'à une fréquence de 200 kHz. Cette gamme de frequence répond largement aux exigences imposées par ces techniques. Ensuite une application sur un moteur à courant continu de faible puissance alimenté par un hacheur quatre quadrants a été réalisée.

Comme perspective à ce travail :

- Reprendre la DUE est analyser ses performances en acquisition, cela nous permettra éventuellement de comprendre l'origine du facteur de proportionnalité de la base de temps qui est de l'ordre de 1,3 que nous avons rencontré dans le cas des cartes UNO ;
- Usage de deux cartes simultanément en maitre/esclave l'une doit être programmée avec un langage évolué pour les boucles de régulation l'autre en langage machine pour la gestion des signaux PWM ;
- Réalisation d'un convertisseur statique répondant à un cahier des charges dans le milieu industriel en utilisant les cartes Arduinos comme organe de contrôle.

Bibliographie

[1] Giorgio Olivero, Mario Ciardulli, Vanessa Poli, Michelle Nebiolo, Stefania Vulpi <<le livre de projets arduino>>.

[2] B.KHERKHOUR Z.ZERIOUL <<Conception et réalisation d'un système Domotique à base Arduino>>, Mémoire de fin d'étude de Master à l'UMMTO, 2015/2016.

[3] E.Birtmann<< LE GRAND LIVRE D'ARDUINO>>- 2eme édition, Eyrolles, 2013.

[4] Michel Pinard <<Convertisseurs et électronique de puissance. Commande, Description et Mise en œuvre>>. Edition DUNOD 2007.

[5] R. Kechenoura, <<étude des pertes dans les convertisseurs statiques>>, Thèse de doctorat juin 2015 UMMTO.

[6] G.Séguier, F.labrique, Ph.Delarue, électronique de puissance <<Structure, commandes, applications. 10^{e} édition DUNOD 2015.

[7] G.Séguier, F.labrique, B.Robert,<<les convertisseurs de l'électronique de puissance>> Conversion Continu-Alternatif, Volume 4, Edt&Tec-1995.

[8] Astalaseven, Eskimon et Olyte << Arduino pour bien commencer en électronique et en programmation>>, License Créative Commons BY-NC-SA 2.0, 04/08/2012.

[9] M. Salah-Eddine <<Réalisation d'un onduleur de tension monophasé contrôlé par une carte Arduino>>, Mémoire de fin d'étude Universalité de Constantine 1, 2013/2014.

[10] Livret Arduino en français par Jean-Noël Montagné, Centre de Ressources Art Sensitif, novembre 2006, sous licence CC.

[11] CH. Nizar, CH. Amine <<commande d'un système thermique à l'aide de la carte Arduino UNO>>, FORMATION ARDUINO-MATLAB/SIMULINK , Hammamet ¾ Mai 2014.

[12] L. Frédéric <<Génération d'un signal par modulation de largeur d'impulsion>>, Licence Créative Commons.

[13] S. Fabrice <<Electrotechnique>> ; version 3.0.5. http://pagesperso-orange.fr/fabrice.sincere/

Bibliographie

Site internet

[1] www.fritzing.org

[2] http://www.locoduino.org/spip.php?article57

[3] www.wikidebrouillard.org

[4] www.arduino.cc

[5] www.eskimon.fr

[6] www.arduino.org

[7] https://framablog.org/2011/12/10/arduino-histoire/

[8] www.wikipedia.org

[9] www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielUno

[10] www.generationrobots.com/fr/401275-carte-arduino-due.html

[11] https://fr.wikipedia.org/wiki/Microcontr%C3%B4leur

[12] B. Cottenceau B311 ISTIA bertrand.cottenceau@univ-angers.fr

[13] https://fr.wikipedia.org/wiki/Microcontr%C3%B4leur

[14] www.Mathworks.com

[15] http://www.alldatasheet.com/view.jsp?Searchword=Lm358n

[16] http://www.ti.com/lit/ds/symlink/l293.pdf

[17] https://www.carnetdumaker.net/articles/la-conversion-analogique-numerique-avec-arduino-genuino/

[18] https://f-legrand.fr/scidoc/docmml/sciphys/arduinodue/can/can/.html

[19] http://www.ile-reunion.org

[20] http://physique.vije.net

[21] http://www.plexim.com

[22] http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE

[23] http://www.powersimtech.com

[24] http://www.euedia.tuiasi.ro/lab_ep/ep_files/Lucrarea_6_img.pdf

Annexe

Annexe A

TXROBOTIC
ROBOTS PARTS & TOOLS

Tableau comparatif des différentes cartes Arduino

Cartes Arduino / Caractéristiques	UNO R3 (classique & CMS)	UNO R3 Ethernet (classique & POE)	Leonardo	Mega 2560	Mega ADK	DUE	Esplora	Mini	Nano	Yun (classique & POE)	Zero PRO
Microcontrôleur	ATmega328P	ATmega328P	ATmega32u4	ATmega2560	ATmega2560	AT91SAM3X8E	ATmega32u4	ATmega328P	ATmega328P	ATmega32u4	ATSAMD21G18
Cadencement Horloge	16 MHz	16 MHz	16 MHz	16 MHz	16 MHz	84 MHz	16 MHz	16 MHz	16 MHz	16 MHz	48 Mhz
Tension d'entrée	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 9V	7 - 9V	5V	5V
Tension de fonctionnement	5V	5V	5V	5V	5V	3,3V	5V	5V	5V	5V	3,3V
Entrée/Sortie Numérique	14/6	14/4	20/7	54/15	54/15	54/12	✗	14/6	14/6	20/7	14/12
Entrée-Sortie (PWM) Analogique	6/0	6/0	12/0	16/0	16/0	12/2 (DAC)	✗	8/0	8/0	12/0	6/1 (DAC)
Mémoire vive (Flash)	32 Ko	32 Ko	32 Ko	256 Ko	256 Ko	512 Ko	32 Ko	32 Ko	32 Ko	32 Ko	256 Ko
Mémoire vive (SRAM)	2 Ko	2 Ko	2,5 Ko	8 Ko	8 Ko	96 Ko	2,5 Ko	2 Ko	2 Ko	2,5 Ko	32 Ko
Mémoire morte (EEPROM)	1 Ko	1 Ko	1 Ko	4 Ko	4 Ko	✗	1 Ko	1 Ko	1 Ko	1 Ko	16 Ko
Interface USB	USB-B mâle	USB-B mâle	Micro-USB	USB-B mâle	USB-B mâle & USB-A pour Androïd	2 ports micro-USB (Native et programming)	Micro-USB	✗	Mini-USB	Micro-USB	2 ports micro-USB (Native et programming)
Port UART	1	1	1	4	4	4	✗	✗	1	1	2
Carte SD	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
Ethernet	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
Wi-Fi	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Dimensions	68x53mm	68x53mm	68x53mm	101x53mm	101x53mm	101x53mm	165x60mm	30x18mm	45x18mm	68x53mm	68x53mm

Comme on peut le voir sur le tableau ci dessus, certaines cartes sortent du lot. **Les caractéristiques en rouge** sont les paramètres clés pour affiner votre choix sur la carte Arduino à choisir.

Annexe B

- **Programme pour la commande de M.C.C** [1]

```
const int controlPin1=2;
const int controlPin2=3;
const int enablePin=9;
const int directionSwitchPin=4;
const int onOffSwitchStateSwitchPin=5;
const int potPin=A0;

int onOffSwitchState=0;
int previousOnOffSwitchState=0;
int directionSwitchState=0;
int previousDirectionSwitchState=0;
int motorEnabled=0;
int motorSpeed=0;
int motorDirection=1;
void setup(){
  pinMode(4,INPUT);
  pinMode(5,INPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(9,OUTPUT);
  digitalWrite(9,LOW);
}
void loop(){
  onOffSwitchState=
  digitalRead(onOffSwitchStateSwitchPin);
  delay(1);
  directionSwitchState=
```

```
 digitalRead(4);
 motorSpeed=analogRead(A0)/4;
 if(onOffSwitchState!=previousOnOffSwitchState){
  if(onOffSwitchState==HIGH){
   motorEnabled=!motorEnabled;
  }
 }
 if(directionSwitchState!=previousDirectionSwitchState){
  if(directionSwitchState==HIGH){
    motorDirection=!motorDirection;
  }
 }
 if(motorDirection==1){
  digitalWrite(controlPin1,HIGH);
  digitalWrite(controlPin2,LOW);
  }
  else{
  digitalWrite(controlPin1,LOW);
  digitalWrite(controlPin2,HIGH);
  }
  if(motorEnabled==1){
    analogWrite(enablePin,motorSpeed);
  }
  else{
    analogWrite(enablePin,0);
  }
previousDirectionSwitchState=directionSwitchState;
previousOnOffSwitchState=onOffSwitchState;
 }
```

- **<u>Programme pour gérer une MLI à 200 KHz pour la DUE</u>** [12]

```
#include "Arduino.h"
#define MAX_PWM_VALUE 400
// pin 7 = PC23 = PWML6
// pin 45 = PC18 = PWMH6
// pin 8 = PC22 = PWML5
// pin 44 = PC19 = PWMH5
#define NECHANT 128
#define SHIFT_ACCUM 25
uint32_t table_onde[NECHANT];
uint32_t accum1,accum2,increm;
volatile void (*TC0_function)();
char pwm_chan[2] = {6,5};
char pwm_pin_1[2] = {7,8};
char pwm_pin_2[2] = {45,44};
void declencher_timer(uint32_t ticks, volatile void (*function)()) {
uint8_t clock = TC_CMR_TCCLKS_TIMER_CLOCK1; // horloge 84MHz/2=42 MHz
uint32_t channel = 0;
TC0_function = function;
pmc_set_writeprotect(false);
pmc_enable_periph_clk((uint32_t)TC0_IRQn);
TC_Configure(TC0, channel, TC_CMR_WAVE | TC_CMR_WAVSEL_UP_RC | clock);
TC0->TC_CHANNEL[channel].TC_RC = ticks;
TC_Start(TC0, channel);
TC0->TC_CHANNEL[channel].TC_IER=TC_IER_CPCS;
TC0->TC_CHANNEL[channel].TC_IDR=~TC_IER_CPCS;
NVIC_EnableIRQ(TC0_IRQn);
}
void TC0_Handler()
```

```
{
TC_GetStatus(TC0, 0);
(*TC0_function)();
}
void init_pwm(int i, uint32_t frequency, uint32_t ulValue) {
pmc_enable_periph_clk(PWM_INTERFACE_ID);
PWMC_ConfigureClocks(frequency*MAX_PWM_VALUE,0,VARIANT_MCK);
uint32_t chan = pwm_chan[i];
uint32_t ulPin1 = pwm_pin_1[i];
uint32_t ulPin2 = pwm_pin_2[i];
PIO_Configure(g_APinDescription[ulPin1].pPort,
g_APinDescription[ulPin1].ulPinType,
g_APinDescription[ulPin1].ulPin,
g_APinDescription[ulPin1].ulPinConfiguration);
PIO_Configure(g_APinDescription[ulPin2].pPort,
g_APinDescription[ulPin1].ulPinType,
g_APinDescription[ulPin2].ulPin,
g_APinDescription[ulPin2].ulPinConfiguration);
PWMC_ConfigureChannel(PWM_INTERFACE, chan, PWM_CMR_CPRE_CLKA, 0, 0);
PWM_INTERFACE->PWM_CH_NUM[chan].PWM_CMR |= PWM_CMR_DTE; // dead time enable
PWMC_SetPeriod(PWM_INTERFACE, chan, MAX_PWM_VALUE);
PWMC_SetDutyCycle(PWM_INTERFACE, chan, ulValue);
//uint32_t dead_time = MAX_PWM_VALUE/20;
//PWMC_SetDeadTime(PWM_INTERFACE, chan, dead_time, dead_time);
PWMC_EnableChannel(PWM_INTERFACE, chan);
}
volatile void synthese_table() {
accum1+= increm;
accum2+= increm;
```

```
PWM_INTERFACE->PWM_CH_NUM[pwm_chan[0]].PWM_CDTYUPD =
table_onde[accum1 >> SHIFT_ACCUM];

PWM_INTERFACE->PWM_CH_NUM[pwm_chan[1]].PWM_CDTYUPD =
table_onde[accum2 >> SHIFT_ACCUM];

}

void set_sinus_table(float amp) {

int i;

float dt = 2*3.1415926/NECHANT;

for(i=0; i<NECHANT; i++) {

table_onde[i] = MAX_PWM_VALUE*0.5*(1.0+amp*sin(i*dt));

}

}

void setup() {

set_sinus_table(1.0);

uint32_t pwm_freq = 200000; // fréquence du PWM en Hz

init_pwm(0,pwm_freq,0);

init_pwm(1,pwm_freq,0);

uint32_t frequence = 50; // Fréquence de la sinusoïde en Hz

uint32_t fechant = 42000; // Fréquence d'échantillonnage en Hz

uint32_t techant = 1.0/fechant; // en secondes

uint32_t ticks = 42000000/fechant; // nombre de tops d'horloges à 42 MHz pour la période
increm = (uint32_t) (((float)(0xFFFFFFFF))*((float)(frequence)*techant)); // incrément

accum1 = 0;

accum2 = ((uint32_t)(NECHANT * 0.25)) << SHIFT_ACCUM;

declencher_timer(ticks,synthese_table);

}


void loop() {


}
```