الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي و البحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة أبي بكر بلقايد – تلمسان –

Université Aboubakr Belkaïd – Tlemcen –

Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : (Electrotechnique)

Spécialité : (Commandes Electriques)

Par : Lallam Mohammed El-Mehdi
Saidi Benali

Sujet

Réalisation d'un onduleur triphasé piloté à travers une interface graphique

Soutenu publiquement, le / / , devant le jury composé de :

Mr MECHERNENE Abdelkader	MCA	Université de Tlemcen	Président
Mr LOUCIF Mourad	MCB	Université de Tlemcen	Examinateur
Mr MELIANI Sidi Mohammed	Pr	Université de Tlemcen	Encadreur

Année universitaire : 2021 / 2022

Dédicaces I

J’ai le grand plaisir de dédier ce modeste travail

A mes très chers parents, qui me donne toujours l’espoir de vivre et qui n’ont jamais cessé de prier pour moi.

A mes frères et mes sœurs.

A tous mes professeurs qui m’ont aidé durant tout le cursus.

Mehdi

Dédicaces II

Je dédie ce modeste travail

A mes parents, qui m'ont encouragé à aller de l'avant et qui m'ont donné tout leur amour pour faire mes études, que Dieu les préserve en Bonne santé.

A mon frère et mes sœurs qui m’ont toujours soutenu durant toutes ces années.

Benali

Remerciements

Je remercie Dieu le tout puissant de m'avoir donné le courage et la patience qui m'ont permis d'accomplir ce modeste travail.

La réalisation de ce mémoire a été possible grâce à l'aide de plusieurs personnes à qui nous voudrons témoigner toute notre reconnaissance.

Nous voudrons tout d'abord adresser toute notre gratitude à l'encadreur de ce mémoire, Mr Meliani Sidi Mohamed, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter notre réflexion.

Nos profonds remerciements vont aux membres du Jury qui nous 'ont honoré de leur participation à notre jury de mémoire Mr Mchernen Abdelkader et Mr Loucif Mourad et pour avoir accepté d'évaluer ce travail.

Nous tenons à remercier également nos professeurs qui nous ont appris tout ce qu'on sait sur ce parcours de cinq ans, ils ont grandement facilité notre travail.

Table des matières

Table des Figure

Liste des tableaux

Nomenclatures

Indices et exposants

Symbole	Signification
Coef	*Rapport V sur f*
Tension $_{Desirée}$	*Tension imposer par l'utilisateur lors de la commande*
Fré $_{Desirée}$	*Tension imposer par l'utilisateur lors de la commande*
Fré	*Fréquence du moteur*
Step	*pas de calcul*
V_alpha	*Tensions numériques dans le référentiel α, β*
V_as	*Tensions numériques dans le référentiel a, b,*
angle(θ)	*angle de calcul*

Glossaire

Acronyme	Signification
CA	*Courant Alternatif*
CC	*Courant Continu*
PWM	*Pulse Width Modulation*
MOSFET	*Transistor à effet de champ à grille métal-oxyde*
IGBT	*Insulated Gate Bipolar Transistor*
BJT	*Transistors Jonction Bipolaire*
N	*Neutre*
O	*Point milieu*
PWM	*Pulse width Modulation*
HIN	*Logic input for high side gate driver output (HO), in phase*
LIN	*Logic input for low side gate driver output (LO), in phase*
SD	*Logic input for shutdown*
HO	*High side gate drive output*
LO	*Low side gate drive output*
BF	*Base Fréquence*
COM	*LOW side return*
ADC	*Analog-to-Digital Converter*
PDC	*PWMx Generator Duty Cycle Value bits*

Introduction général et Etat de l’art

A - Introduction

Depuis l'heure de l'industrialisation, Les entreprises sont soumises à des rudes épreuves de concurrence, elles doivent proposer des produits de qualité en temps réduit en assurant la protection et la sécurité des installations et du personnels, les chercheurs ont été affronté au "comment commander les machines électriques à des vitesses variables". Car les entraînements électriques exigent de plus en plus de hautes performances, une fiabilité accrue, et un coût réduit [1]

Ces contraintes ont donc orienté la recherche dans le domaine de la vitesse variable vers les machines à courant alternatif, et plus particulièrement vers les machines asynchrones.
Pour assurer ces critères, Une politique de gestion a été mise au point sur les équipements, ces applications industrielles nécessitent des variateurs de vitesse ayant des hautes performances dynamiques.

Les progrès réalisés en matière d'électronique de puissance et de circuits de commande ont Contribué à l'utilisation grandissante des machines asynchrones dans les systèmes d'entraînements électriques. Le recours aux machines asynchrones est surtout lié à leur robustesse, leur puissance massique et à leur coût de fabrication.[2]

B - Etat d'art

En début des année 1885, Galileo Ferraris un ingénieur et scientifique italien découvreur du champ magnétique tournant et créateur du moteur électrique en courant alternatif, donne naissance au premier modèle de moteurs électrique à système biphasée en utilisant le champ magnétique rotatif nommé moteur à induction ou moteur asynchrone. [3]

Cependant en 1888, Nikola tesla a été breveté en créant le premier modèle du système triphasé pour une utilisation pratique à l'échelle industrielle, un modèle très efficace et fiable avec une structures simple et qui nécessite peu pour sa maintenance. Sa particularité est qu'il peut être exécuté par connexion directe à une alimentation triphasée. [4]

À cette époque entre 1950 et 1970, Juste après la mise au point du moteur à courant alternatifs, l'idée de varier la vitesse a été envisagée par la variation de la fréquence, en utilisent

un moteur à courant continu qui fait tourner un alternateur à courant alternatifs qui génère une tension à fréquence variable. Cette fréquence est proportionnelle à la vitesse du courant de l'alternateur, comme le montre la relation suivante :

$$Ns = \frac{f}{p}$$

Cependant, l'utilisation principale de ces lignes était dans les lignes multimoteurs à commande de précision, et lorsque la fréquence de l'alternateur était variée, tous les moteurs suivaient ensemble avec une précision synchrone. [5]

Le contrôle de la vitesse du moteur en courant alternatif n'était pas un processus simple et qui coute très cher. Après de nombreuse tentative et essai, dans les milieux des années 80, une nouvelle méthode s'applique nommée la commande à fréquence variable statiques, ce dernier était facilement disponible avec un concept à tension variable.

Malgré ces efforts, le contrôle des moteurs à induction était une tâche très difficile, leur application était limitée par rapport au moteur à courant continu à cette époque.

Figure 1 : Variateur de vitesse d'un MAS à l'ancien époque

Dans les débuts des années 1960, l'utilisation des composant semi-conducteur a commencé à réaliser des exploits remarquables dans les domaines de la commande des moteurs, grâce au chercheur martti harmoinene qui a créé le première modèle de variateur de fréquence PWM. [6]

C'était le premier pas de cette nouvelle création qui a poussé le développement des signaux PWM codé en sinus qui sont devenus la norme pour commander les variateurs de vitesses.

Ce qui a fait une flambée dans le développement des topologies des entraînements dans les années 1990. Dans tous les domaines la commutation des semi-conducteur, les techniques de simulations et de contrôle.

Ce qui a rendu les variateurs de vitesse plus fiables et peu coûteux en matière, Et leur utilisation a augmenté petit à petit pour la variation de la vitesse des moteur CA.

Au début des années 1990, les microprocesseurs, tels que les 8080, 8031, 8098, 80196 d'Intel et le 68000 de Motorola, étaient principalement utilisés dans les applications de commande de moteurs, puis sont venus les microcontrôleurs qui ont été utilisés pour les applications d'entraînement variable jusqu'à la fin du XX^{eme} siècle.

Dans notre travaille nous avons utilisé comme microcontrôleur ESP 32

C - Qu'es qu'un ESP32 ?

Créée par Espressif Systems, la famille ESP32 est une série de systèmes sur puce (SoC) à faible coût et faible consommation, avec des capacités Wi-Fi et Bluetooth bi-mode. Dans son cœur, on trouve un microprocesseur Tensilica Xtensa LX6 à double ou simple cœur, avec une fréquence d'horloge allant jusqu'à 240 MHz. L'ESP32 est hautement intégré avec des commutateurs d'antenne intégrés, un balun RF, un amplificateur de puissance, un amplificateur de réception à faible bruit, des filtres et des modules de gestion de l'alimentation. Conçu pour les appareils mobiles, l'électronique vestimentaire et les applications IoT, l'ESP32 atteint une consommation d'énergie ultra-faible grâce à des fonctions d'économie d'énergie, notamment le gating d'horloge à résolution fine, les modes d'alimentation multiples et l'échelonnement dynamique de la consommation. [7]

Apres avoir donné un historique sur le variateur de vitesse et leur développement sur le fil du temps, dans ce qui suit, nous allons parler des moteur asynchrones et sa relation avec le variateur de vitesses.

D - Aperçu sur les moteurs asynchrones

Les moteurs asynchrones triphasés crées en 1887, appelé aussi moteur à induction ou à rotor en court-circuit, grâce à des phénomènes électromagnétiques il transforme l'énergie électrique en énergie mécanique.

Ces moteurs sont les plus employés dans l'industrie, Ils possédants en effet plusieurs avantages : simplicité, robustesse, prix peu élevé et entretien facile, avec une vitesse de rotation presque constante sur une large plage de puissances. [8]

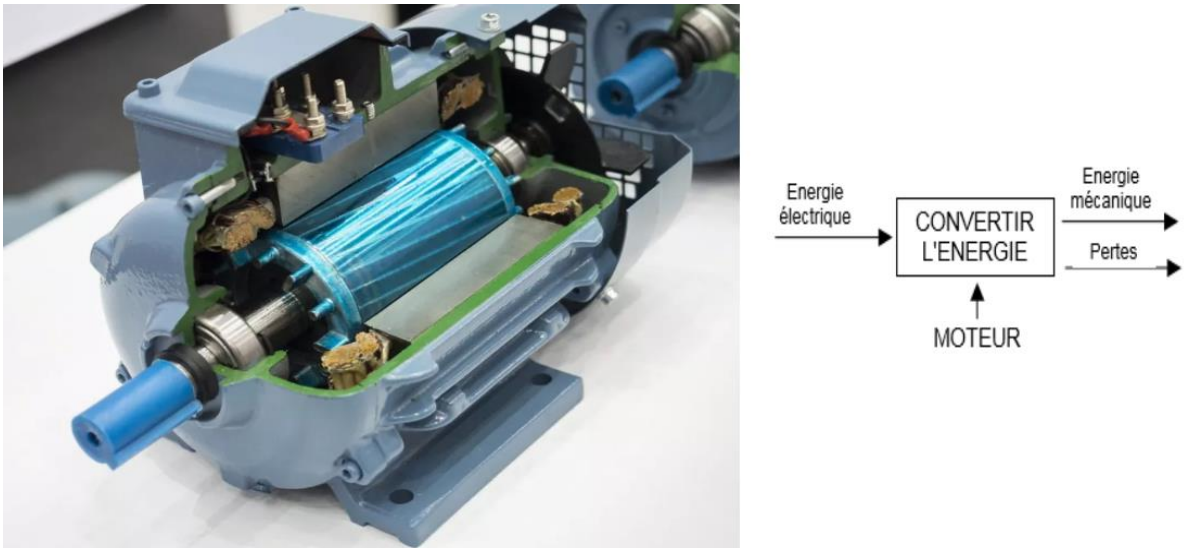


Figure 2 : Moteur asynchrone triphasé

D.1 - Constructions

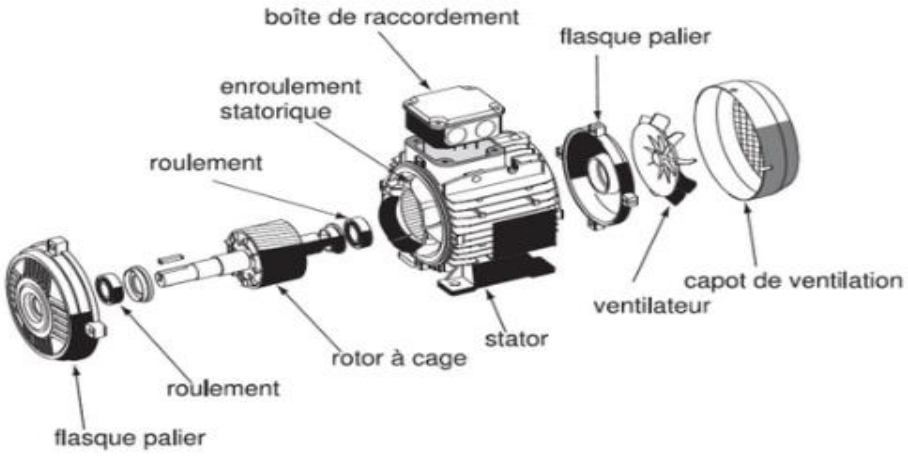


Figure 3 : Constitution du moteur asynchrone triphasé

- **Stator**

C'est la partie fixe du moteur, il est formé d'une carcasse ferromagnétique qui contient trois enroulements électriques alimentés chacune par une phase de tension U et d'une fréquence f.

Le passage de courant dans les enroulements crée un champ magnétique tournant $\vec{B}$ à l'intérieur du stator, sa vitesse de rotation n_s s'appelle vitesse de synchronisme.

$$Ns = \frac{f}{p} \quad ou \quad \omega s = 2 \times \pi \times ns$$

Ns : fréquence de rotation du champ $\vec{B}$ s'exprime en tours/s.

f : fréquence de tension d'alimentation des bobines en Hertz (Hz).

p : nombre de paires de pôles du rotor.

ωs : vitesse de synchronisme s'exprime en (rad/s).

- **Rotor**

C'est la partie tournante du moteur asynchrone, il tourne moins vite que le champ tournant du stator, sa vitesse est :

$$n < Ns$$

Il porte soit un bobinage accessible par trois bagues et trois balais, soit une cage d'écureuil non accessible à base des barres conductrices en aluminium. La vitesse angulaire relative du champ par rapport au rotor est la vitesse angulaire de glissement :

$$\omega g = \omega s - \omega r \rightarrow (rad/s)$$

$$ng = ns - nr \rightarrow (Tr/s)$$

D.2 - Principe de fonctionnement

Le champ magnétique tournant $\vec{\boldsymbol{B}}$ crée par les bobinages statoriques balaie le bobinage rotorique et induit des forces électromotrices (f.é.m.) d'après la loi de Lenz. Le bobinage rotorique étant en court-circuit, cette f.e.m produisent des courants induits. L'action de ce champ sur les courants induits crée le couple moteur. Ce dernier tend à réduire la cause qui a donné

naissance aux courants, c'est à dire la rotation relative du champ tournant par rapport au rotor. Le rotor va donc avoir tendance à suivre ce champ.

D.3 - Bilan de puissance

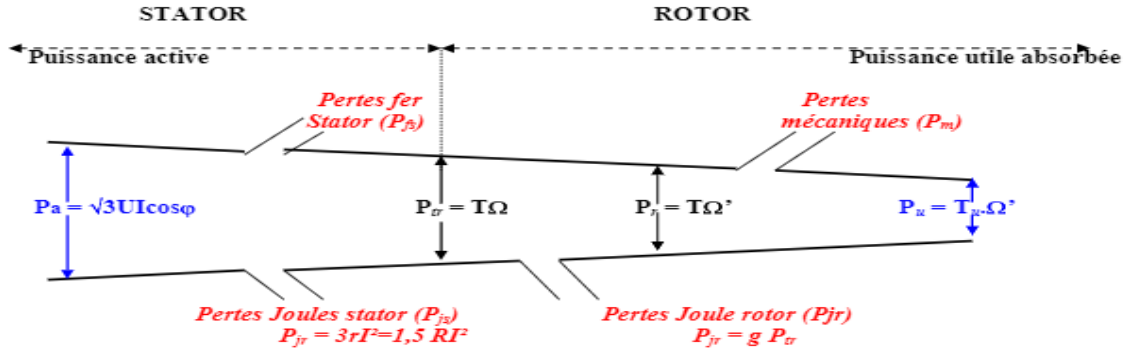


Figure 4 : Bilan de puissance moteur asynchrone

Cependant, ces moteurs ont une vitesse pratiquement constante et ils se prêtent assez mal au réglage de la vitesse, à l'aide des systèmes d'entraînement électroniques (variateurs de vitesse, démarreurs ralentisseurs) qui permettent de faire varier la vitesse des moteurs d'induction.

Aujourd'hui, nous allons faire appel à des variateurs de vitesse électroniques, qui sont plus performant par rapport à l'ancienne régulation mécanique. A l'aide des composants électroniques, nous avons pu se perfectionner pour attendre des régulations de vitesse précises. Pour les MAS, la vitesse mécanique du rotor est liée à la fréquence des courants au stator. Ce lien mathématique rend possible une commande de la vitesse du rotor par la commande de la fréquence du courant au stator. C'est ce que l'on appelle la condition de synchronisme qui s'exprime différemment selon que l'on considère une machine synchrone ou une machine asynchrone.

Le recours aux variateurs de vitesse offre plusieurs avantages : comme démarrage progressif qui réduit les chutes de tension dans le réseau et limité les courant de démarrage une bonne régulation de vitesse et une amélioration sur le facteur de puissance et une augmentation de la dure de vie des machines et qui diminue la consommation de l'électricité.

F - Objectif de notre travail

L'objectif principale de notre projet de fin d'étude est la réalisation d'un onduleur triphasé piloté à travers une interface graphique, qui sera utilisé comme un variateur de vitesse a distance via WIFI, basé sur la commande scalaire.

Pour atteindre cet objectif, nous sommes passés de la conception du banc d'essai par le logiciel PROTEUS ISIS et ARESS pour le circuit PCB, jusqu' à la réalisation pratique.

Notre mémoire comporte trois chapitres et une conclusion générale.

- Le premier chapitre donne un aperçu sur historique des variateurs des vitesse, ainsi qu'un aperçu sur les moteurs asynchrone.
- Le deuxième chapitre explique les choix de technologies et l'implémentation de la commande scalaire pour l'onduleur triphasé.
- Le dernier chapitre contient les différents circuits électroniques qui ont été conçus à l'aide du logiciel PROTEUS et leur réalisation pratique avec tous les résultats obtenus.

Chapitre I

Variateur de vitesse à base d'électronique de puissance

1.1 Introduction

Le variateur de vitesse est un dispositif électronique qui permet le contrôle de la vitesse du moteur au-dessus et/ou au-dessous de la vitesse nominale, en changeant la fréquence et l'amplitude de la tension d'alimentation du moteur en allant de (f=0 Hz) à la vitesse nominale du moteur (f=50 Hz).

Ils assurent une décélération progressive et permettent une adaptation précise de la vitesse aux conditions d'exploitations.

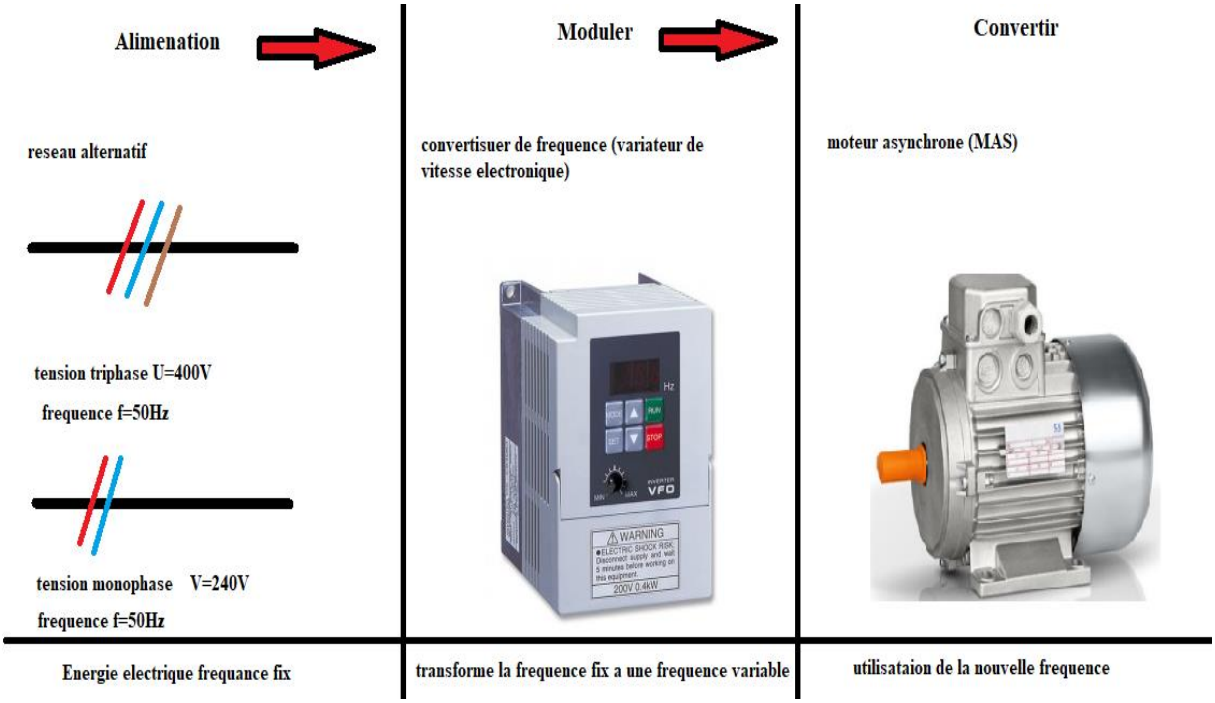


Figure I-5 : Variateur de vitesse avec l'alimentation du réseau et moteur asynchrone.

1.2 Structures du variateur de vitesse

Il se compose de deux paramètres essentiels : le circuit de puissance et le circuit de commande.

1.3 Bus continu

Premièrement, nous allons parler du circuit de puissance qui se compose d'un redresseur triphasé ou monophasé, d'un circuit intermédiaire, d'un onduleur et d'un driver.

Les redresseurs sont des convertisseurs statiques de l'électronique de puissance. Il permet la conversion AC / DC. On peut classer les redresseurs selon le nombre d'alternances et le

nombre de phases, pour le redresseur non commandable la valeur moyenne de la tension de sortie est constante. [9]

1.3.1 Redresseur monophasé

En monophasé, le redresseur le plus utilisés est le redresseur double alternance.

Les redresseurs double alternance : en général, on utilise un pont de Graetz.

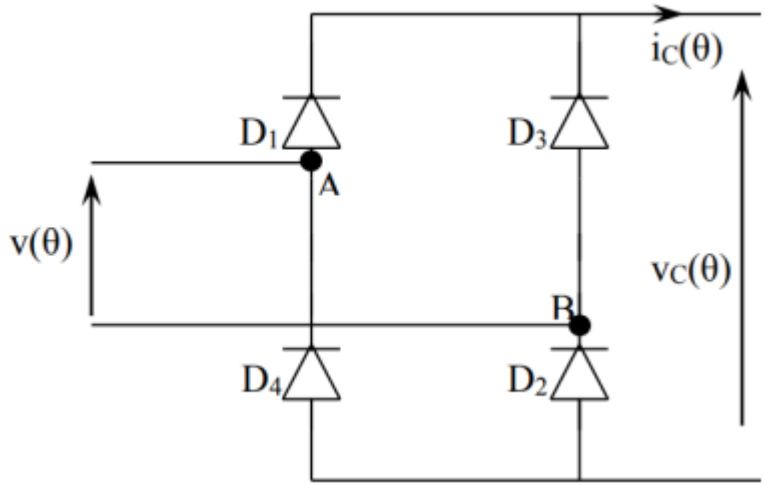


Figure I-6 : Schéma global d'un redresseur monophasé non commandé.

On peut détermine sa valeur moyenne en utilisant la formule suivante :

$$Vc_{moy} = \frac{2}{\pi} \times V_{max}$$

1.3.2 Redresseur triphasé

Il se compose de 6 diodes, et il se divise à 3 bras, chaque bras se compose de 2 diodes qui fonctionne en mode complémentaire, qui fournit une tension continue constante à ondulation résiduelle.

La structure du redresseur triphasé non commandé :

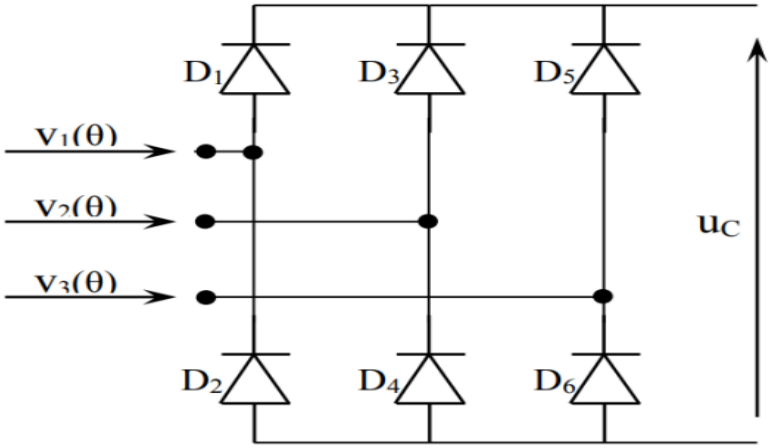


Figure I-7 : Schéma global d'un redresseur triphasé non commandé.

La valeur moyenne de ce type est calculée par la formule suivante :

$$Uc_{moy} = \frac{3\sqrt{3}}{\pi} \times V_{mox}$$

1.4 Circuit intermédiaire

C'est un circuit électronique qui nous donne la possibilité de lissée la tension continue à l'aide d'un condensateur, et qui permet le contrôle des niveaux de tension continuent qui aliment l'onduleur, il se situé entre le redresseur et l'onduleur.

Il offre des avantages :

-il se comporte comme un circuit de freinage pour annuler les surtensions lors des phases régénératives.

-Découpler le redresseur de l'onduleur

-Réduit les composante alternative et les harmoniques.

-Stocker l'énergie due aux pointes intermittentes de charge.

1.4.1 Le circuit intermédiaire à courant continu variable

Généralement ce type de circuit et utilisé pour le variateur a source de courant, il est composé d'une bobine et d'un filtre passe bas, se métabolisme sert a transformé la tension de sortie du redresseur a une tension DC.

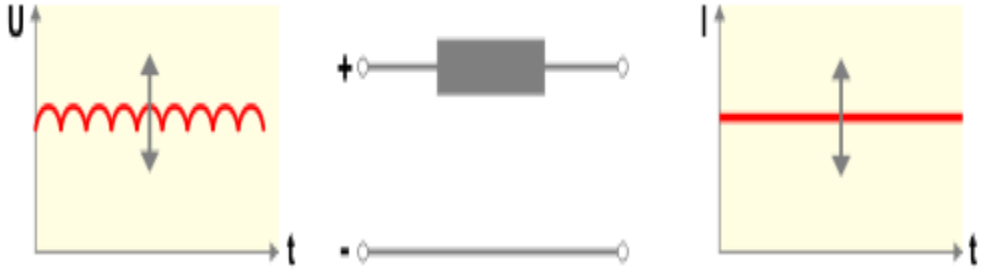


Figure I-8 : Circuit intermédiaire à courant continu variable.

1.4.2 Le circuit intermédiaire à tension continue constante ou variable

Ce type de circuit et différent par rapport au précédent, il est utilisé pour le variateur à source de tension. Nous avons toujours une bobine comme filtre passe bas et un condensateur comme filtre passe haut pour réduire les ondulations.

Pour un redresseur non-commandé, la tension à l'entrée de l'onduleur est une tension continue dont l'amplitude est constante. [10]

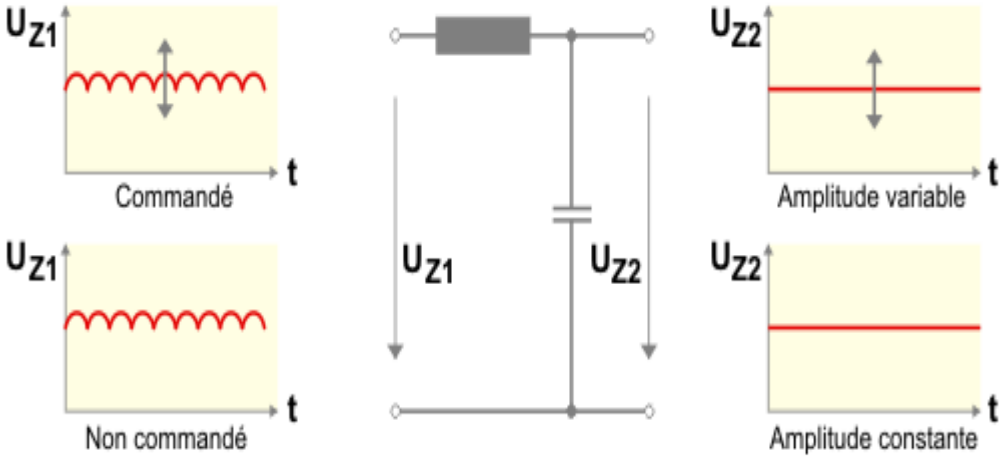


Figure I-9 : Circuit intermédiaire à tension continue constante ou variable.

1.5 Onduleur

Un onduleur de tension est un convertisseur statique à base de dispositif d'électronique de puissance, qui permet la conversion d'une tension continue en alternative.

Il modifie de façon périodique les connexions entre l'entrée et la sortie et permet d'obtenir l'alternatif à la sortie. [10]

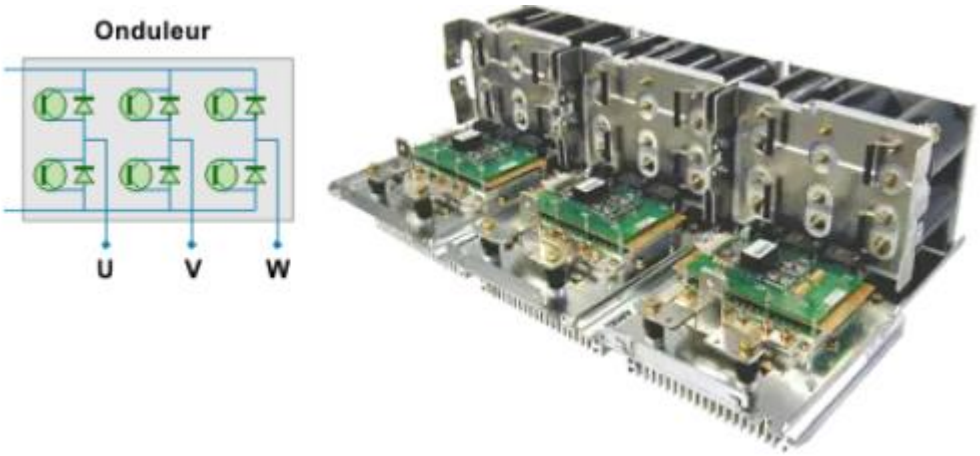


Figure I-10 : Onduleur.

1.5.1 Choix d'un onduleur

On peut choisir notre onduleur à l'aide de ces trois principaux éléments :

-Puissance nominale : il faut être conscient de la puissance totale consommé des appareils utilisés

-Tension d'entrée : la source d'alimentation continue

-Valeur et la forme de la tension de sortie : doivent correspondre à la tension des appareilles qui doivent utilisés.

1.5.2 Caractéristique d'un onduleur

Les onduleurs ont une conception qui se base sur leurs composants de puissance principaux constitués de MOSFET ou IGBT.

Pour ces composants ils sont de la même famille des transistors, mais chacun d'entre eux a ces propres applications, pour les MOSFET conviennent mieux aux applications à commutation rapide et basse tension, tandis que les IGBT sont plus adaptés aux applications haute tension à

commutation lente, et son oublié le BJT qui est en général utilisé dans les circuits amplificateurs.

En fonction de ces composants nous allons déterminer la puissance la tension et le courant maximale commuté.

Grâce au temps de commutation nous pouvons déterminer la fréquence maximale, cela revient à l'interruption des MOSFET qui sont commandé, ce qui crée un temps mort qui élimine le risque de court-circuit sur un bras.

Pour notre application, nous disposions de deux types d'onduleur, l'onduleur monophasé et le triphasé.

1.5.3 Onduleur monophasé

L'onduleur à pont complet se compose de deux bras, chacune contient deux inter- rupteurs commandés qui peuvent être des MOSFET, ou des IGBT à semi-conducteurs avec deux diodes de roue libre antiparallèles qui permettent d'assurer la continuité du courant lors du changement de sens.

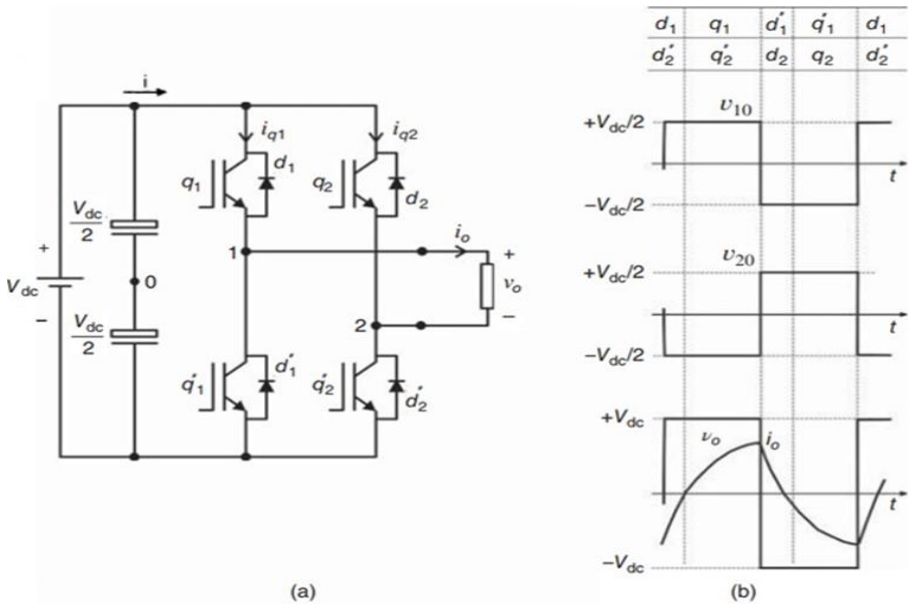


Figure I-11 : Onduleur monophasé à pont complet. (b) Formes d'onde principales

Les transistors q_1 et q_1' comme q_2 et q_2' sont complémentaires en théorie et pratique, c'est-à-dire lorsqu'un interrupteur est à l'état ouvert l'autre est à l'état fermé et inversement, ce qui signifie que $q_1 = 1 - q_1'$ et $q_2 = 1 - q_2'$. Donc, l'état de conduction de tous les commutateurs peut

être représenté par la variable binaire avec q_x = 1 indiquant un commutateur fermé et q_x= 0 indiquant un commutateur ouvert (avec x = 1 ou 2).

On a les tensions polaires qui se trouve entre le potentiel médian du bus continu (v_{10} et v_{20}) et le point de sortie des branches, ces tensions peuvent être déterminé en fonction des états de commutation qui se résume comme suit :

$$v_{10} = (2q_1 - 1) \times \frac{v_{dc}}{2}$$

$$v_{20} = (2q_2 - 1) \times \frac{v_{dc}}{2}$$

La tension de sortie est donnée par

$$v_{10} = v_{12} = v_{10} - v_{20}$$

Ou bien :

$$v_0 = (q_1 - q_2)v_{dc}$$

Grâce a ces paramètres nous pouvons obtenir des formes d'onde principale du convertisseur qui sont illustrées sur la figure (b), dans cette situation les tensions polaires v_{10} et v_{20} sont déphasées de 180 degrés avec un rapport cycliques de 50%.

Lorsque les tensions sont déphasées de Θ degré, comme indique la figure (a), cette figure contient 8 modes de fonctionnement.

Le 1er mode (b) : lorsque on commute q_1 et q_2' engendre une augmentation du courant de charge positif ($t_0 < t < t_1$), lorsque q_2' ce désactive il force q_1 à conduire le courant de charge avec d_2 en mode roue libre.

Le 2em mode (c) : ($t_1 < t < t_2$) jusqu'à ce que q_1 soit désactivé (t_2).

Le 3em mode (d) : en a d_2 et d_1' qui conduise ensemble, ce qui implique que la tension de charge devient négative avec un courant de charge force à zéro ($t_2 < t < t_3$).

Le 4em mode (e) : et déterminé par le passage à zéro dans lequel q_1' et q_2 sont activé ($t_3 < t < t_4$), ce qui fait d'augmenter le courant dans le sens négative.

q_2 et désactivé forçant q_1' à conduire le courant de charge avec d_2' en mode roue libre.

Le 5em mode ($t_4 < t < t_5$), jusqu'à ce que q_1' soit désactivé (t_5).

Le 6em mode, qui commande avec d_1 et d_2' conduisant ensemble, la tension de charge devient positive et le courant de charge est alors forcé à zéro ($t_5 < t < t_6$), Et ainsi de suite, jusqu'à la fin du cycle.

L'objectif de cette stratégie est l'ajout d'angle de déphasage Θ qui modifie l'amplitude des fondamentaux et des harmoniques de la tension de charge.

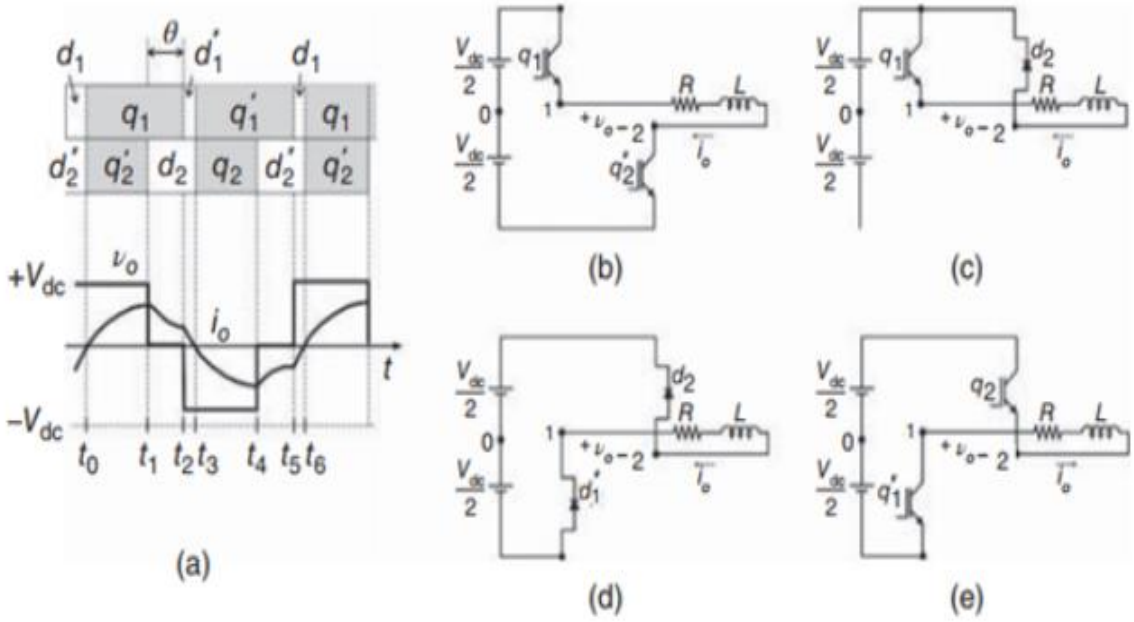


Figure I-12 : Onduleur monophasé à pont complet. (a) Signaux de déclenchement et formes d'onde principales. (b) Mode 1. (c) Mode 2. (d) Mode 3. (e) Mode 4.

Pour Calculer les valeurs de Vs et de Is on a :

Vs = U : définie comme la tension d'entrée de l'onduleur.

$I_s = I$: définie comme le courant d'entrée de l'onduleur.

$V_{c=}V_0$: définie comme la tension de sortie de l'onduleur

$I_c = I_0$: définie comme le courant de sortie de l'onduleur.

Donc en fonction de l'allure de $v_s = f(wt)$:

$$Vs_{moy} = \frac{1}{\pi}\int_0^{\pi} \sin(wt - \theta)\, dwt = \frac{2}{\pi} V_{cm} \cos(\theta) = \frac{2\sqrt{2}}{\pi} V_c \cos(\theta)$$

À pertes minimales :

$$P_s = P_c$$

Ce qui donne :

$$I_s Vs_{moy} = V_c I_c \cos(\theta)$$

$$I_s = \frac{\pi}{2\sqrt{2}} \times I_c$$

Avec Vc et Ic : valeurs efficaces de $vs\ et\ i_0$ respectivement

1.5.4 Onduleur triphasé

Pour réaliser un onduleur de tension triphasée en utilisant un diviseur capacitif commun et de trois demi-points monophasés en les regroupant pour réaliser 3 bras, chaque bras représente une phrase. On utilise ce type de diviseur pour les charges déséquilibrées.

Pour qu'on puisse maintenir les valeurs des courant continus « i'_a , i'_b, i'_c », on doit garder les commandes complémentaires par paire des interrupteur «k_1, k'_1 ,k_2 , k'_2 , k_3 , k'_3 ».

Pour assurer des tensions de sortie identique $v'_1 et\ v'_2\ et\ v'_3$ a un tiers du cycle T de leur fondamental, ce qui fait qu'on doit contrôler chaque demi-pont avec un retard de $T/3$ par rapport au précédent.

Les signaux de mise en marche sont donc appliqués à :

$$k1\ pour\ wt = 0\ avec\ k_1'\ pour\ wt = \pi$$

$$k2\ pour\ wt = \frac{2\pi}{3} avec\ k_2'\ pour\ wt = \pi + 2\pi/3$$

$$k3\ pour\ wt = \frac{4\pi}{3} avec\ k_3'\ pour\ wt = \pi + 4\pi/3$$

Si les courants « i_a' , i_b', i_c' » sont sinusoïdaux, ils forment un système triphasé équilibré :

$$i_a' = I'\sqrt{2}\sin\ (wt - \varphi)$$

$$i_b' = I'\sqrt{2}\sin\ (wt - \varphi - 2\pi/3)$$

$$i_c' = I'\sqrt{2}\sin\ (wt - \varphi - 4\pi/3)$$

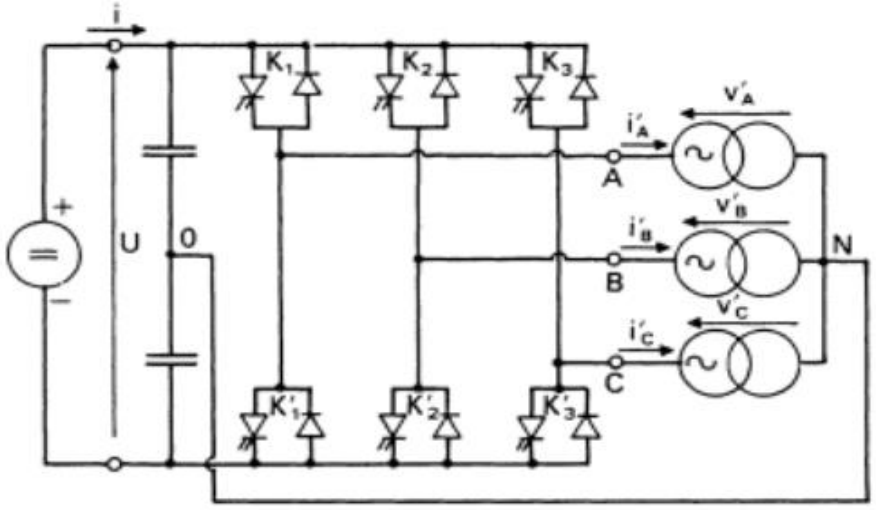


Figure I-13 : Onduleur de tension triphasé à diviseur capacitif.

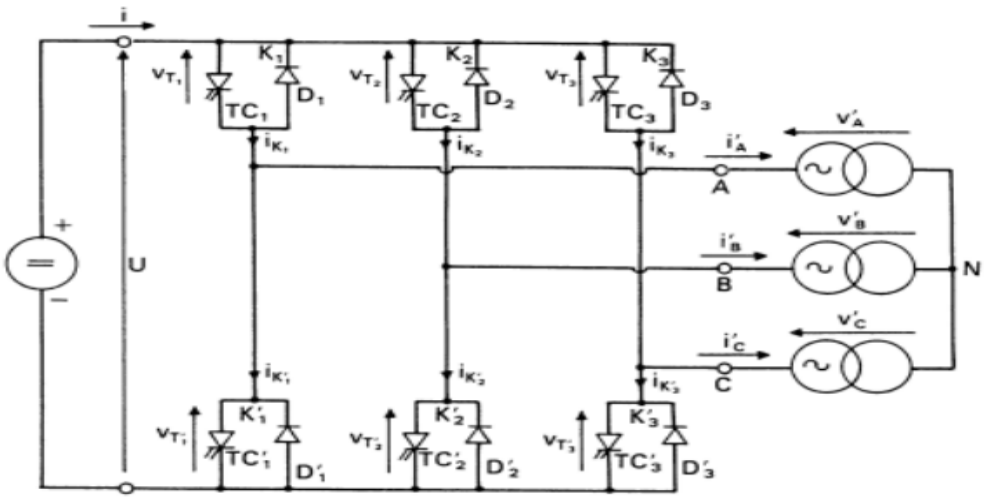


Figure I-14 : Onduleur de tension triphasé.

On montrera que, dans les cas où les courants sont équilibrés mais non sinusoïdaux, il est non seulement possible mais un avantage positif de supprimer la connexion O – N

Principe de fonctionnement

Quels que soient les courants, les interrupteurs imposent les tensions entre les bornes de sortie A, B, C et le point milieu « O » de la source de tension continue :

$v_A - v_0 = +\frac{U}{2}\ pour\ que\ k_1\ ferme\ et\ donc\ k_1'\ ouvert\ , ou\ -$
$\frac{U}{2}\ pour\ k_1\ ouvert\ et\ donc\ k_1'\ ferme$

$v_B - v_0 = +\frac{U}{2}\ pour\ que\ k_2\ ferme\ et\ donc\ k_2'\ ouvert\ , ou\ -$
$\frac{U}{2}\ pour\ k_2\ ouvert\ et\ donc\ k_2'\ ferme$

$v_C - v_0 = +\frac{U}{2}\ pour\ que\ k_3\ ferme\ et\ donc\ k_3'\ ouvert\ , ou\ -$
$\frac{U}{2}\ pour\ k_3\ ouvert\ et\ donc\ k_3'\ ferme$

Les interrupteurs imposent les tensions composées à la sortie de l'onduleur. Donc pour la première de ces tensions :

$$v_A' - v_B' = (v_A - v_0) - (v_B - v_0) = +U\ , si\ k_1\ ferme\ et\ k_2\ ouvert$$

$$v_A' - v_B' = (v_A - v_0) - (v_B - v_0) = 0\ , si\ k_1\ et\ k_2\ ferme\ ou\ k_1\ et\ k_2\ ouverts$$

$$v_A' - v_B' = (v_A - v_0) - (v_B - v_0) = -U\ , si\ k_1\ ouvert\ et\ k_2\ ferme$$

Pour un system équilibré :

$$i'_A + i'_B + i'_C = 0 \; ce\ qui\ implique\ v'_A + v'_B + v'_C = 0$$

Ce qui donne :

$$\frac{1}{3}(v'_A - v'_B) - \frac{1}{3}(v'_C - v'_A) = \frac{2v'_A}{3} - \frac{v'_B}{3} - \frac{v'_C}{3} = v'_A$$

Et de même :

$$v'_B = -\frac{v'_A}{3} + \frac{2v'_B}{3} - \frac{v'_C}{3}$$

$$v'_B = -\frac{v'_A}{3} - \frac{v'_B}{3} + \frac{2v'_C}{3}$$

Pour généralise le tableau suivant donne les huit configurations que peut prendre le montage pour l'état fermé (F) ou ouvert (O) des trois interrupteurs K1, K2 et K3 :

Tableau 1 : Configurations des interrupteurs

K1	K2	K3	$v'_A - v'_B$	$v'_B - v'_C$	$v'_C - v'_A$	v'_A	v'_B	v'_C	i'_{K1}	i'_{K2}	i'_{K3}	i
F	F	F	0	0	0	0	0	0	i'_A	i'_B	i'_C	0
F	O	F	U	-U	0	U/3	-2U/3	U/3	i'_A	0	i'_C	$-i'_B$
F	F	O	0	U	-U	U/3	U/3	-2U/3	i'_A	i'_B	0	$-i'_C$
F	O	O	U	0	-U	2U/3	2U/3	-U/3	i'_A	0	0	i'_A
O	F	F	-U	0	U	-2U/3	-2U/3	U/3	0	i'_B	i'_C	$-i'_A$
O	O	F	0	-U	U	-U/3	-U/3	2U/3	0	0	i'_C	i'_C
O	F	O	-U	U	0	-U/3	-U/3	-U/3	0	i'_B	0	i'_B
O	O	O	0	0	0	0	0	0	0	0	0	0

Calcule des valeurs de U et i

En fonction de l'allure de $i = f(wt) pour\ 0 < wt < \pi/3$:

$$i = i'_B - I'_m \sin\left(wt - \frac{2\pi}{3} - \theta\right)$$

Sa valeur moyenne est comme suit :

$$I_{moy} = -\frac{1}{\frac{\pi}{3}} \int_0^{\frac{\pi}{3}} I'_m \sin\left(wt - \frac{2\pi}{3} - \theta\right) dwt = \frac{3\sqrt{2}}{\pi} I' \cos(\theta)$$

On pose : $V'_A = V'_B = V'_C = V'$ (valeur efficaces)

À pertes minimales :

$$P = P'$$

Ce qui implique :

$$UI_{moy} = 3V'I' \cos(\theta)$$

$$U = \frac{\pi}{\sqrt{2}} V'$$

1.6 Driver

Un pilote ou driver est un terme que l'on attribue à un circuit électronique qui permet d'être un intermédiaire physique entre un circuit source. Il est caractérisé par une faible tension entre 5V à 15V et des faibles courants (mA) avec des charges de caractéristique spéciale afin que la source du signal marche correctement sans problème.

Ce driver doit fournir des fonctions par rapport à la charge comme suit :

-Fournir des niveaux de tension et courant appropries, soit numérique ou l'analogique.

-Fournir un changement périodique de la tension et courant qu'elle que soit la charge.

-Ne doit pas être endommagé par des défaillances au niveaux de la charge [11]

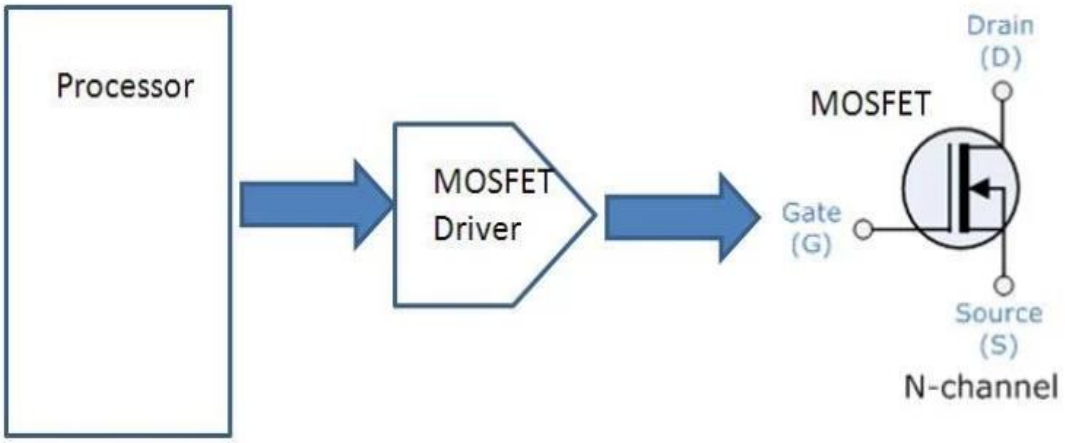


Figure I-15 : Schéma fonctionnel d'un driver

1.7 Temps morts

Générateur de temps mort génère un signal de retard qui commandes les signaux de sortie sans chevauchement, la durée du temps mort stabilise selon l'approche prédictive retenue, le retard appliqué entre les fronts d'ouverture et de fermeture est ajusté en fonction du temps mort observé lors du dernier cycle de commutation. [12]

1.8 Conclusion

Dans ce chapitre, Nous avons discuté de l'ensemble des éléments qui constituent notre variateur de vitesses et principalement la partie puissances, il faut savoir que tout notre processus se repose sur le circuit de commande a base du Microcontrôleur ESP32. Ce dernier doit générer trois Signaux PWM, Pour le fonctionnement du banc d'essais.

Dans le chapitre suivant nous allons décrire en détaille l'implémentation de la loi de commande sur le Microcontrôleur et ainsi les étapes de développement d'une application Android.

Chapitre II

Pilotage du variateur de vitesse par ESP32 à travers une application Android

2.1 Introduction

Dans ce chapitre nous allons expliquer toutes les étapes qui ont une relation avec la commande et la génération des 3 signaux sinusoïdaux, nous allons aussi faire une comparaison entre les différentes technologies disponibles dans le marché afin de choisir la meilleure solution.

Notre prototype est basé principalement sur 2 parties :

- Partie commande qui doit assurer la génération du signal MLI
- Partie puissance qui doit acheminer ce signal vers le moteur avec d'autres modifications qu'on va l'aborder dans le chapitre suivant.

Comme déjà indiqué précédemment nous voulons concevoir un onduleur triphasé et monophasé piloté à travers une interface graphique via Wifi.
Donc il est clair que notre système a besoin de deux corps, un qui va envoyer l'information qui contient les valeurs qui vont former le signal et un autre qui va recevoir cette information.
Pour cette raison nous allons réaliser le premier corps qui va envoyer l'information
Donc nous avons décidé de créer une application mobile Android et une autre application Windows.
Il faut noter que ces 2 applications qui vont s'exécuter sur 2 plateformes différentes vont être bâties à partir d'un seul code source avec **Flutter**. Donc nous allons gagner énormément de temps lors de la programmation.
Après cela nous allons attaquer le deuxième corps. Donc, nous avons décidé après plusieurs recherches et comparaisons de choisir le microcontrôleur "**esp32**".
Les choix de ces technologies seront justifiés dans la section suivante.

Mais avant de passer à la pratique, nous devons d'abord entamer l'étude théorique. Comme nous avons déjà mentionné avant, pour varier la vitesse d'un MAS il faut varier la fréquence et la tension a afin de garder le couple maximal, pour cela il existe deux principales méthodes :

- La commande Scalaire
- La commande Vectorielle

Pour notre cas, nous allons choisir la commande scalaire car elle est simple, robuste, et moins cher comparé à la commande vectorielle car elle ne nécessite pas un calcul lourd, donc des calculateurs puissants.

2.2 Loi de Commande

Notre but est de commander le moteur asynchrone via notre interface graphique donc autrement dit fournir 03 signaux sinusoïdaux au moteur de même amplitude, de même fréquence mais déphasé de 2pi/3 entre eux. Pour un meilleur résultat nous avons opté pour la méthode la plus simple, la commande scalaire.

2.2.1 Commande scalaire

La commande U/f se base sur la mesure de grandeurs scalaires (valeurs d'amplitude en tension et en fréquence).

Afin de garder un flux constant dans le moteur et donc aussi une variation de vitesse à couple constant la tension et la fréquence varient proportionnellement jusqu'à la fréquence nominale du moteur (50 Hz). Lorsque la tension nominale est atteinte, la tension ne doit plus augmenter mais il sera toujours possible d'augmenter la fréquence, Dans ce cas, la variation se fait à puissance constante, par contre le couple a tendance de diminuer avec la vitesse. Ce mode de fonctionnement est intéressant pour des charges à couple constant tels que les ascenseurs. [13]

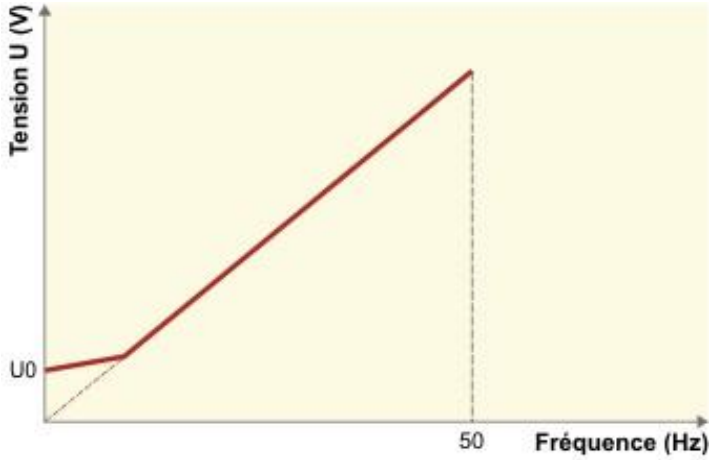


Figure II-16 : Fonctionnement U/f constant.

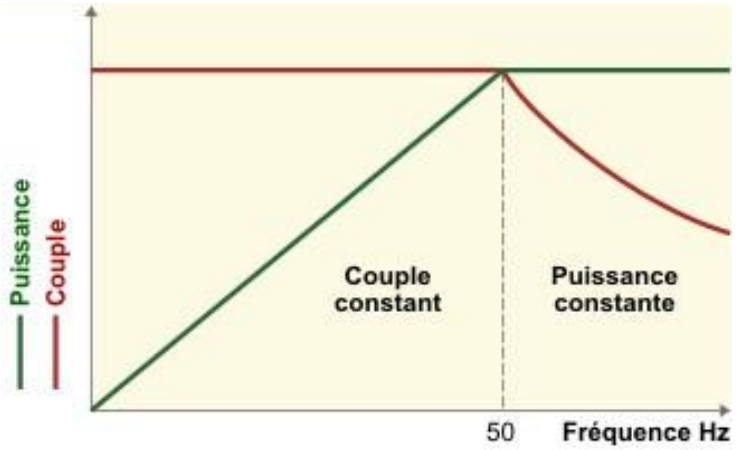


Figure II-17 : Fonctionnement à couple constant sous une fréquence de 50 Hz.

La figure ci-dessous montre les profils des courbes du couple en fonction de la vitesse pour différents rapports U/f

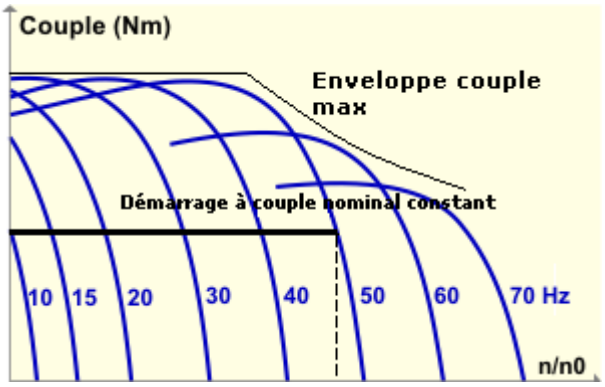


Figure II-18 : Fonctionnement à couple constant.

Avec cette commande, nous pouvons garder le couple constant et maximal grâce à la relation **V/F** durant toute la plage du fonctionnement du moteur jusqu'à la tension maximum.

Donc pour pouvoir appliquer la commande scalaire, il faut que nous construisions un signal sinusoïdal pour chaque fréquence. Nous allons choisir la méthode la plus populaire c'est la modulation de largeur d'impulsion.

2.2.2 Modulation de largeur d'impulsion MLI ou PWM

La modulation de largeur d'impulsion (PWM) est une technique puissante pour contrôler des circuits analogiques avec les sorties numériques d'un microcontrôleur. Le PWM est utilisé dans de nombreuses applications, allant des communications au contrôle et à la conversion de puissance. Par exemple, le PWM est couramment utilisé pour contrôler la vitesse des moteurs électriques, la luminosité des lumières, dans les applications de nettoyage par ultrasons, et bien d'autres encore.

Un PWM est essentiellement un signal numérique unipolaire à onde carrée dont la durée d'activation peut être ajustée (ou modulée) à volonté. De cette façon, la puissance délivrée à la charge peut être contrôlée à partir d'un microcontrôleur. [14]

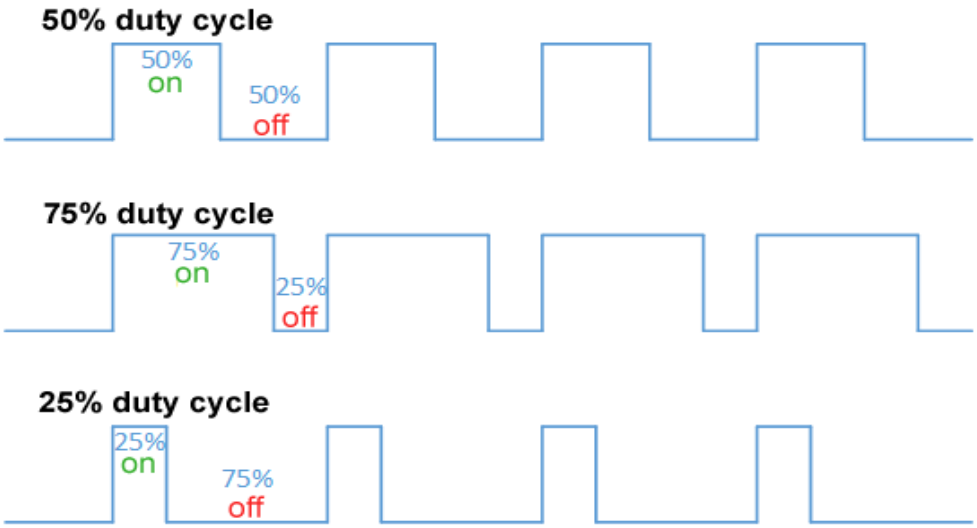


Figure II-19 : Exemple Duty cycle.

Maintenant la question qui se pose c'est comment calculer ces valeurs « Duty cycle » . C'est simple, Nous allons faire ça à l'aide de la transformation de park .

2.2.3 Transformation de Park

Robert H. Park (en) a proposé pour la première fois la transformée éponyme en 1929. En 2000, cet article a été classé comme étant la deuxième publication ayant eu le plus d'influence dans le monde de l'électronique de puissance au XXe siècle . Soit (a,b,c) le repère initial d'un système triphasé, (d,q,o) le repère d'arrivée. À titre d'exemple, la

transformation est réalisée sur un courant, mais nous pouvons l'utiliser pour transformer des tensions et des flux. [15]

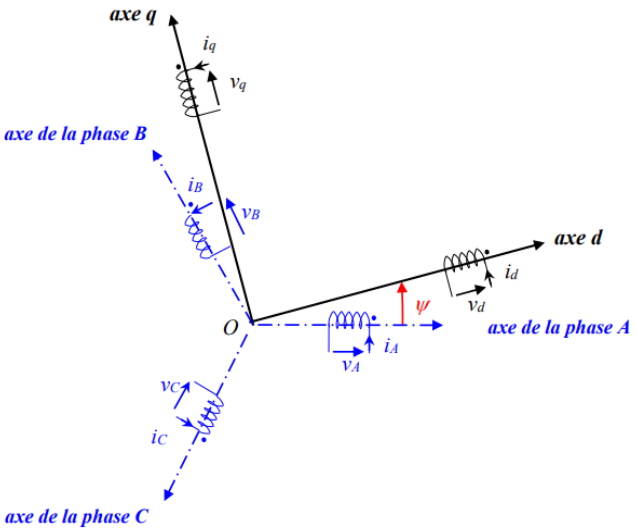


Figure II-20 : Repère abc - dq

La transformation matricielle associée au changement de repère est :

a- Matrice directe :

$$i_{a'b'0} = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\ -\sin(\theta) & -\sin(\theta - \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} . \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}$$

b- Matrice inverse :

$$i_{a'b'0} = \frac{2}{3}\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 1 \\ \cos(\theta - \frac{2\pi}{3}) & -\sin(\theta - \frac{2\pi}{3}) & 1 \\ \cos(\theta \frac{2\pi}{3}) & -\sin(\theta - \frac{2\pi}{3}) & 1 \end{bmatrix} . \begin{bmatrix} i_{a'} \\ i_{b'} \\ i_0 \end{bmatrix}$$

Principe

La transformée dq0 permet dans un système triphasé équilibré de transformer trois quantités alternatives en deux quantités continues. Cela simplifie considérablement la résolution d'équations. Une fois la solution calculée, la transformation inverse est utilisée pour retrouver les grandeurs triphasées correspondantes.

2.3 Création des trois Signaux sinusoïdaux

Pour réussir à faire commander le moteur nous avons besoin de suivre quelque étape :

- Choisir une fréquence PWM
- Déterminer la fréquence fournie au moteur
- Déterminer le pas de calcule sur le cercle
- Calculer V_{alpha} et V_{as} à l'aide de la transformée de park et clark inverse
- Calculer PDCx

Nous allons détailler tous ces étapes dans les sections suivantes :

2.3.1 Choix d'une fréquence PWM f_{PWM}

Pour un résultat parfait nous avons posé la fréquence du porteuse 16000 Hz afin d'éviter tout sorte d'harmonique, cette fréquence est aussi supportable par ESP32.

2.3.2 Détermination de la fréquence "$fré_{Désirée}$" et la tension " $tension_{Désirée}$" fournies

Cette fréquence est imposée par l'utilisateur via l'interface graphique, elle est calculée à partir de la vitesse désirée en utilisant la relation V/F.

On peut aussi déduire la tension désirée.

Notre système oblige l'utilisateur à saisir la configuration du moteur ($tension$, $fré$, $vitesse$, $nombre\ de\ pair\ de\ pôle$) lors de la première utilisation

Sachant que l'utilisateur peut saisir que la vitesse "$vitesse_{Désirée}$" lors de la commande donc :

$$coef = \frac{vitesse_{Désirée}}{vitesse}$$

$$tension_{Désirée} = tension * coef$$

$$fré_{Désirée} = fré * coef$$

2.3.3 Détermination du pas de calcul sur le cercle step

Pour produire le signal sinusoïdal, il faut donc connaître sa valeur à chaque instant, remplaçons le mot instant par “pas” ou “step” en anglais.
Ce terme signifie les positions possibles pour calculer la valeur du signal.
Pour déterminer le nombre de ces points et ces valeurs, nous divisons tout simplement la fréquence f_{pwm} par la fréquence imposée par l'utilisateur **f.**

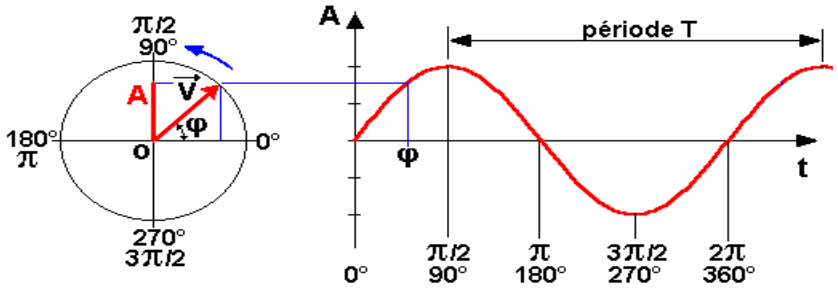


Figure II-21 : Signal sinusoïdale.

$$step = \frac{f_{pwm}}{fré_{Désirée}}$$

2.3.4 Calcul de V_{alpha} et V_{as}

A partir du “pas” ou “step” calculé précédemment nous pouvons déterminer l'angle en degré et la convertir en radian pour calculer son sinus :

$$angle(\theta) = 0$$

$$angle(\theta) = angle(\theta) + step$$

$$V_{alpha} = tension_{désirée} * sin(angle(\theta))$$

$$on\ pose\ \ V_{as} = V_{alpha}$$

2.3.5 Calcul de PDC1

Maintenant nous devons déterminer les rapports cycliques pour qu'on puisse fournir un signal PWM propre, le signal peut être représenté en signe (-), mais les rapports cycliques ne peuvent pas être négatif c'est pour ça nous devons remonter le signal au-dessus du zéro.

$$PDC1 = \frac{Vas + tension_{désirée}}{tension_{désirée} * 2}$$

2.3.6 Déduction de la valeur de PDC2 et PDC3

Afin d'optimiser notre système et minimiser le temps de calcul nous avons décidé de ne pas calculer V_{beta}, V_{bs}, V_{cs} , $PDC2$ et $PDC3$ mais juste les déduire à partir du $PDC1$.

Pour cela nous avons calculé tous les PDC1 et les stocker dans une liste, après nous allons lire les $PDC2$ et $PDC2$ a partir de cette liste.

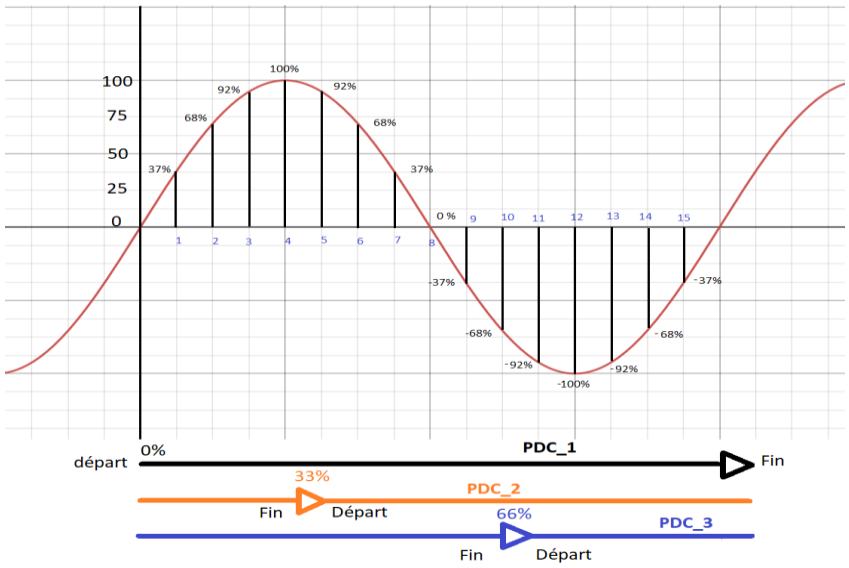


Figure II-22 : Lecture pdc2 et pdc3 à partir du pdc1.

Donc pour produire le deuxième signal nous commençons par 33% de la liste et pour produire le troisième signal nous commençons par 66% de la liste.

2.4 Choix du Microcontrôleur

Il faut savoir que le choix du microcontrôleur est une étape cruciale pour ce genre de projet car nous avons un cahier de charge à respecter et des tâches bien définies à accomplir.

Donc notre choix ne doit pas couvrir le maximum des critères mais il doit assurer tous les Critères et pour déterminer un choix optimal il faut comparer les différents microcontrôleurs disponibles dans le marché.

Nos critères majeurs sont :

- La présence des sortie MLI
- Une fréquence qui assure une commande à chaque 125 microsecondes
- La possibilité d'ajouter un moyen de connectivité pour connecter avec l'interface graphique
- Un prix raisonnable

Au début nous voulons travailler avec un dspic33F, un microcontrôleur que nous le connaissons bien car nous l'avons appris en première année master mais ce microcontrôleur présente 2

Inconvénients majeurs pour notre projet : le premier est que sa communauté est très petite Environ 80000 résultats sur Google.

Autrement dit si nous avons des questions ou nous souhaitons avoir de l'aide on recevra probablement aucune réponse.

le deuxième Inconvénient est l'absence d'un moyen de communication native wifi ou Bluetooth <built-in> c'est à dire un module qui est pris en charge de la part de la carte nativement sans rien ajouter. Ce point est très important pour la rapidité du système.

Pour régler ces problèmes, nous devons trouver d'autre solution, nous avons pu régler le premier problème avec un Arduino Due qui a la même fréquence que le dspic33F (80 Mhz). Et qui a une communauté très grande par rapport au dspic environ 37 Millions résultats sur Google mais le deuxième problème est toujours présent pas de moyen de connectivité native en plus cette carte est très chère par rapport au dspic environ 40$ contre seulement 11$ pour le dspic.

Notre dernier candidat c'est l'esp 32 une carte avec une fréquence qui peut atteindre jusqu'à 240 Mhz, sa communauté est petite par rapport à l'Arduino Due environ 11 Millions sur Google mais le point fort de cette carte est le module wifi et Bluetooth qui est pris en charge nativement par la carte en plus de ces avantages il n'est pas cher environ 11$.

C'est vrai que cette carte a l'air de satisfaire nos besoins mais pour prendre une décision juste et correcte entre ces trois cartes il faut comparer d'autres aspects.

C'est pour ça nous avons établi un tableau bien détaillé des différents aspects de ces cartes

Pour déterminer laquelle est la meilleure pour notre projet.

La couleur verte veut dire : _ choix optimal _

Tableau 2 : Comparaisons entre les différents microcontrôleurs

	Dspic33F	**Arduino Due**	**Esp Wroom 32**
Fréquence	80 Mhz	84 Mhz	240 Mhz
Ram	30 KB	96 KB	520 KB

	Dspic33F	Arduino Due	Esp Wroom 32
Flash Memory	256 KB	512 KB	4 MB
Tension d'alimentation	3.0 V - 3.6 V	7 V -12 V	3.3 V - 5 V
Architecture	16 bit	32 bit	32 bit
PWM Résolution	16 bit	16 bit	16 bit
PWM Output	8 canaux	12 canaux (2 -13)	16 canaux * 19 pin
ADC Résolution	12bit	12 bit	12 bit
ADC Input	24 canaux	12 canaux	16 canaux
Température Max	85°C	70°C	85°C
Température Min	-40°C	- 25°C	-40°C
Connectivité WIFI	Non	Non	Oui
Connectivité Bluetooth	Non	Non	Oui
SPI	2	1	4
I2C	2	1	2
I2S	1	2	2
UART	2	4	3
Timer	9	9	16
Langage de programmation	• C et C++	• Sous-ensemble De C et C++	• Sous-ensemble de C et C++ • Python
Broche	85 PIO	54 PIO	38 PIN
taille	12mm * 12mm	101.6mm * 53.3mm	54.6mm * 27.94mm

	Dspic33F	Arduino Due	Esp Wroom 32
prix	11.19 $	40.30 $	10.50 $
Besoin de kit débogueur ?	Oui	Non	Non
Communauté sur Internet	78 900 Résultats	37 200 000 Résultats	11 700 000 Résultats

Maintenant après cette petite comparaison il est clair que l'esp 32 est très performant que les autres cartes donc pour le reste de ce projet nous allons choisir cette carte pour réaliser ce projet.

2.5 Présentation de Esp 32

L'ESP 32 est une série de microcontrôleurs à faible coût et faible consommation, avec Wi-Fi et Bluetooth en mode intégrés. La série ESP32 utilise soit un microprocesseur Tensilica Xtensa LX6 en version double cœur et simple cœur, soit un microprocesseur Xtensa LX7 double cœur, soit un microprocesseur RISC-V simple cœur, et comprend des commutateurs d'antenne intégrés, un amplificateur de puissance, un amplificateur de réception à faible bruit, des filtres et des modules de gestion de l'alimentation.
L'ESP32 est créé et développé par Espressif Systems, une société chinoise basée à Shanghai, et il est fabriqué par l'entreprise taïwanaise TSMC. [16]

2.5.1 Caractéristique

Tableau 3 : Caractéristique de ESP32

Nombre de cœurs	2 (dual core)

Wi-Fi	2.4 GHz up to 150 Mbits/s
Bluetooth	BLE (Bluetooth Low Energy) and legacy Bluetooth
Architecture	32 bits
Fréquence	240 MHz
RAM	512 KB
Pins	30 or 36 (ça dépend le model)
Périphériques	Capacitive touch, ADC (analog to digital converter), DAC (digital to analog converter), I2C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

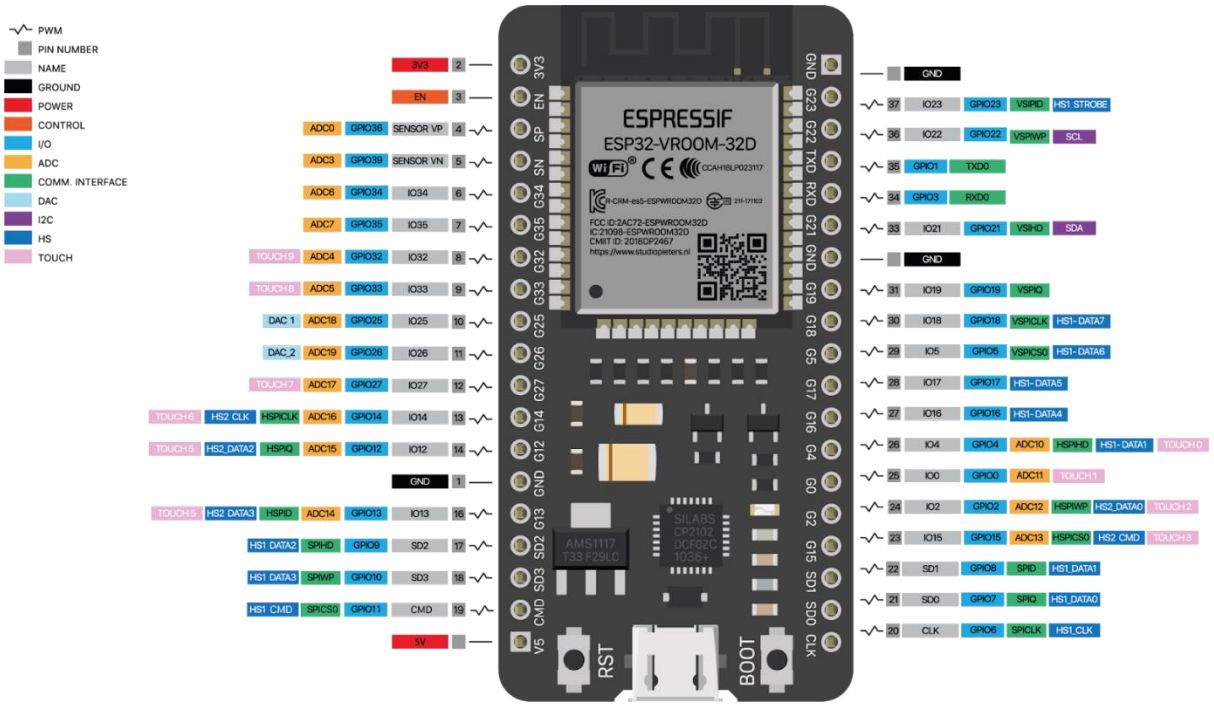


Figure II-23 : ESP-WROOM-32 38 PIN Development Board.

2.5.2 Avantage de L'esp 32

Après cette présentation, nous pouvons constater que cette carte a plusieurs avantages par rapport aux autres cartes au même prix. Parmi ces avantages il y a la fréquence qui peut atteindre jusqu'à 240 Mhz et aussi la Ram avec 512 KB.
Un autre point fort de cette carte est la mémoire qui varie entre 4 Mo et 16 Mo selon le modèle et qui est répartie entre (EEPROM - SPIFFS - PREFERENCE). [17]

- **Conception robuste**
  L'ESP32 est capable de fonctionner de manière fiable dans des environnements industriels, avec une température de fonctionnement allant de -40°C à +125°C. Grâce à des circuits d'étalonnage avancés, l'ESP32 peut éliminer dynamiquement les imperfections des circuits externes et s'adapter aux changements des conditions extérieures.

- **Consommation d'énergie ultra-faible**
  Conçu pour les appareils mobiles, l'électronique portable et les applications IoT, l'ESP32 atteint une consommation d'énergie ultra-faible grâce à l'association de plusieurs types de logiciels propriétaires. L'ESP32 comprend également des fonctionnalités de pointe, telles que le gating d'horloge à grain fin, divers modes d'alimentation et la mise à l'échelle dynamique de la consommation.

- **Haut niveau d'intégration**
  L'ESP32 est hautement intégré avec des commutateurs d'antenne, un balun RF, un amplificateur de puissance, un amplificateur de réception à faible bruit, des filtres et des modules de gestion de l'alimentation intégrés. L'ESP32 ajoute une fonctionnalité et une polyvalence inestimables à vos applications avec des exigences minimales en matière de cartes de circuits imprimés (PCB).

- **Puce hydride Wi-Fi et Bluetooth**
  L'ESP32 peut fonctionner comme un système autonome complet ou comme un dispositif esclave d'un MCU hôte, ce qui réduit la surcharge de la pile de communication sur le processeur d'application principal. L'ESP32 peut

s'interfacer avec d'autres systèmes pour fournir des fonctionnalités Wi-Fi et Bluetooth via ses interfaces SPI / SDIO ou I2C / UART.

2.5.3 Inconvénient de l'esp 32

Les broches ADC2 ne peuvent pas être utilisées lorsque le Wi-Fi est utilisé.
Les canaux d'entrée ADC ont une résolution de 12 bits. Cela signifie que vous pouvez obtenir des lectures analogiques allant de 0 à 4095, dans lesquelles 0 correspond à 0V et 4095 à 3,3V. Vous avez également la possibilité de définir la résolution de vos canaux sur le code, ainsi que la plage ADC.
Mais les broches ADC de l'ESP 32 n'ont pas un comportement linéaire. Vous ne pourrez probablement pas faire la différence entre 0 et 0,1V, ou entre 3,2 et 3,3V.

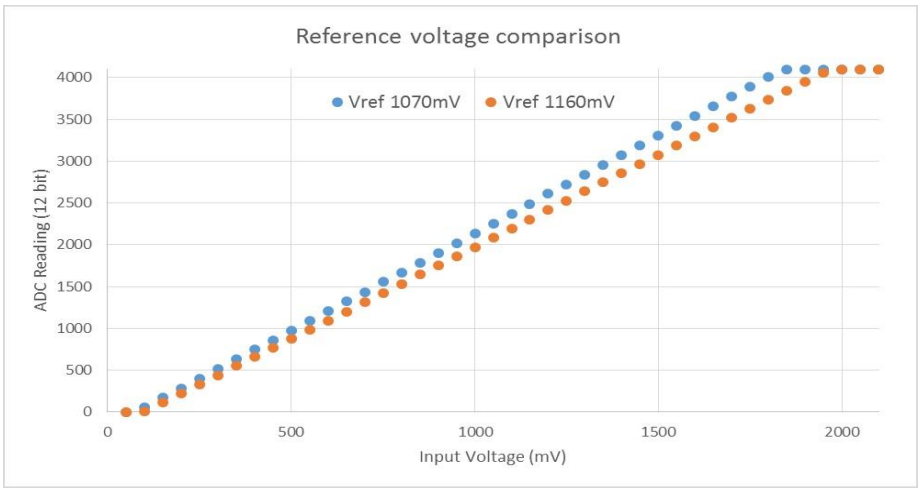


Figure II-24 : Référence de conversion ADC.

2.6 Présentation Flutter

Flutter est un kit de développement logiciel d'interface utilisateur "UI" open-source créé par Google. Il est utilisé pour développer des applications multiplateformes pour Android, iOS, Linux, Mac OS, Windows, Google Fuchsia et le web à partir d'une base de code unique.

Contrairement à d'autres solutions populaires, Flutter n'est pas un Framework ou une bibliothèque, c'est un SDK (kit de développement logiciel) complet. Clarifions brièvement les choses ici pour nous assurer que nous sommes sur la même longueur d'onde [18]

Une bibliothèque est essentiellement un élément de code réutilisable que vous placez dans votre application pour exécuter une certaine fonction commune.

Un Framework est une structure qui vous fournit un squelette d'architecture pour construire un logiciel. Il s'agit d'un ensemble d'outils qui sert de base à votre application, vous devez remplir les blancs avec votre code pour compléter la structure entière et obtenir la fonctionnalité souhaitée.

Une SDK a une portée beaucoup plus large puisqu'il s'agit d'une collection d'outils, comprenant des bibliothèques, de la documentation, des API, parfois des frameworks, et plus encore, vous donnant tout ce dont vous avez besoin pour le développement de logiciels. Et c'est le cas avec Flutter - il contient déjà tout ce qui est nécessaire pour construire des applications multiplateformes.

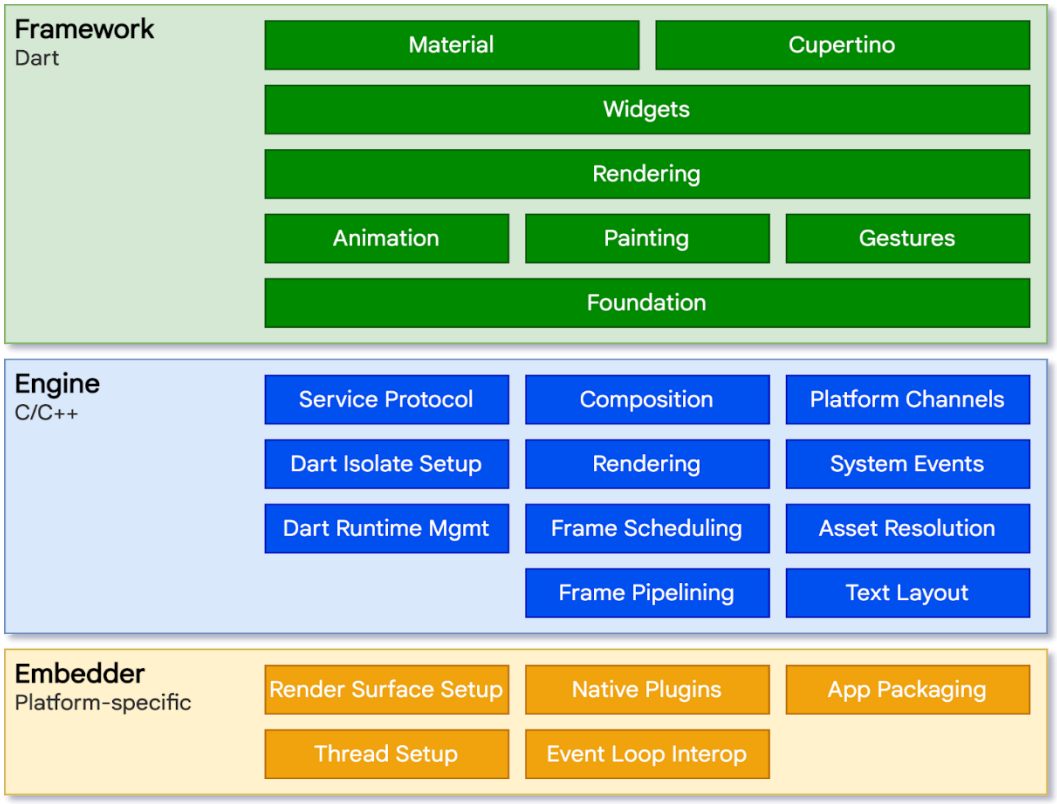


Figure II-25 : Architecture du Flutter.

Les trois principales couches architecturales sont

1. Un embarder qui utilise un langage spécifique à la plateforme et fait fonctionner l'application sur n'importe quel système d'exploitation

2. Un moteur écrit en C/C++ qui fournit une mise en œuvre de bas niveau des API de base de Flutter. Cela inclut les graphiques (par le biais de la bibliothèque graphique 2D Skia), la mise en page du texte, les entrées/sorties de fichiers et de réseaux, la prise en charge de l'accessibilité, l'architecture des plugins et une chaîne d'outils d'exécution et de compilation Dart

3. Un Framework basé sur le langage de programmation Dart. Sa mise en œuvre est facultative, mais il fournit un riche ensemble de bibliothèques qui peuvent être divisées en couches : classes de base fondamentales, couche de rendu, couche de widgets et bibliothèques Matériel/Cupertino.

2.6.1 Pourquoi Choisir Flutter

- **Open Source**

  Flutter est une boîte à outils de développement logiciel open source de Google. Il permet de poster facilement des problèmes et d'accéder à la documentation des forums de développeurs ouverts. Il aide les programmeurs Flutter à apprendre et à évoluer avec la communauté en constante évolution des développeurs qui contribuent activement à la plateforme. Elle augmente l'efficacité et la productivité du codeur et permet de réduire le temps et le coût de l'ensemble du projet. [19]

- **Coder une seule fois**

  Comme il s'agit d'un Framework multiplateforme, il permet aux programmeurs d'écrire le code une seule fois et de l'utiliser sur plusieurs plateformes. Cela signifie qu'une seule version d'une application fonctionne à la fois sur Android, iOS, MacOs, Windows, Linux, and Web. Cela permet d'économiser beaucoup de temps et d'efforts dans l'écriture du code pour différentes plateformes, comme c'est le cas avec les Framework

natifs. Cela permet donc de réduire considérablement le coût global du développement et du lancement de l'application. [19]

- **Dart comme langage de programmation**

  Flutter utilise Dart comme langage de programmation orienté objet pour créer des applications. Les principales caractéristiques de Dart comprennent une bibliothèque standard riche, le garbage collection, le typage fort, les génériques et les async-awaits. Dart ressemble à Java et utilise également un grand nombre de caractéristiques populaires d'autres langages. Son style de programmation réactif permet aux développeurs d'accomplir leurs tâches courantes avec facilité. [19]

- **Rechargement à chaud et développement**

  Il s'agit d'une fonctionnalité unique de Flutter, qui permet aux développeurs de voir instantanément les modifications apportées au code. Toutes les mises à jour sont disponibles pour les concepteurs et les développeurs en quelques secondes. Ils n'ont pas besoin d'attendre les mises à jour et peuvent continuer à utiliser le framework pour développer d'autres fonctionnalités sans être interrompus. Cela augmente la productivité du développeur et réduit le temps nécessaire à la création d'une application robuste. Cela permet d'économiser beaucoup d'argent dans le développement global du projet. [19]

- **Des performances dignes d'une application native**

  Lorsque vous faites appel à des développeurs d'applications mobiles pour créer des applications sur Flutter, vous obtenez une exécution rapide sur n'importe quelle plateforme. Cela est dû au fait qu'il utilise le langage de programmation Dart, qui est rapide, simple et peut facilement être compilé en code natif. Il améliore les performances de l'application par rapport à toute autre plateforme de développement d'applications. [19]

- **La communauté technologique**

  Il existe une puissante communauté de développeurs qui travaillent en permanence pour contribuer à l'amélioration de Flutter. Ils facilitent l'accès et l'apprentissage du Framework pour les nouveaux arrivants. Il existe plus de 50 vidéos qui peuvent aider à construire avec la boîte à outils logicielle. Tout le monde peut commencer à développer une application sans effort avec l'aide de l'équipe. [19]

- **Utilisation de widgets personnalisés**
  Utilisation de widgets personnalisés Flutter propose une myriade de widgets pour aider les développeurs dans leur processus de création. Il rend la conception d'une interface utilisateur de base beaucoup plus facile et rapide. Il vous suffit de créer un élément d'interface utilisateur une fois pour qu'il s'adapte à différentes résolutions, d'écrans et plateformes. Vous pouvez même envelopper un widget dans un autre pour activer différentes fonctions. [19]

2.7 Présentation de l'environnement de travail Vscode

Visual Studio Code est un éditeur de code source conçu par Microsoft léger mais puissant qui s'exécute sur votre bureau et il est disponible sur les systèmes d'exploitation Windows, MacOs et Linux. Il est livré avec une prise en charge intégrée de JavaScript, TypeScript et Node.js et dispose d'un riche écosystème d'extensions pour d'autres langages (tels que C++, C#, Java, Python, PHP, Go) et environnements d'exécution (tels que .NET et Unity).

Un éditeur de code source peut sembler banal pour certaines personnes, mais il est le cœur de tant de sociétés de logiciels dans le monde.

Bien qu'il existe de nombreux éditeurs, le plus approprié tend à accélérer la productivité du développeur en fournissant différents plugins et composants qui prennent la responsabilité de choses comme la coloration syntaxique, le diff, les macros, les extraits de code, les options de prévisualisation, le débogage, la compilation et le déploiement de tout programme logiciel [20]

2.8 Code Flutter

Nous allons maintenant expliquer quelque partie de notre code Flutter, l'architecture, la base de données et aussi les écrans.
Avec notre Application "**Async Motor**" nous voulons commander notre moteur avec la moindre clique possible, tout ce que vous devez faire c'est enregistrer les caractéristiques de votre moteur une seule fois et après vous pouvez le commander directement en remplissant les deux champs vitesse désiré en Tr/min et le glissement.

2.8.1 Implémentation de notre architecture

Pour la facilité de lire et comprendre notre projet ainsi pour la maintenance nous allons adopter une structure spécifique nous allons créer 3 dossiers

- **View** : ce dossier contient des fichiers qui vont construire notre vue ou écrans
- **Fonctions** : par contre ce dossier contient tous fonctions et les calculs nécessaire
- **Model** : ce dossier contient seulement notre modèle de base de données il ne contient pas les données qui seront stocké ultérieurement dans la base de données

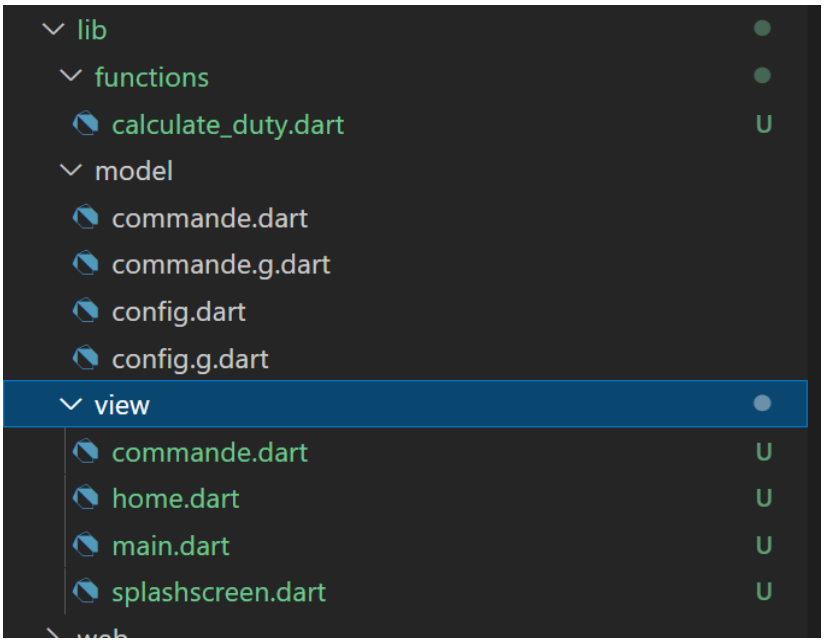


Figure II-26 : Logique de l'application.

2.8.2 Explication de la fonction DUTY

Paramètre de la fonction

```
String duty(

int vitesseDesire,
int vitesse,
int frequence,
int tension,
double glissemnt) {}
```

Cette fonction **duty** accepte 5 paramètres, my speed et glissement ce sont les valeurs reçues de la part d'utilisateurs, vitesse, fréquence, tension, sont les valeurs déjà stockées lors de la configuration.

Détermination de la tension et fréquence Désiré

```
var tensionDesire;
var mySpeed = vitesseDesire + (vitesseDesire * (glissemnt * 0.01));

var coef = mySpeed / vitesse;

if (coef >= 1) {
  tensionDesire = tension;
}

if (coef < 1) {
  tensionDesire = tension * coef;
}

var frequenceDesire = frequence * coef;
```

Dans cette partie, nous calculons la vitesse désirée avec le glissement après nous déduisons le coefficient pour déterminer la valeur de la nouvelle tension et fréquence.

Les deux conditions ici c'est pour ne pas faire des calcule avec une tension plus élevée que la tension du moteur

Initialisation des variables

```
String pdc;
String transPdc = "";

double V_as = 0;

double Pdc_1 = 0;

int Pdc_1_To_num = 0;

int counter = 0;

var double_tension = tensionDesire * 2;
int frequence_PWM = 16000;
double degre = 0;
```

```
double step = (360 * frequenceDesire) / frequence_PWM;
```

Après nous initialisons tous les variables qu'on aura besoin pour faire nos calcules
Fréquence du porteuse 16000 Hz.
La variable double tension c'est pour générer le signal qui sera comprise entre 0 et 1 et le point 0.5 représente le passage par 0.

Calcule de Duty Cycle

```
for (degre = 0; degre < 360; degre = degre + step) {
  counter++;

  double rad_step = degre * (math.pi / 180);
  double sintheta = math.sin(rad_step);

  var vAlpha = tensionDesire * sintheta;

  V_as = vAlpha;

  Pdc_1 = (V_as + tensionDesire) / double_tension;

  Pdc_1_To_num = (Pdc_1 * 1024).round();
```

Tout d'abord nous devons convertir le pas de degré en radian car la fonction sinus n'accepte que des valeurs en radian après nous calculons le sinus Thêta, ensuite le V_{alpha}et nous calculons la valeur PDC sous forme de pourcentage, finalement il faut convertir cette valeur en numérique car la fonction PWM dans esp32 n'accepte que des valeurs numériques.
Pour notre cas nous avons calculé seulement V_{alpha}et V_{as} pour calculer Pdc_1
On va juste déduire Pdc_2 , Pdc_3 dans esp32.
Cette méthode nous a fait gagner beaucoup de temps non seulement de calcule mais de transfert aussi

Préparation de la liste des valeurs PDC pour l'envoi :

```
    if (Pdc_1_To_num < 10) {
      transPdc = "000" + Pdc_1_To_num.toString();
    }
    if (Pdc_1_To_num >= 10 && Pdc_1_To_num < 100) {
      transPdc = "00" + Pdc_1_To_num.toString();
    }
    if (Pdc_1_To_num >= 100 && Pdc_1_To_num < 1000) {
      transPdc = "0" + Pdc_1_To_num.toString();
    }
    if (Pdc_1_To_num >= 1000) {
      transPdc = Pdc_1_To_num.toString();
    }

    List_PDC_1.add(transPdc);
  }
```

il faut noter la que le seul type accepté par la fonction web Socket dans esp32 qui est responsable de la communication entre esp32 et les autre appareil c'est le type **String** donc nous sommes obligé de convertir notre résultat en **String** et après la réception esp32 doit la convertit encore une fois en **Int**

C'est pour cette raison nous rassemblons tous les valeurs dans une chaine de caractère

Mais là nous sommes face à un problème lors de la réception car les valeurs peuvent être constitué d'un chiffre 2 ,3 ou 4 chiffres par exemple (512,1024,0,107), esp32 ne peut pas détecter cette information dans une grande chaine de caractère

C'est pour cela on doit implémenter une seule logique de transmission et réception

Donc nous avons opté pour une solution simple forcer toutes les valeurs d'être constitué de 4 caractères avec les 4 conditions **"if",** Donc esp32 n'a qu'à lire cette chaîne de caractère après la réception 4 par 4 et convertir le résultat en **Int** et automatiquement les zéros à gauche de la valeur seront supprimés.

Finalement la fonction **Duty** va retourner une valeur **String** prête pour être envoyer à ep32 .

2.9 Présentation de l'application développée

Notre application **"Async Motor"** est constituée de trois écrans essentiellement, Un écran pour accueillir l'utilisateur et un autre pour insérer et visualiser les différentes configurations des moteurs et le dernier écran permet aux utilisateurs d'envoyer des commandes au moteur.

2.9.1 Écran d'accueil

Après l'ouverture de l'application l'écran Home sera afficher en premier Depuis cet écran vous pouvez ajouter une nouvelle configuration, ou voir vos configurations déjà ajoutées, Et pour simplifier la recherche nous avons ajouté un filtre étoile / triangle. Finalement Pour commander le moteur il suffit de cliquer sur la configuration et vous allez être amené vers l'écran "commande".

Figure II-27 : Ecran d'accueil.

2.9.2 Ajouter une configuration

Pour ajouter une configuration, c'est simple, il faut juste remplir les 5 champs :
Le nom du moteur c'est unique vous ne pouvez pas nommer 2 moteurs avec un seul nom
Le glissement du moteur sera envoyé lors de la commande à chaque fois car il peut être changer selon la charge.

Figure II-28 : Ajouter une configuration.

2.9.3 Écran de Commande

Depuis cet écran vous pouvez savoir d'abord si esp32 est connecté ou non

Sinon on peur actualiser la connexion avec le bouton noir.

Après que vous envoyez une commande, des informations seront affichés sur l'écran ainsi qu'un graphe qui montre les trois signaux.

Figure II-29 : Ecran "commande 01".

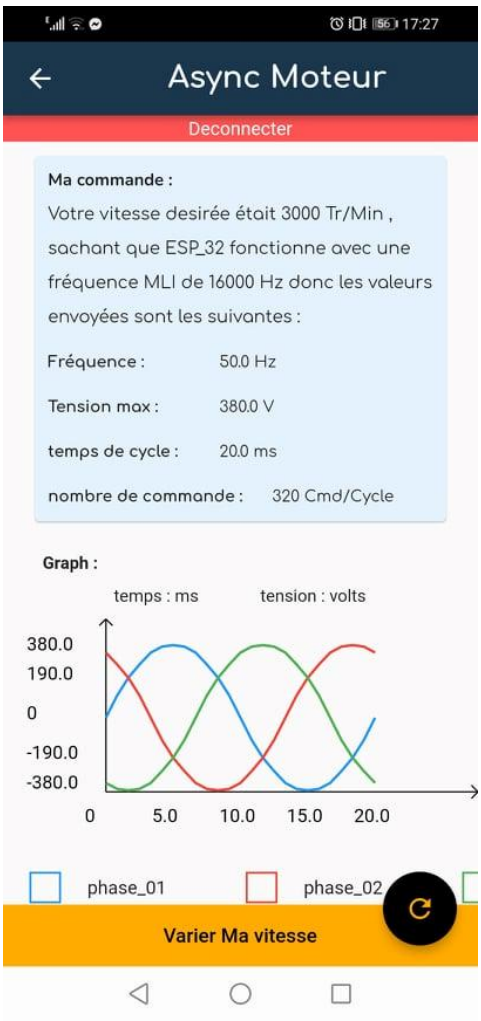


Figure II-30 : Ecran "commande 02".

2.9.4 Envoi d'une commande

Pour envoyer une commande assurez-vous que vous êtes connecté à esp32 via wifi après vous entrez votre vitesse désirée et le glissement du moteur selon les tests réalisés précédemment et cliquer sur envoyer.

Figure II-31 : Envoyer une commande.

Les informations affichées sur cet écran seront changées immédiatement et à chaque fois que vous envoyez une commande.

2.10 Code esp32

Le Rôle du code esp32 c'est de recevoir une chaine de caractère qui contient les valeurs PDC1 ,ensuite la première opération est de savoir la longueur de cette chaîne et la diviser par 4 car comme nous avons déjà expliqué chaque valeur est constitué de 4 caractère ,le résultat obtenu c'est le nombre total des commandes la deuxième opération c'est de stocker cette chaine de caractère dans un fichier PDC.txt à l'intérieur de esp32 grâce à la bibliothèque SPIFFS donc le prochaine démarrage de esp32 sera avec la dernière commande envoyée .

Nous commençons par l'initialisation de tous les variables et les bibliothèques qu'on aura besoin pour produire nos signaux.

```
#include <FS.h>
#include <WebSocketsServer.h>
#include <Preferences.h>
#include "SPIFFS.h"

int x;
int y;

int pdc_2 ;
int pdc_3 ;
unsigned long CurrentTime = 0 ;
unsigned long ElapsedTime = 0;
unsigned long StartTime = 0;

const int outputPWM_03 = 23;
const int outputPWM_02 = 26;
const int outputPWM_01 = 14;

const int freq = 16000;
const int resolution = 10;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int ledChannel_3 = 3;

const char *ssid =  "ESP32";
const char *pass =  "00000000";

File file_writing;
File file_reading;
String MyDuty;

int pdc1_from_spiffs[1400];
```

```
int counter = 0;
String pdc = "";
int LenPDC ;
```

Ensuite nous déclarons la fonction qui est chargé de stocker notre chaine de caractère dans la mémoire de esp32 **readingFromFile ()**

- On ouvre le fichier en mode lecture après on stock la valeur dans une variable Après on ferme le fichier
- On détermine la longueur de la chaîne après on déduit le départ des signaux **pdc2** et **pdc3**
- Finalement on sépare les valeurs de la chaîne de caractère et les stocker dans une liste **"pdc1_from_spiffs"**

```
void readingFromFile(){
    SPIFFS.begin();
    file_reading = SPIFFS.open("/pdc.txt","r");
    String My_pdc  = file_reading.readString();
    file_reading.close();
    int lenOfPdc =  My_pdc.length();
    counter = lenOfPdc/4;
    pdc_2 = round(counter*0.333);
    pdc_3 = round(counter*0.666);

    int j=0;
     for (size_t i = 0; i < lenOfPdc; i=i+4)
     {
        j++;
        MyDuty = My_pdc.substring(i,i+4);
        pdc1_from_spiffs[j] = MyDuty.toInt();
     }
}
```

Apres nous avons configurer la fonction chargée de recevoir la chaine de caractère envoyé Dans le cas où l'information est disponible la fonction web_Socket lis cette chaîne et la stocker dans le fichier "**pdc.txt"** après on fait appel à la fonction **readingFromFile ()** pour séparer les valeurs et mettre à jour la liste **"pdc1_from_spiffs.**
Finalement la fonction envoie un message à l'application montrant que la tâche à bien été exécuté.

```
void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
//webscket event method

    switch(type) {
        case WStype_DISCONNECTED:
            Serial.println("Websocket is disconnected");
            break;
        case WStype_CONNECTED:{
            Serial.println("Websocket is connected");
            Serial.println(webSocket.remoteIP(num).toString());
            webSocket.sendTXT(num, "hello");
        }
            break;
        case WStype_TEXT:
            pdc = "";
            for(int i = 0; i < length; i++) {
                pdc = pdc + (char) payload[i];
            }
            Serial.println(pdc);
            LenPDC  = pdc.length();
            counter = LenPDC/4;
            Serial.println("counter");
            Serial.println(counter);
            file_writing =  SPIFFS.open("/pdc.txt","w");
            file_writing.print(pdc);
            file_writing.close();
            readingFromFile();
            webSocket.sendTXT(num, "esp32 : executed after spiffs writing");

            break;
        case WStype_FRAGMENT_TEXT_START:
            break;
        case WStype_FRAGMENT_BIN_START:
            break;
        default:
            break;
    }
}
```

La fonction setup, nous avons une configuration générale

- Initialiser le serial et le spiffs.
- Appeler la fonction **readingFromFile**
- Configurer la fonction PWM les canaux, les fréquences, et la résolution
- Attacher cette configuration a une des Output
- Lancer le wifi et le web Socket

```
void setup() {

Serial.begin(115200);
SPIFFS.begin();

readingFromFile();

Serial.println("nombre de commande");
Serial.println(counter);

ledcSetup(ledChannel_1, freq, resolution);
ledcSetup(ledChannel_2, freq, resolution);
ledcSetup(ledChannel_3, freq, resolution);

ledcAttachPin(outputPWM_01, ledChannel_1);
ledcAttachPin(outputPWM_02, ledChannel_2);
ledcAttachPin(outputPWM_03, ledChannel_3);

Serial.println("Connecting to wifi");

IPAddress apIP(192, 168, 0, 1);
WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
WiFi.softAP(ssid, pass);

webSocket.begin();
webSocket.onEvent(webSocketEvent);
Serial.println("Websocket is started");

}
```

Dans La fonction Loop nous forçons web Socket à tourner on Loop pour les informations venant de la commande à n'importe quel moment

Après dans la boucle for on envoie notre signal via le pin déjà défini et à partir d'une seule liste de pdc1_from_spiffs

Finalement nous forçons le système à attendre 44 microsecondes car chaque opération de **PWM** prend environ 6 microsecondes donc :

$$6\,microsecondes * 3 = 18$$

$$On\ sachant\ que$$

$$1\,s\,/\,16000\,Hz = 62.5\,microsecondes$$

$$Donc$$

$$62.5 - 18 = 44.5\ microsecondes$$

C’est le temps qu'il faut attendre pour envoyer les valeurs suivantes.
Et pour être sûr que notre signal respecte la fréquence designer on affiche le temps en milliseconde à chaque fin de cycle.

```
void loop() {

    webSocket.loop();
    StartTime = millis();
    x = pdc_2;
    y = pdc_3;

   for (size_t i = 1; i < counter; i++)
   {
        x++;
        y++;
        ledcWrite(ledChannel_1,pdc1_from_spiffs[i]);
        ledcWrite(ledChannel_2,pdc1_from_spiffs[x]);
        ledcWrite(ledChannel_3,pdc1_from_spiffs[y]);
        if(x == counter ){
            x = 0;
1.  }
        if(y == counter ){
            y = 0;
        }


 delayMicroseconds(44);
  }

    CurrentTime = millis();
    ElapsedTime = CurrentTime - StartTime;
    Serial.print("Cycle's Time :  ");
    Serial.println(ElapsedTime);

}
```

2.10.1 Pourquoi utilise-t-on SPIFFS

Serial Peripheral Interface Flash File System, ou SPIFFS pour faire court. Il s'agit d'un système de fichiers léger pour les microcontrôleurs dotés d'une puce flash SPI. La puce flash embarquée de l'ESP8266 dispose de beaucoup d'espace pour vos fichiers. La version standard (dev kit) de ESP32 dispose de 4 Mo de flash et vous pouvez en utiliser 3Mo pour vos fichiers. SPIFFS vous permet d'accéder à la mémoire flash comme s'il s'agissait d'un système de fichiers normal comme celui de votre ordinateur (mais beaucoup plus simple, bien sûr)
Vous pouvez lire et écrire des fichiers, créer des dossiers, etc. [21]
La limite de cette mémoire est qu'elle n'a que 100000 cycles d'écriture.
L'utilisation de SPIFFS avec la carte ESP32 est particulièrement utile pour :

- Créer des fichiers de configuration avec des paramètres
- Sauvegarder les données de façon permanente
- Créer des fichiers pour sauvegarder de petites quantités de données au lieu d'utiliser une carte microSD
- Enregistrer des fichiers HTML et CSS pour construire un serveur web
- Sauvegarder des images, des figures et des icônes ;
  Et bien plus encore

2.11 Conclusion

Dans ce chapitre nous avons expliqué en détails les différentes étapes nécessaires pour produire les trois signaux sinusoïdaux pour faire tourner le moteur selon la vitesse désignée par l'utilisateur. Nous avons vu aussi comment nous avons procédé pour choisir les meilleures technologies que ce soit pour le microcontrôleur ou pour l'application mobile.
Esp32 a montré de bonnes performances lors de nos essais notamment avec un wifi qui peut atteindre jusqu'à 15 mètres avec des murs et une vitesse allant jusqu'à 150 Mbits/s.
L'application mobile elle aussi a montré un comportement natif, Une rapidité remarquable et une transition fluide entre les écrans.
Le seul problème rencontré lors de ce projet et que nous n'avons pas réussir à acheter des versions originales de l'esp32, donc nous avons travaillé juste avec une version cloner qui ne permet pas l'alimentation de l'esp32 via le pin 5V et 3.3V mais seulement avec un câble.

Chapitre III

Réalisation de notre prototype

3.1 Introduction

Après avoir présenté et discuté les différentes structures d'un variateur de vitesse, ainsi que le principe de fonctionnement de notre système dans les deux chapitres précédents, nous sommes arrivés à l'étape cruciale de rassembler tous les aspects de l'idée d'un variateur de vitesse. C'est l'objectif principal du chapitre. Ainsi, nous allons regrouper tous les composants majeurs du banc de test, y compris les circuits d'alimentation, de contrôle et de pilotage, afin de créer une architecture complète pour notre produit, comme le montre le schéma ci-dessous.

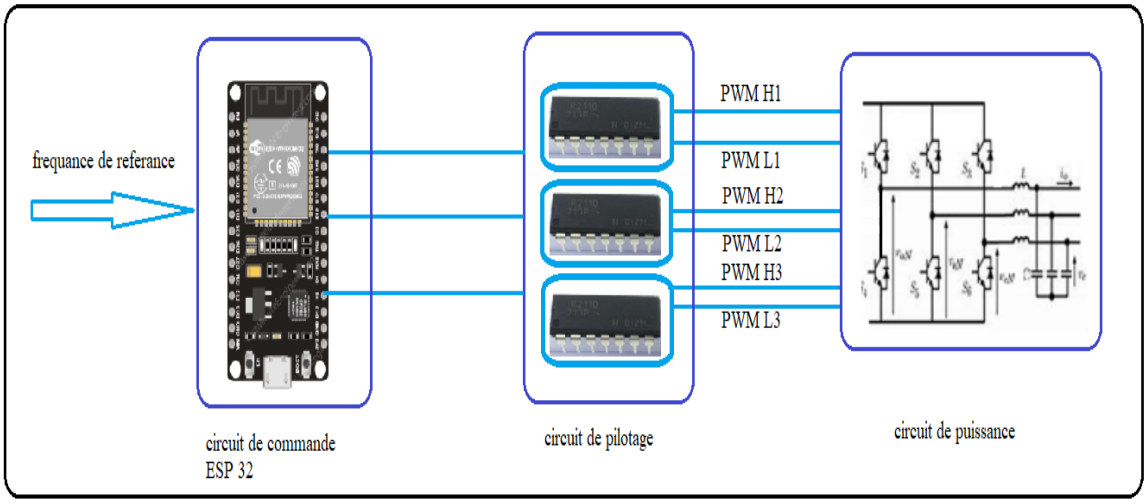


Figure III-32 : Structure matérielle d'un variateur de vitesse.

3.2 Présentation du banc d'essai

3.2.1 Circuit de puissance

Notre circuit de puissance repose sur des transistors IGBT IRG4P H20KD imposés par le cahier des charges, avec une tension max supportée de 1200 V et un courant de 11A.

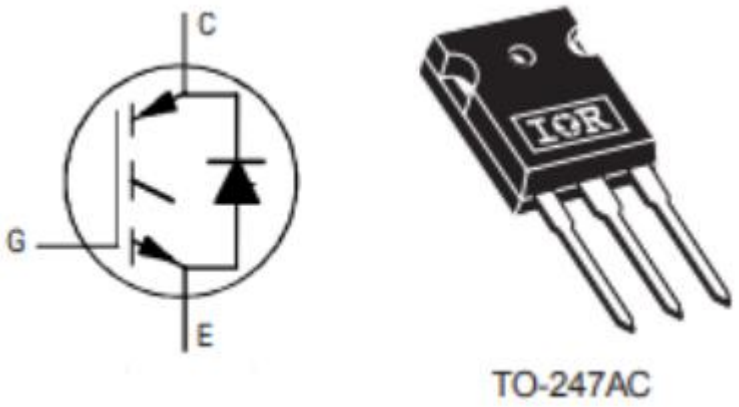


Figure III-33 : IGBT IRG4PH20KD.

Le tableau suivant montre les caractéristiques de l0IGBT IRG4P H20KD :

Tableau 4 : Caractéristiques de l0IGBT IRG4P H20KD

$V_{CES(\max)}$	$V_{CE(sat)}$	I_C@25°	$V_{GE(\max)}$	$V_{GE(th)}(V)$	$f_{sw}(KHz)$	$R_{j-b(\max)}$
1200V	4.04V	11A	±20V	3.5≤ $V_{GE(th)}$ ≤ 6.5	3.5≤ f_{sw} ≤ 6.5	2.1 c/w

Avec :

$V_{CES(\max)}$: Tension Collecteur − Emetteur en blocage .

$V_{CE(sat)}$: Tension Collecteur − Emetteur en conduction.

I_C@25° : Courant de Collecteur à 25°C.

$V_{GE(\max)}$: Tension max Grille − Emetteur.

$V_{GE(th)}(V)$: Tension de seuil Grille − Emetteur.

$f_{sw}(KHz)$: Fréquence de commutation (Switching fréquency).

$R_{j-b(\max)}$: Résistance thermique entre la jonction et le boitier.

Pour plus d'informations sur ce type d'IGBT voir le datasheet donné dans l'annexe B.

3.2.2 Circuit de pilotage

L'IR2110 est un demi-pilote bien connu pour les ponts en H au des demi ponts. Il est destiné à commander des circuits haute tension MOSFET et IGBT, Le circuit et combiné d'une diode et des condensateurs de découplage pour génère une tension de grille.

L'idée est de contrôler un moteur triphasé asynchrone à l'aide d'un demi pont à base d'IGBT de référence « IRG4PH20KD ». Chaque bras se compose de 02 IGBT 01 côté haut et le 2ème côte bas.

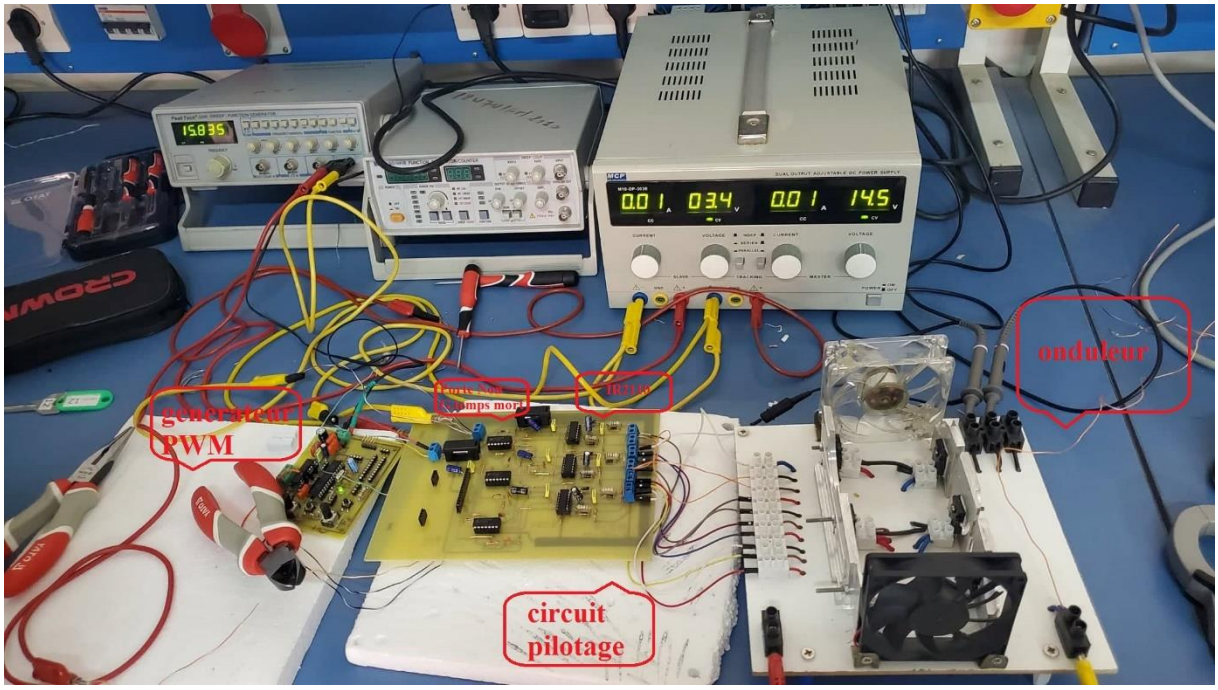


Figure III-34 : Ensemble des éléments du variateur de vitesse.

Dans ce cas, l'IR2110 est un circuit intégré DIP à 14 broches et j'en ai utilisé 3, chaque un de ces circuit intégré contrôle une phase.

Pour alimente notre circuit, nous utilisant une tension de grille 15V. L'alimentation du moteur est séparée et limitée par les IGBT. La tension utilisée du moteur elle peut atteindre jusqu'à 500 volts.

Le canal flottant peut être utilisé pour conduire un N-canal puissance MOSFET ou IGBT dans la configuration de côté élevé qui fonctionne jusqu'à 500.

Pour citer en ce qui concerne VB, VS et HO :

Ce canal a été intégré dans une "cuve d'isolation" capable de flotter de 500 ou 600V à -5V par rapport à la masse d'alimentation (COM). La cuve "flotte" au potentiel de VS, qui est établi par la tension appliquée à VB. Généralement, cette broche est connectée à la source du périphérique côté haut.

Figure III-35 : IR2110.

3.2.3 Avantage du pilote IR2110

IR2110 a une capacite détecteurs internes de sous-tension (UV), Lorsque la tension de grille chute en dessous de ~8,5 V, les sorties sont désactivées.

3.3 Générateur de Temps morts

3.3.1 1er circuit à base de porte XOR

Pour le générateur de temps mort, nous utilisant une port XOR pour retarder le signal entre le HIN et le LIN, donc nous avons implémenté un condensateur de 330pF en parallèle avec une résistance de 510 Ohm pour crée un retard de 0,8µs comme indique la figure 35 qui représente la simulation du circuit avec Proteus ISIS.

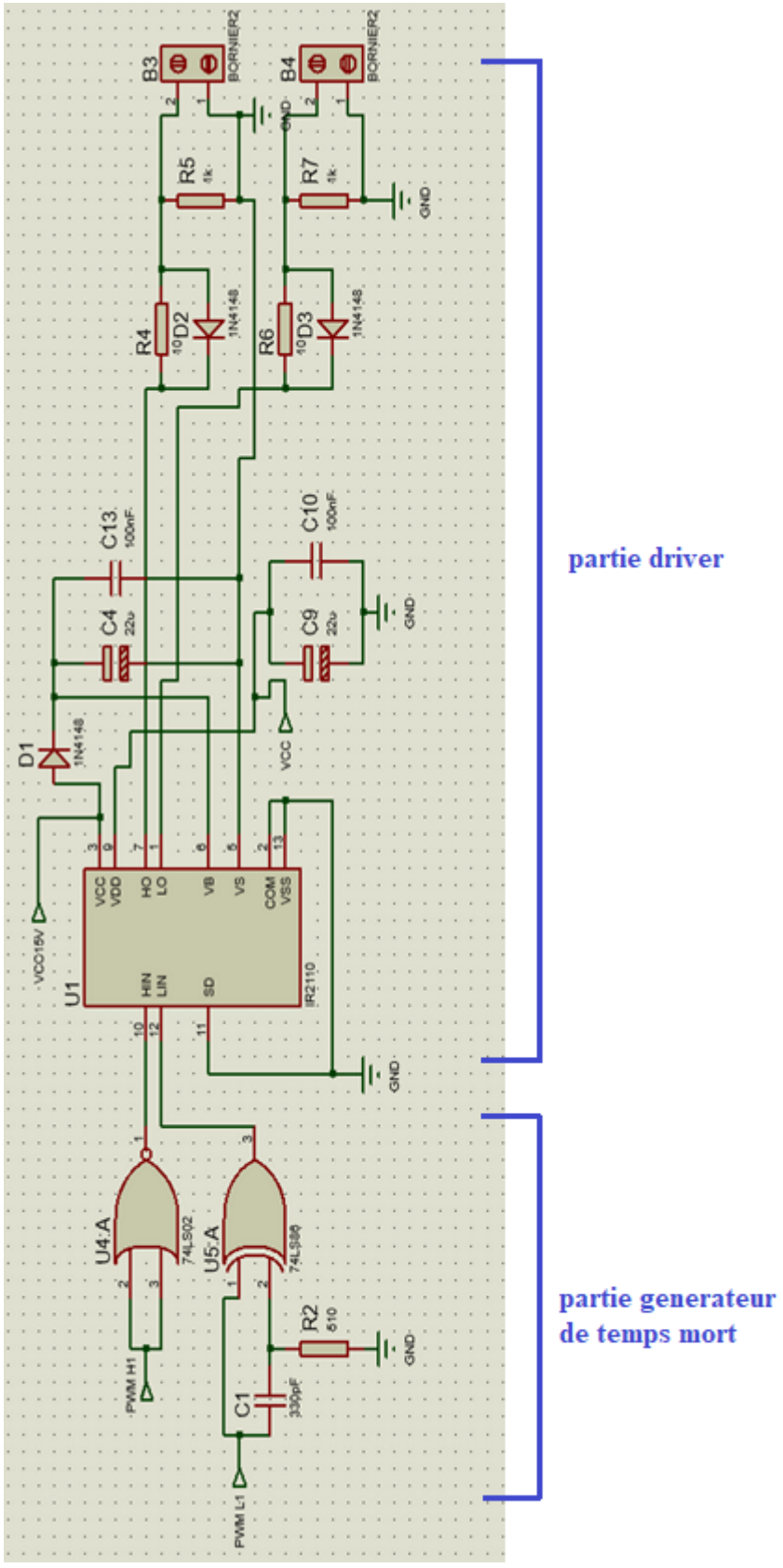


Figure III-I36 : Circuit ISIS du pilotage pour 1 seul bras réalisé avec port XOR et NON.

On a testé sur plaque d'essai la porte Non pour confirmer l'inversion du signal PWM en utilisant des LED comme représente la figure 36.

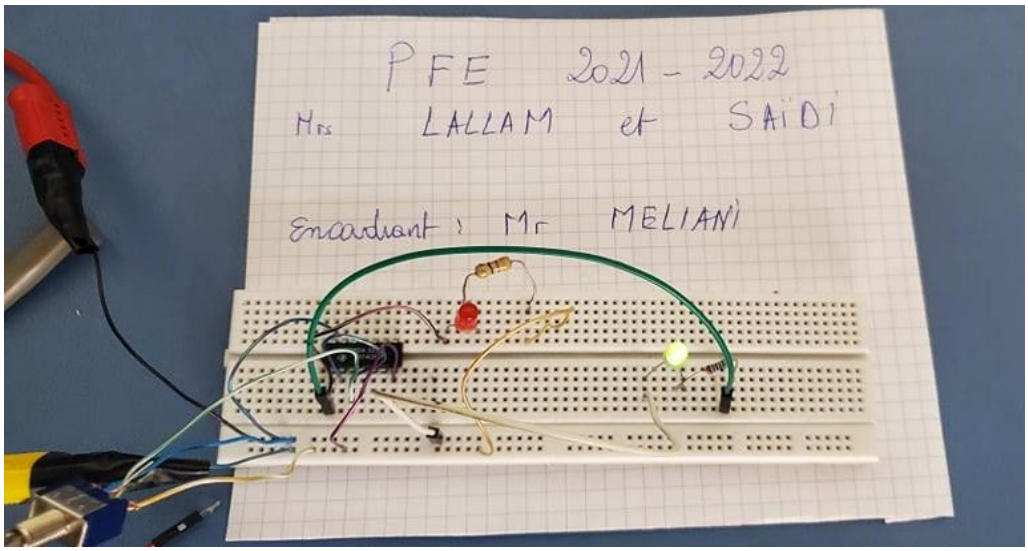


Figure III-37 : Réalisation de la plaque d'essai pour port NON.

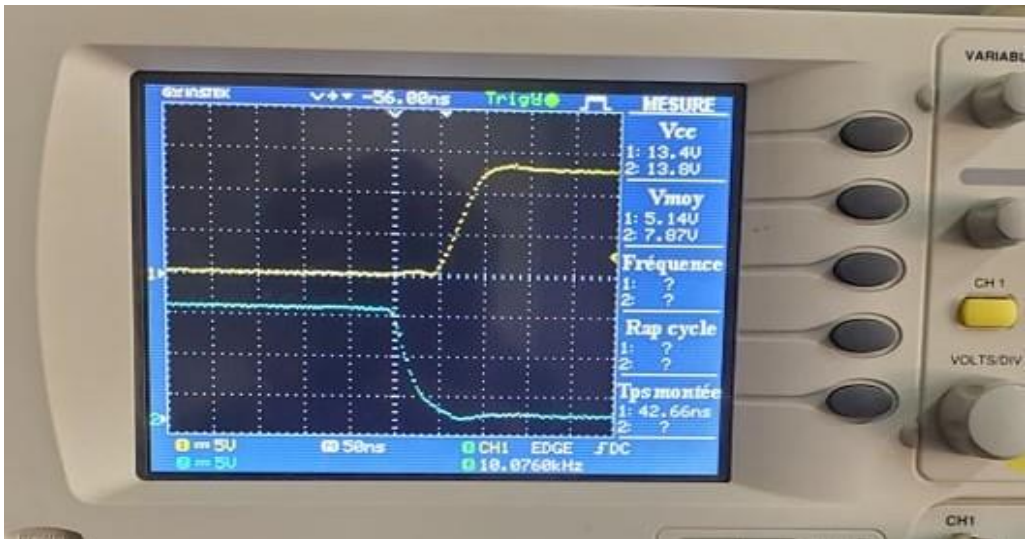


Figure III-38 : Temps mort génère par XOR et port NON.

La figure 37 représente le décalage du temps mort entre les sortie logique HO et LO du circuit de pilotage IR2110 pour une fréquence de 2KHz. Nous avons mesuré sur l'oscilloscope un temps mort de 50Ns entre l'ouverture et la fermeture du côté HO et du côté LO respectivement. Mais malheureusement ça n'a pas vraiment aidé à retarder le signale entre le HIN et le LIN dans les hautes fréquences, ce qui nous a poussé à modifier notre générateur de temps mort par un autre circuit.

3.3.2 2em circuit a base de porte Non 74HC04

Les figure 38 et 39 représentent le nouveau circuit qui se compose d'une porte HEX inverters 74HC04 sous ISIS et plaque d'essai respectivement, le signal PWM passe par une résistance en parelle avec un condensateur et une diode de haute fréquence qui aide à évacuation du courant, puis il inverse le signal 2 fois ce qui crée un temps mort sur la broche HIN.

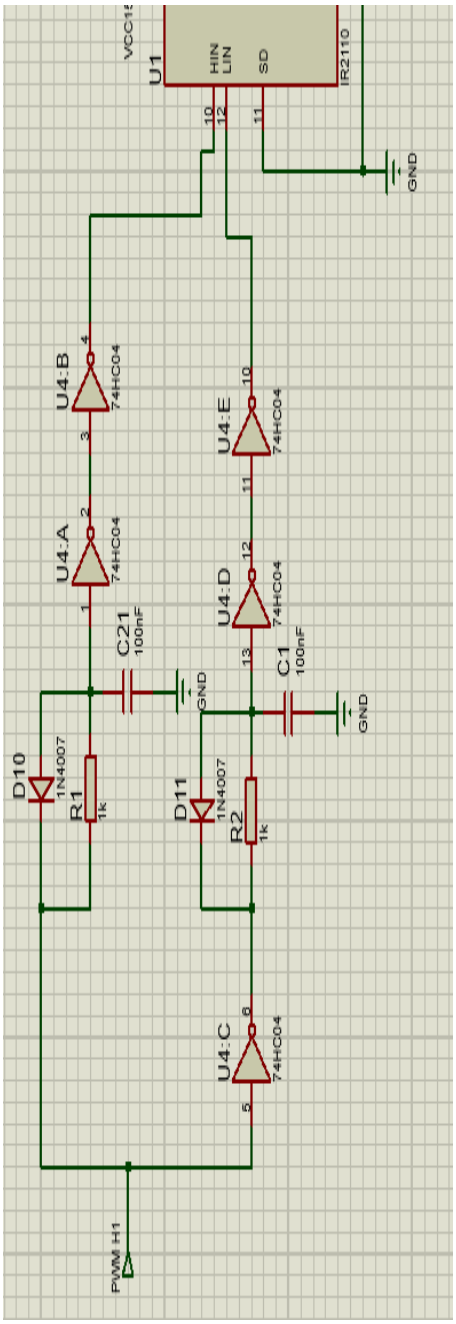


Figure III-39 : Générateur de temps mort avec 74HC04 HEX inverters.

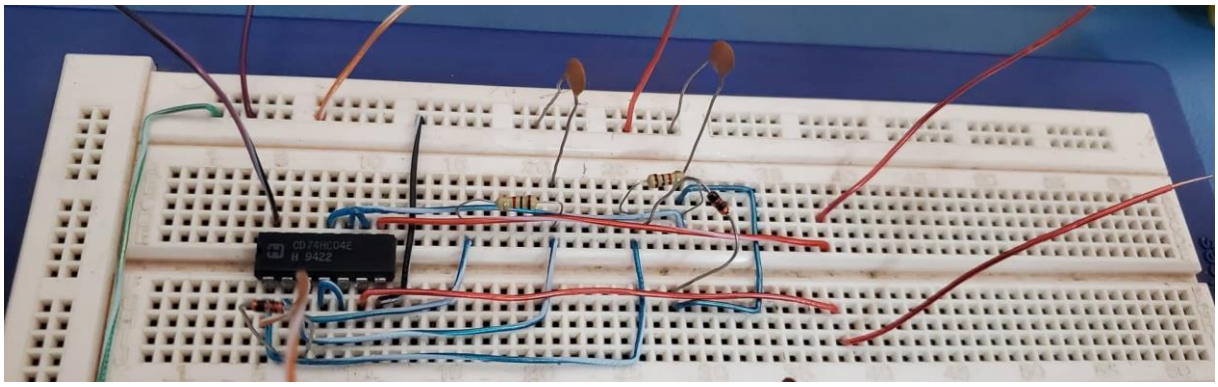


Figure III-40 : Réalisation de la plaque d'essai avec 74HC04.

Les résultats des signaux obtenus comme illustre la figure suivante :

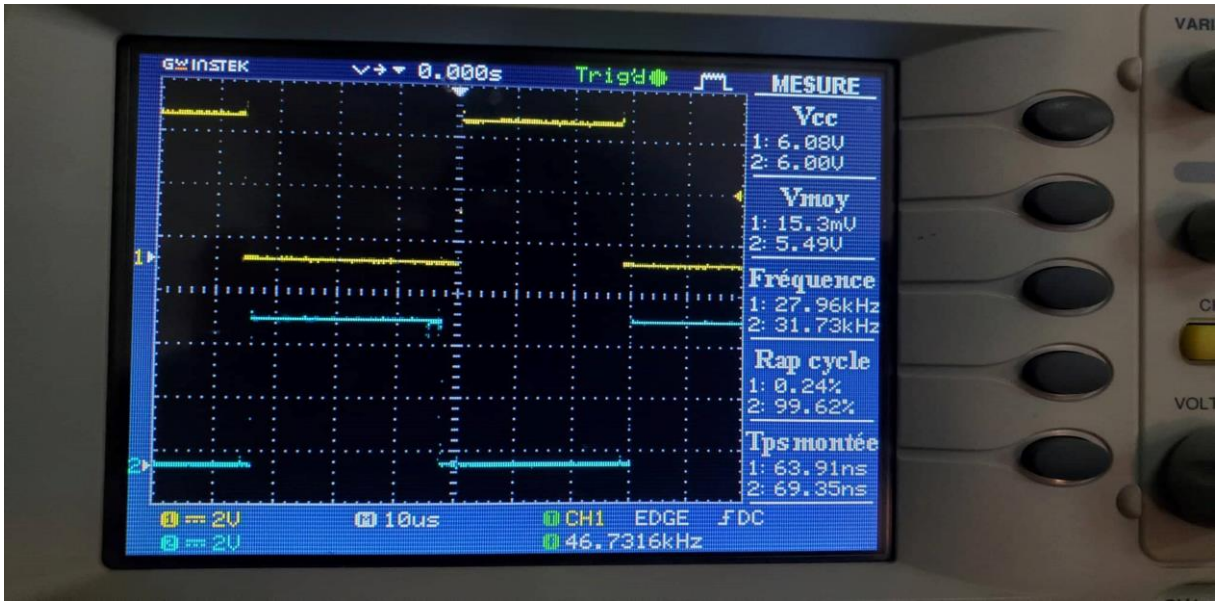


Figure III-41 : Signal de sortie du générateur de temps morts.

La figure 40 représente les signaux obtenus du nouveau circuit réalisé, ce qui a donné un temp mort de 3Ms entre la fermeture et ouverture, avec une fréquence de 16Khz.

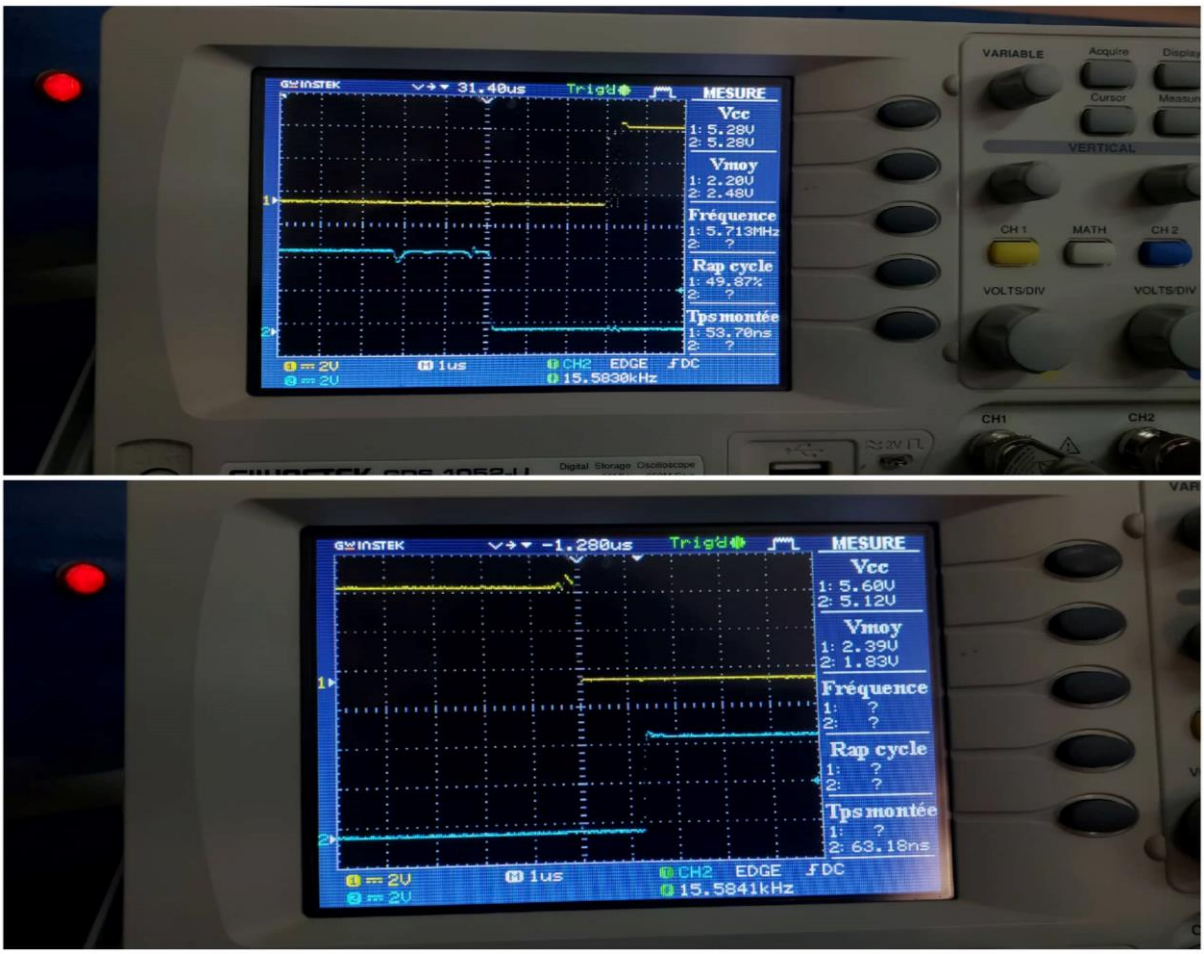


Figure III-42 : Signaux de fermeture et ouverture du cote HO et LO.

Ce nouveau circuit nous a donné de bon résultat comme illustré sûr la figure, nous pouvons voir qu'on a un temps mort entre fermeture et d'ouverture de 3ms, et un temps moins important par rapport aux premiers de 1.3Ms cette différence revient à l'utilisation de 3 porte inverseuse pour l'entre logique Lin et 2 pour le Hin.

Réalisation du circuit équivalent sur ISIS d'un demi ponte en H pilote par IR2110

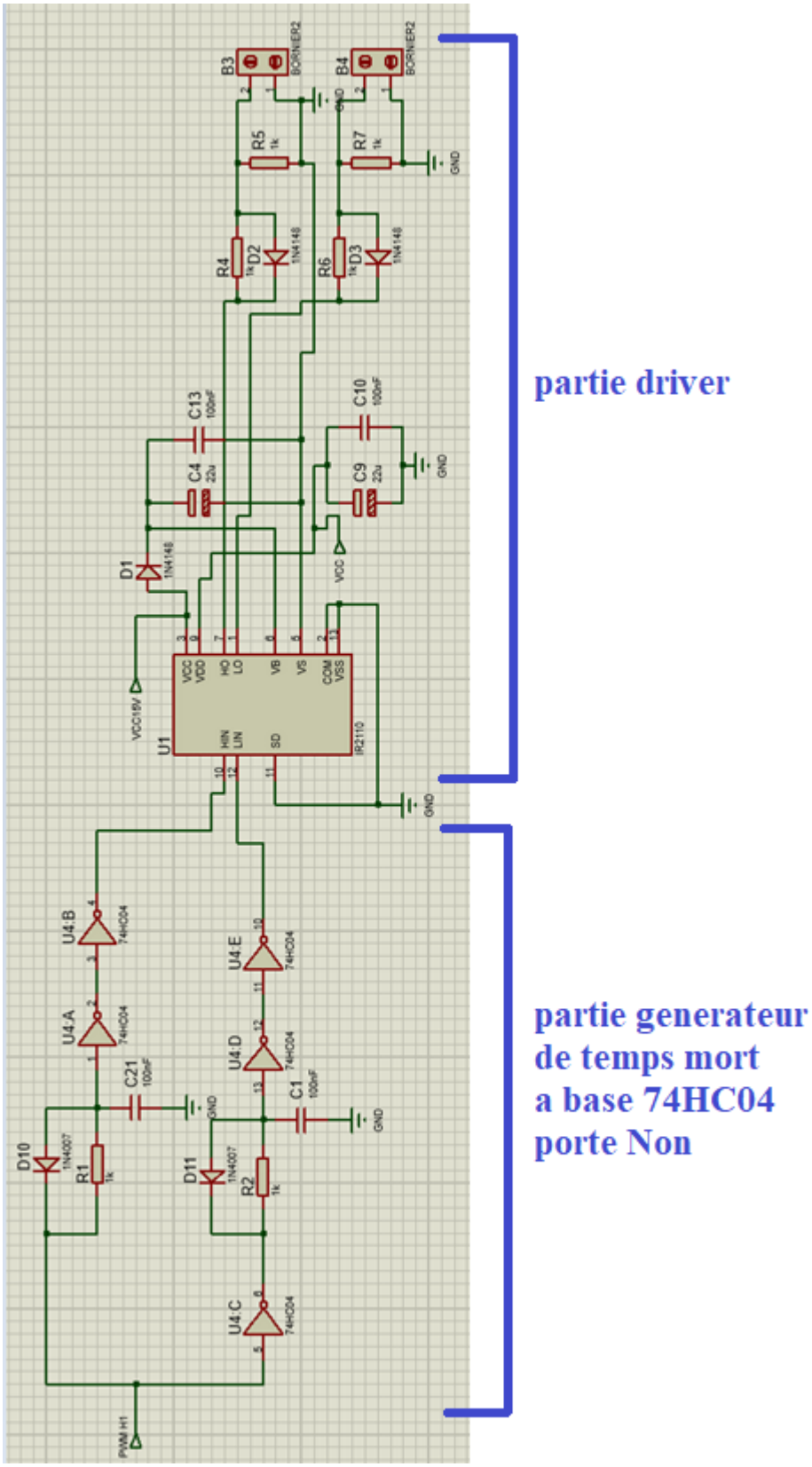


Figure III-43 : Circuit complet d'un bras avec un circuit intègre 74HC04.

La figure 42 représente le circuit complet d'un demi pont on H commande par ESP32.

Résultat obtenu entre les bornes de sortie cote haut et bas :

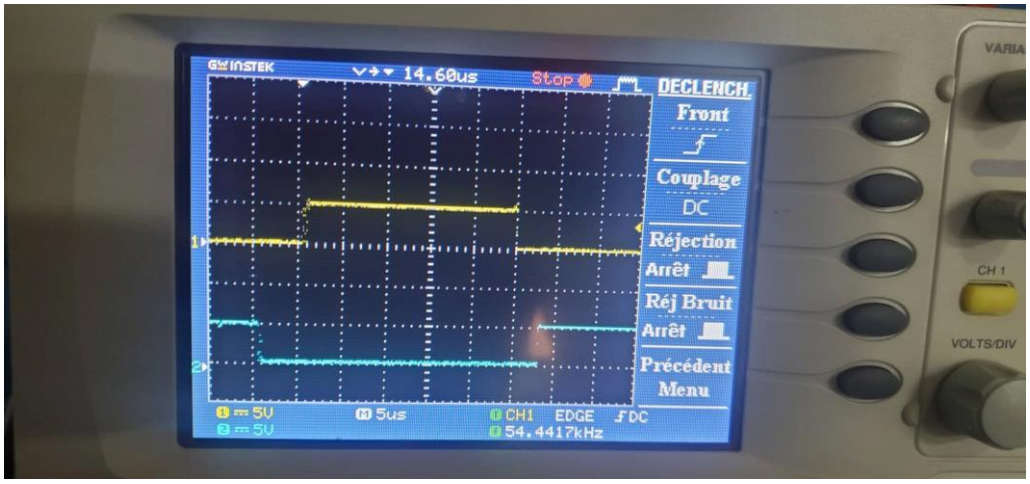


Figure III-44 : Signaux de sortie entre le cote Haut et bas.

Le signal PWM jaune représente celle du cote haut et le bleu représente du cote bas, Nous remarquons qu'on n'a pas de chevauchement entre les 2 signaux, avec un temps mort de Ms pour le côté gauche et un temps mort de Ms pour le côté droit.

3.4 Circuit de pilotage complet avec 3 bras ISIS

La figure 44 représente notre circuit de pilotage simulé avec Proteus ISIS et ça réalisation pratique respectivement, ce circuit permet le pilotage de 3 signaux PWM, grâce à ce processus nous pouvons varier la vitesse d'un moteur triphasé asynchrone.

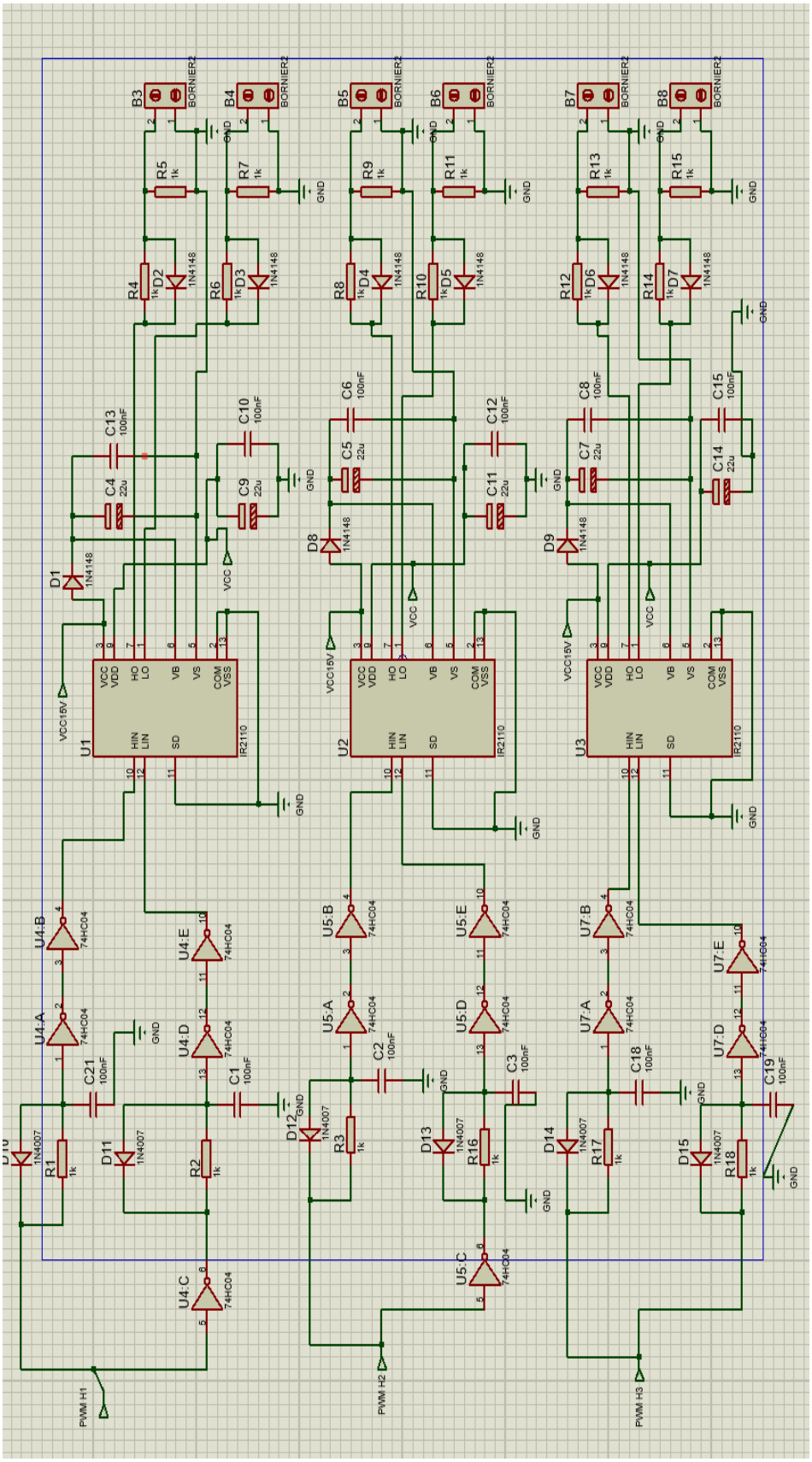


Figure III-45 : Circuit complet du circuit de pilotage.

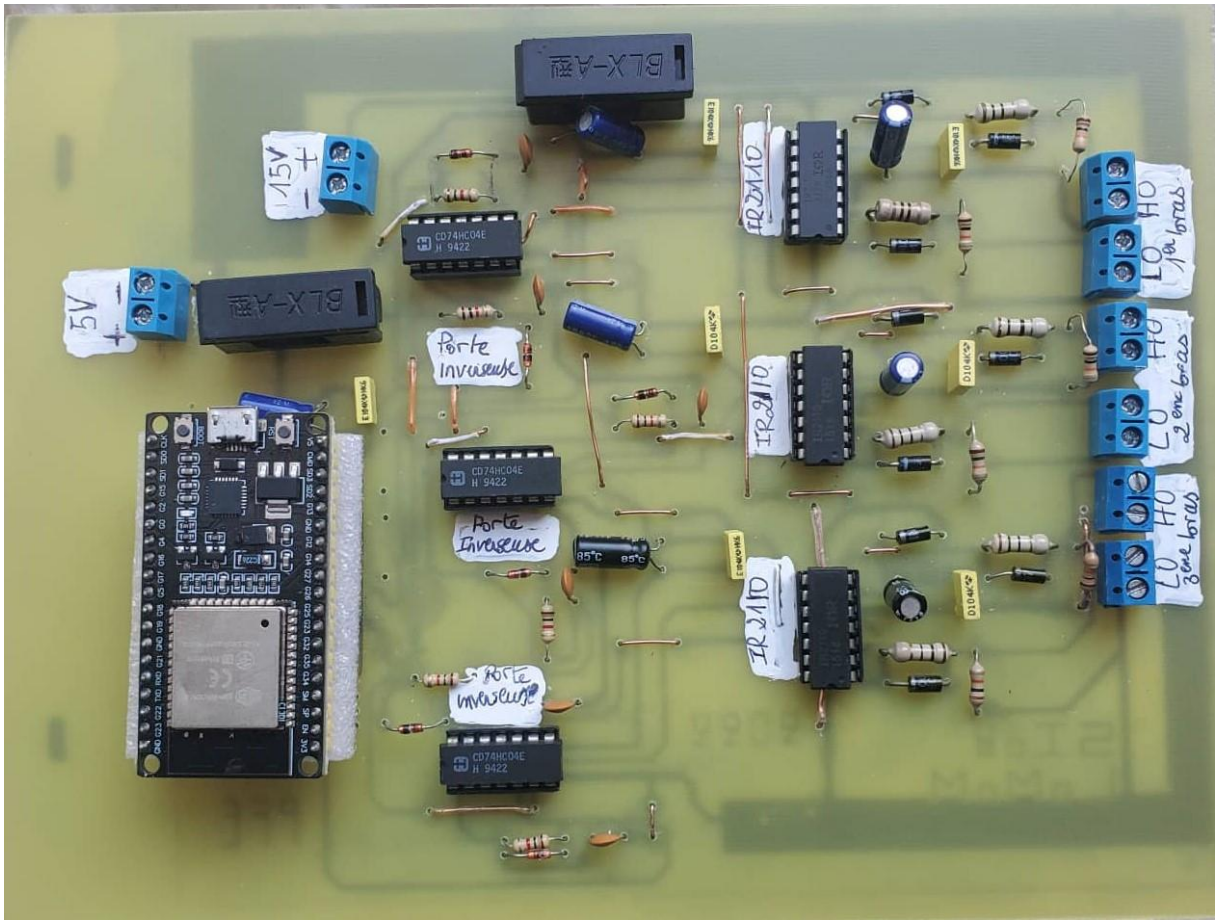


Figure III-46 : Circuit de pilotage.

3.5 Schéma de câblage onduleur

La figure 45-46 représente le schéma de câblage de notre onduleur et le circuit réalisé respectivement, la couleur bleue représente la masse commune, la couleur rouge représente l'alimentation des IGBT, la couleur orange représente le signal génère du cote high, la couleur gris représente le signal génère du cote LOW, la couleur noire représente le out qui alimente le moteur, le marron représente le point a tension flottante.

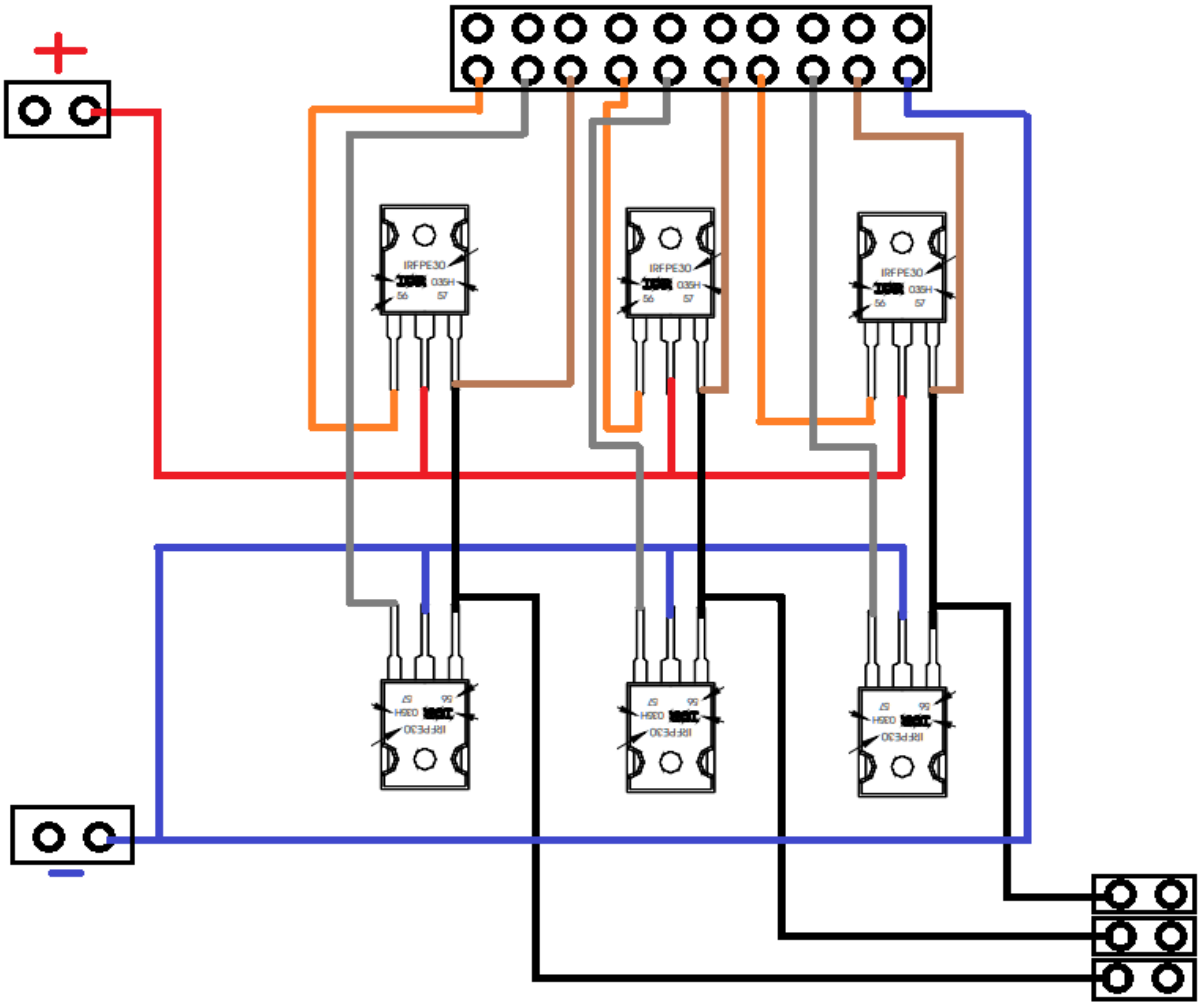


Figure III-47 : Schéma de câblage de l'onduleur.

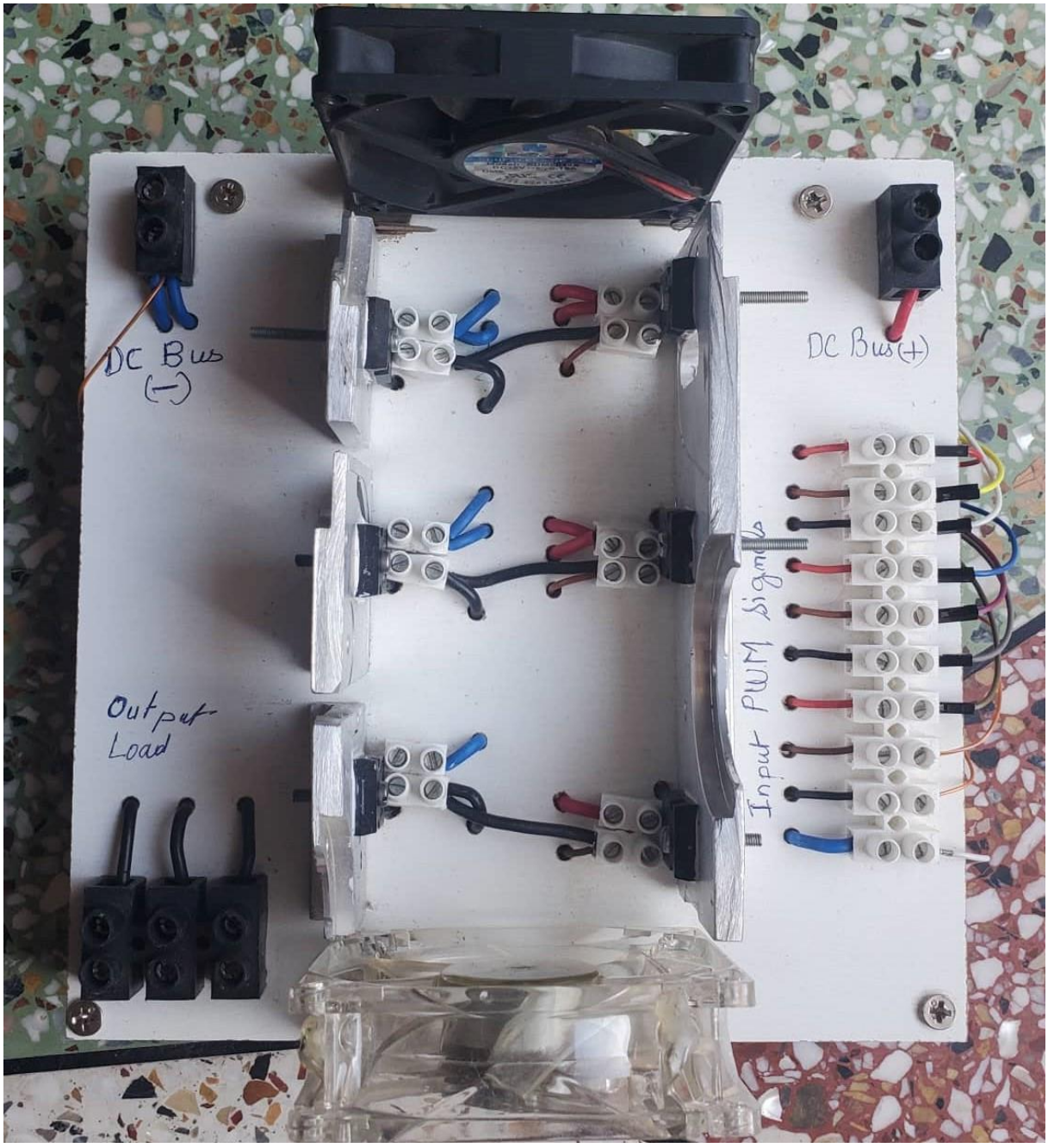


Figure III-48 : Onduleur triphasé a base de IGBT.

3.6 Résultat obtenue

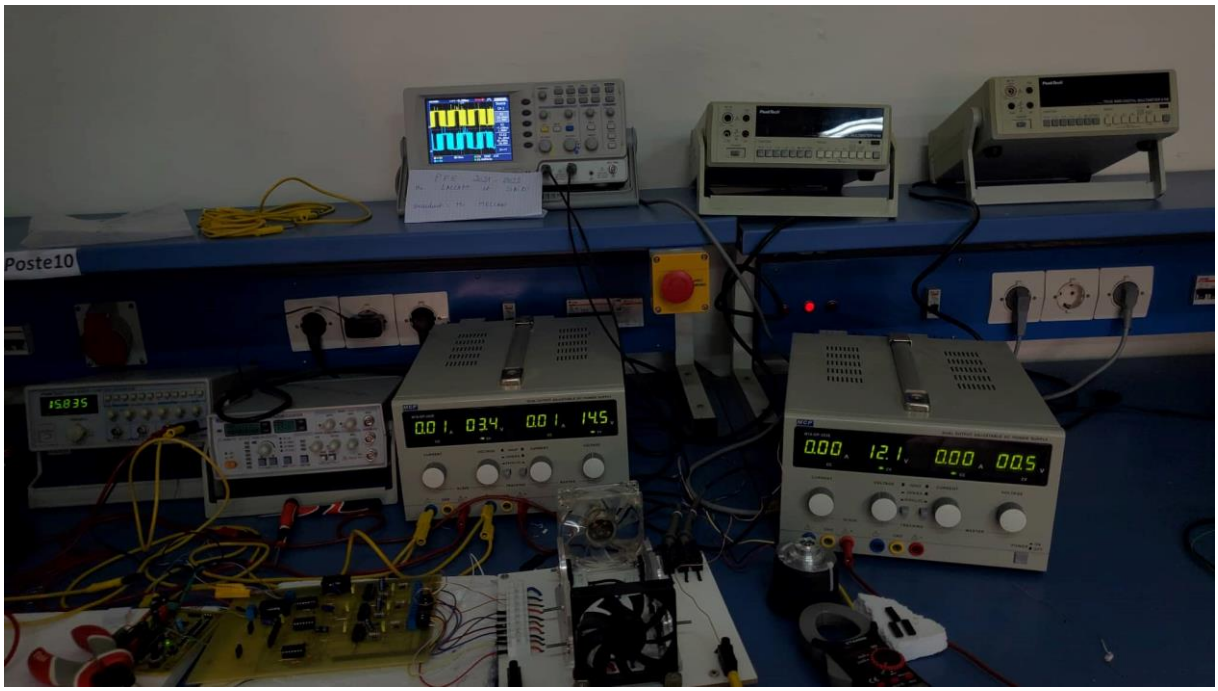


Figure III-49 : Ensemble entre le circuit de pilotage et de puissance.

La figure 46 représente l'ensemble des circuits réalisé qui se compose du circuit de pilotage et l'onduleur.

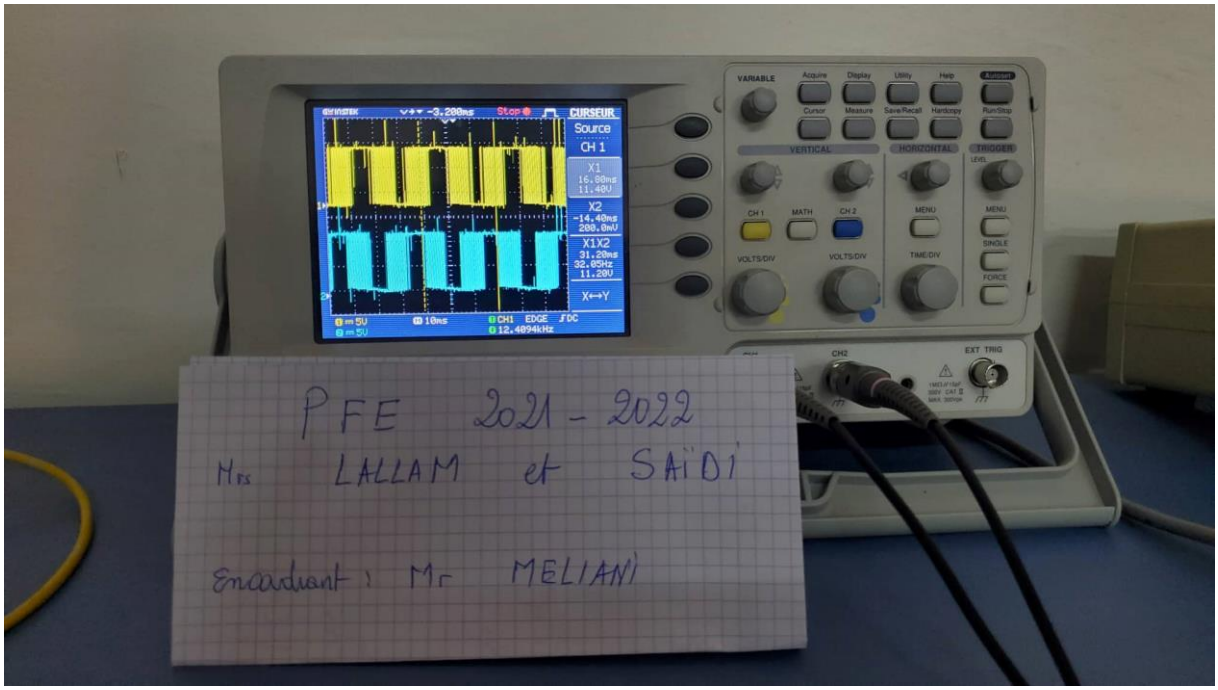


Figure III-50 : Signal entre 2 phase de sortie « out »

La figure 47 représente les signaux obtenus entre les bornes du signal génère du cote LOW et la sortie aux bornes du out, Nous remarquons qu'on n'a pas de chevauchement entre l'ouverture et la fermeture des interrupteurs.

3.6.1 Déphasage entre phase 1 et 2

Nous remarquons dans cette photo que le déphasage entre la phase une et la phase deux et 10.80 ms, sachant que la période d'un cycle et 31 ms donc nous pouvons dire que le déphasage est presque parfait

Figure III-51 : Déphasage entre phase 1 et phase 3

3.6.2 Déphasage entre phase 1 et 3

Comme la photo précédente nous remarquons un déphasage presque parfait entre la phase une et la phase trois 20.80 ms sachant que la période est de 31ms.

Figure III-52 : Déphasage entre la phase 1 et 2

3.6.3 Période

Nous remarquons ici la période 31 ms c'est à dire une fréquence de 32Hz, pour cette même fréquence nous allons vérifier le déphasage entre les trois signaux.

Figure III-53 : Période entre les signaux

3.7 Conclusion

Dans ce chapitre nous avons réalisé notre circuit de puissance qui va se charger de recevoir les trois signaux PWM venons de ESP32 et qui va les modifier en ajoutant trois signaux inverse et un temps mort. Ces signaux vont attaquer les bras du cote haut et bas de notre onduleur à base d'IGBT, ce dernier va commander notre charge.

Les résultats des signaux obtenus sont satisfaisants notamment les résultats du temps mort a base de port Non 74HC04 et le déphasage entre les trois phase de sortie.

Conclusion Générale

Conclusion

Dans ce travail nous avons essayé de concevoir un variateur de vitesse a base de la commande scalaire et qui est commandé via wifi a travers une application mobile

Après une courte introduction nous avons réservé le premier chapitre aux variateurs de vitesse, leur conception et leur principe de fonctionnement.

Après dans le chapitre suivant nous avons expliqué en détail les différentes étapes pour implémenter notre logique afin de produire les signaux nécessaires pour faire tourner le moteur.

Puis le dernier chapitre est réservé à la réalisation du circuit de commande et à l'interprétation des résultats obtenus.

Malheureusement nous avons rencontré plusieurs problèmes lors de la création de ce variateur

Parmi ces problèmes, la présence du collecteur sur le dos de l'IGBT ce que nous a poussé à séparer les trois IGBT dans le coté bas de l'onduleur.

Le deuxième problème est que nos esp32 ne sont pas originaux, donc ils présentent une petite anomalie, leur pin d'alimentation ne fonctionne pas donc nous avons obligé de l'alimenté seulement l'aide d'un câble.

Référence bibliographie

Webographie

[3] «Energie et alimentation pour tous,» [En ligne]. Available: https://crushtymks.com/fr/electric-motor/262-induction-machines-8211-historical-touch.html.

[6] «Gozuk,» [En ligne]. Available: http://www.vfds.in/variable-frequency-drive-history-283991.html.

30 05 2022. [En ligne]. Available: http://fablab37110.ovh/doku.php?id=start:arduino:esp32.

[9] K. R. Guy SEGUIER, 2017/2018. [En ligne]. Available: https://f2school.com/wp-content/uploads/2019/10/Electronique-de-puissance-cours-N%C2%B02.pdf.

[10] «engergie plus,» [En ligne]. Available: https://energieplus-lesite.be/techniques/ascenseurs7/variateurs-de-vitesse/.

[11] B. Schweber, «Electronics360,» 06 08 2016. [En ligne]. Available: https://electronics360.globalspec.com/article/7114/drivers-the-critical-interface-between-a-circuit-and-its-load.

[13] «energieplus,» 25 septembre 2007. [En ligne]. Available: https://energieplus-lesite.be/techniques/ascenseurs7/variateurs-de-vitesse/.

[14] «wikipedia,» April 2009. [En ligne]. Available: https://en.wikipedia.org/wiki/Pulse-width_modulation.

[15] «wikipedia,» mars 2022. [En ligne]. Available: https://fr.wikipedia.org/wiki/Transformée_de_Park .

[16] «wikipedia,» 11 June 2022. [En ligne]. Available: https://en.wikipedia.org/wiki/ESP32 .

[17] «espressif,» [En ligne]. Available: https://www.espressif.com/en/products/socs/esp32.

[18] «flutter.dev,» [En ligne]. Available: https://docs.flutter.dev/resources/architectural-overview.

[19] E. Wilson, «medium,» 28 Dec 2020. [En ligne]. Available: https://medium.com/flutter-community/top-10-reasons-flutter-is-better-for-your-app-development-30d50e345b29.

[20] «microsoft,» 18 05 2022. [En ligne]. Available: https://docs.microsoft.com/en-us/power-apps/maker/portals/vs-code-extension.

[21] R. MISCHIANTI, «mischianti,» 04 06 2020. [En ligne]. Available: https://www.mischianti.org/2020/06/04/esp32-integrated-spiffs-filesystem-part-2/.

[23] «Castel'Lab le Fablab MJC de Château-Renault,» 30 05 2022. [En ligne]. Available: http://fablab37110.ovh/doku.php?id=start:arduino:esp32.

bibliographie

[1] M.ZIAD et N.TAMADART, «Etude et amélioration de la commande d'une,» TIZI-OUZOU, 2010/2011.

[2] B. A. elkader et M. Mohamed, «Etude et réalisation d'un variateur de vitesse,» KHEMIS MILIANA, 2017/2018.

[4] I. Pavel, «L'invention du moteur synchrone par Nikola Tesla,» *OpenEdition Journals,* n° %1381,968, 1887.

[5] M. H. B. G. e. M. ABBAS, «Réalisation d'un onduleur comme un banc d'essai pour le laboratoire pédagogique d'électronique de puissance,» tlemcen, 2020.

[8] B. BAYALA, *LA MACHINE ASYNCHRONE,* Edition revue, 2010.

[12] L. Schirone et M. Macellari, «Générateur de temps mort dynamique à usage général,» chez *2015 Conférence internationale sur l'énergie électrique propre (ICCEP)*, Taormina, Italie, 06/08/2015.

[22] K. R. Guy SEGUIER, *Cours Electronique de Puissance,* 2017/2018.

Annexe A : Composant et caractéristique

Datasheet IGBT(IRG4PH20KD)

International
IOR Rectifier

PD -91776

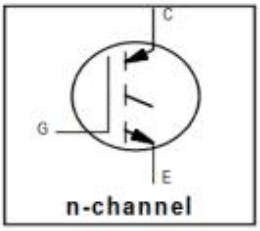
IRG4PH20K

INSULATED GATE BIPOLAR TRANSISTOR

Short Circuit Rated
UltraFast IGBT

Features

- High short circuit rating optimized for motor control, t_{sc} =10µs, V_{CC} = 720V , T_J = 125°C, V_{GE} = 15V
- Combines low conduction losses with high switching speed
- Latest generation design provides tighter parameter distribution and higher efficiency than previous generations



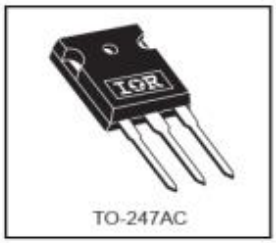
V_{CES} = 1200V

$V_{CE(on)\ typ.}$ = 3.17V

@V_{GE} = 15V, I_C = 5.0A

Benefits

- As a Freewheeling Diode we recommend our HEXFRED™ ultrafast, ultrasoft recovery diodes for minimum EMI / Noise and switching losses in the Diode and IGBT
- Latest generation 4 IGBT's offer highest power density motor controls possible



TO-247AC

Absolute Maximum Ratings

	Parameter	Max.	Units
V_{CES}	Collector-to-Emitter Voltage	1200	V
I_C @ T_C = 25°C	Continuous Collector Current	11	A
I_C @ T_C = 100°C	Continuous Collector Current	5.0	
I_{CM}	Pulsed Collector Current ①	22	
I_{LM}	Clamped Inductive Load Current ②	22	
t_{sc}	Short Circuit Withstand Time	10	µs
V_{GE}	Gate-to-Emitter Voltage	±20	V
E_{ARV}	Reverse Voltage Avalanche Energy ③	130	mJ
P_D @ T_C = 25°C	Maximum Power Dissipation	60	W
P_D @ T_C = 100°C	Maximum Power Dissipation	24	
T_J T_{STG}	Operating Junction and Storage Temperature Range	-55 to +150	°C
	Soldering Temperature, for 10 sec.	300 (0.063 in. (1.6mm) from case)	
	Mounting torque, 6-32 or M3 screw.	10 lbf•in (1.1N•m)	

Thermal Resistance

	Parameter	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	—	2.1	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat, Greased Surface	0.24	—	
$R_{\theta JA}$	Junction-to-Ambient, typical socket mount	—	40	
Wt	Weight	6 (0.21)	—	g (oz)

www.irf.com 1

6/25/98

Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)CES}$	Collector-to-Emitter Breakdown Voltage	1200	—	—	V	V_{GE} = 0V, I_C = 250µA
$V_{(BR)ECS}$	Emitter-to-Collector Breakdown Voltage ④	18	—	—	V	V_{GE} = 0V, I_C = 1.0A
$\Delta V_{(BR)CES}/\Delta T_J$	Temperature Coeff. of Breakdown Voltage	—	1.13	—	V/°C	V_{GE} = 0V, I_C = 2.5mA
$V_{CE(ON)}$	Collector-to-Emitter Saturation Voltage	—	3.17	4.3	V	I_C = 5.0A V_{GE} = 15V
		—	4.04	—		I_C = 11A See Fig.2, 5
		—	2.84	—		I_C = 5.0A , T_J = 150°C
$V_{GE(th)}$	Gate Threshold Voltage	3.5	—	6.5		V_{CE} = V_{GE}, I_C = 250µA
$\Delta V_{GE(th)}/\Delta T_J$	Temperature Coeff. of Threshold Voltage	—	-10	—	mV/°C	V_{CE} = V_{GE}, I_C = 1mA
g_{fe}	Forward Transconductance ⑤	2.3	3.5	—	S	V_{CE} = 100 V, I_C = 5.0A
I_{CES}	Zero Gate Voltage Collector Current	—	—	250	µA	V_{GE} = 0V, V_{CE} = 1200V
		—	—	2.0		V_{GE} = 0V, V_{CE} = 10V, T_J = 25°C
		—	—	1000		V_{GE} = 0V, V_{CE} = 1200V, T_J = 150°C
I_{GES}	Gate-to-Emitter Leakage Current	—	—	±100	nA	V_{GE} = ±20V

Switching Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
Q_g	Total Gate Charge (turn-on)	—	28	43	nC	I_C = 5.0A
Q_{ge}	Gate - Emitter Charge (turn-on)	—	4.4	6.6		V_{CC} = 400V See Fig.8
Q_{gc}	Gate - Collector Charge (turn-on)	—	12	18		V_{GE} = 15V
$t_{d(on)}$	Turn-On Delay Time	—	23	—	ns	
t_r	Rise Time	—	26	—		T_J = 25°C
$t_{d(off)}$	Turn-Off Delay Time	—	93	140		I_C =5.0A, V_{CC} = 960V
t_f	Fall Time	—	270	400		V_{GE} = 15V, R_G = 50Ω
E_{on}	Turn-On Switching Loss	—	0.45	—	mJ	Energy losses include "tail"
E_{off}	Turn-Off Switching Loss	—	0.44	—		See Fig. 9,10,14
E_{ts}	Total Switching Loss	—	0.89	1.2		
t_{sc}	Short Circuit Withstand Time	10	—	—	µs	V_{CC} = 720V, T_J = 125°C
						V_{GE} = 15V, R_G = 50Ω
$t_{d(on)}$	Turn-On Delay Time	—	23	—	ns	T_J = 150°C,
t_r	Rise Time	—	28	—		I_C = 5.0A, V_{CC} = 960
$t_{d(off)}$	Turn-Off Delay Time	—	100	—		V_{GE} = 15V, R_G = 50Ω
t_f	Fall Time	—	620	—		Energy losses include "tail"
E_{ts}	Total Switching Loss	—	1.7	—	mJ	See Fig. 10,11,14
L_E	Internal Emitter Inductance	—	13	—	nH	Measured 5mm from package
C_{ies}	Input Capacitance	—	435	—	pF	V_{GE} = 0V
C_{oes}	Output Capacitance	—	44	—		V_{CC} = 30V See Fig. 7
C_{res}	Reverse Transfer Capacitance	—	8.3	—		f = 1.0MHz

Notes:

① Repetitive rating; V_{GE} = 20V, pulse width limited by max. junction temperature. (See fig. 13b)

② V_{CC} = 80%(V_{CES}), V_{GE} = 20V, L = 10µH, R_G =50Ω, (See fig. 13a)

③ Repetitive rating; pulse width limited by maximum junction temperature.

④ Pulse width ≤ 80µs; duty factor ≤ 0.1%.

⑤ Pulse width 5.0µs, single shot.

Datasheet Driver (IR2110)

International
IOR Rectifier

Data Sheet No. PD60147 rev.V

IR2110(S)PbF/IR2113(S)PbF

HIGH AND LOW SIDE DRIVER

Features

- Floating channel designed for bootstrap operation
  Fully operational to +500V or +600V
  Tolerant to negative transient voltage
  dV/dt immune
- Gate drive supply range from 10 to 20V
- Undervoltage lockout for both channels
- 3.3V logic compatible
  Separate logic supply range from 3.3V to 20V
  Logic and power ground ±5V offset
- CMOS Schmitt-triggered inputs with pull-down
- Cycle by cycle edge-triggered shutdown logic
- Matched propagation delay for both channels
- Outputs in phase with inputs

Product Summary

V_{OFFSET} (IR2110)	500V max.
(IR2113)	600V max.
I_O+/-	2A / 2A
V_{OUT}	10 - 20V
$t_{on/off}$ (typ.)	120 & 94 ns
Delay Matching (IR2110)	10 ns max.
(IR2113)	20ns max.

Packages

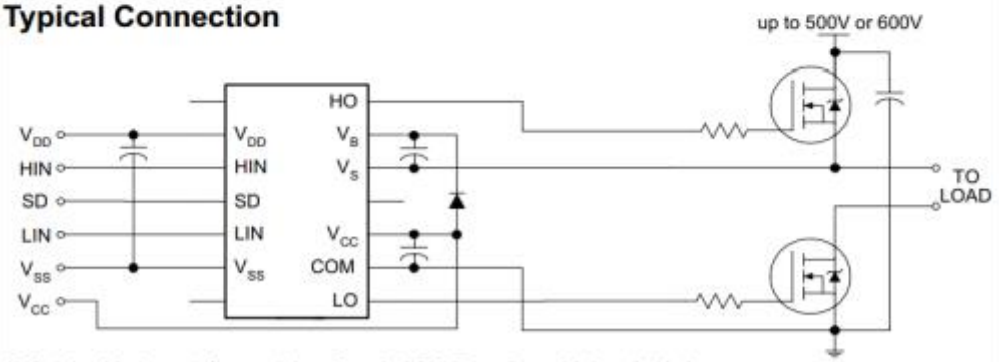
14-Lead PDIP
IR2110/IR2113

16-Lead SOIC
IR2110S/IR2113S

Description

The IR2110/IR2113 are high voltage, high speed power MOSFET and IGBT drivers with independent high and low side referenced output channels. Proprietary HVIC and latch immune CMOS technologies enable ruggedized monolithic construction. Logic inputs are compatible with standard CMOS or LSTTL output, down to 3.3V logic. The output drivers feature a high pulse current buffer stage designed for minimum driver cross-conduction. Propagation delays are matched to simplify use in high frequency applications. The floating channel can be used to drive an N-channel power MOSFET or IGBT in the high side configuration which operates up to 500 or 600 volts.

Typical Connection



(Refer to Lead Assignments for correct pin configuration). This/These diagram(s) show electrical connections only. Please refer to our Application Notes and DesignTips for proper circuit board layout.

Absolute Maximum Ratings

Absolute maximum ratings indicate sustained limits beyond which damage to the device may occur. All voltage parameters are absolute voltages referenced to COM. The thermal resistance and power dissipation ratings are measured under board mounted and still air conditions. Additional information is shown in Figures 28 through 35.

Symbol	Definition		Min.	Max.	Units
V_B	High side floating supply voltage	(IR2110)	-0.3	525	V
		(IR2113)	-0.3	625	
V_S	High side floating supply offset voltage		V_B - 25	V_B + 0.3	
V_{HO}	High side floating output voltage		V_S - 0.3	V_B + 0.3	
V_{CC}	Low side fixed supply voltage		-0.3	25	
V_{LO}	Low side output voltage		-0.3	V_{CC} + 0.3	
V_{DD}	Logic supply voltage		-0.3	V_{SS} + 25	
V_{SS}	Logic supply offset voltage		V_{CC} - 25	V_{CC} + 0.3	
V_{IN}	Logic input voltage (HIN, LIN & SD)		V_{SS} - 0.3	V_{DD} + 0.3	
dV_s/dt	Allowable offset supply voltage transient (figure 2)		—	50	V/ns
P_D	Package power dissipation @ $T_A \leq +25°C$	(14 lead DIP)	—	1.6	W
		(16 lead SOIC)	—	1.25	
R_{THJA}	Thermal resistance, junction to ambient	(14 lead DIP)	—	75	°C/W
		(16 lead SOIC)	—	100	
T_J	Junction temperature		—	150	°C
T_S	Storage temperature		-55	150	
T_L	Lead temperature (soldering, 10 seconds)		—	300	

Recommended Operating Conditions

The input/output logic timing diagram is shown in figure 1. For proper operation the device should be used within the recommended conditions. The V_S and V_{SS} offset ratings are tested with all supplies biased at 15V differential. Typical ratings at other bias conditions are shown in figures 36 and 37.

Symbol	Definition		Min.	Max.	Units
V_B	High side floating supply absolute voltage		V_S + 10	V_S + 20	V
V_S	High side floating supply offset voltage	(IR2110)	Note 1	500	
		(IR2113)	Note 1	600	
V_{HO}	High side floating output voltage		V_S	V_B	
V_{CC}	Low side fixed supply voltage		10	20	
V_{LO}	Low side output voltage		0	VCC	
V_{DD}	Logic supply voltage		V_{SS} + 3	V_{SS} + 20	
V_{SS}	Logic supply offset voltage		-5 (Note 2)	5	
V_{IN}	Logic input voltage (HIN, LIN & SD)		V_{SS}	V_{DD}	
T_A	Ambient temperature		-40	125	°C

Note 1: Logic operational for V_S of -4 to +500V. Logic state held for V_S of -4V to $-V_{BS}$. (Please refer to the Design Tip DT97-3 for more details).
Note 2: When V_{DD} < 5V, the minimum V_{SS} offset is limited to $-V_{DD}$.

Dynamic Electrical Characteristics

V_{BIAS} (V_{CC}, V_{BS}, V_{DD}) = 15V, C_L = 1000 pF, T_A = 25°C and V_{SS} = COM unless otherwise specified. The dynamic electrical characteristics are measured using the test circuit shown in Figure 3.

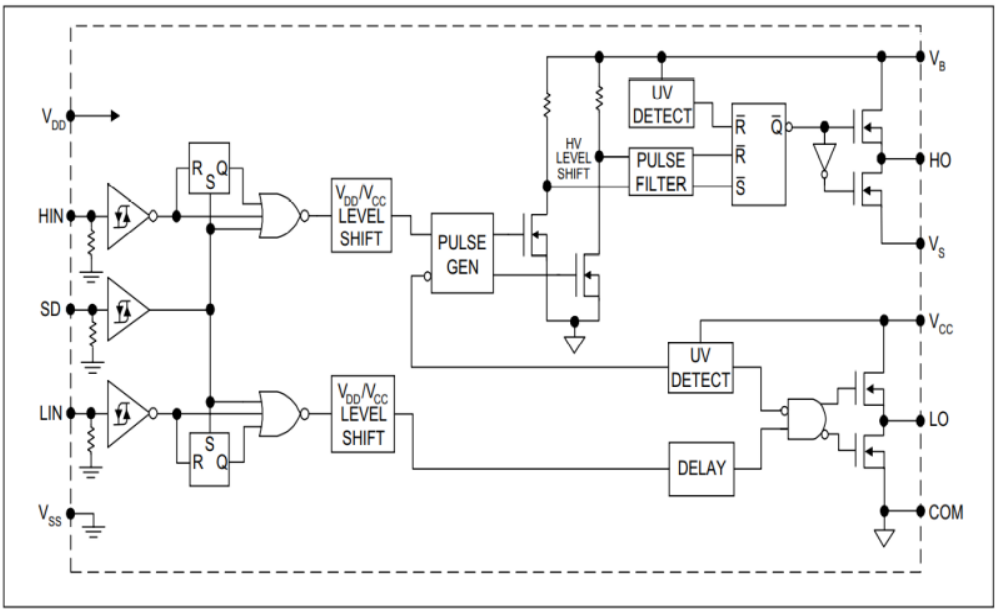
Symbol	Definition	Figure	Min.	Typ.	Max.	Units	Test Conditions
t_{on}	Turn-on propagation delay	7	—	120	150	ns	V_S = 0V
t_{off}	Turn-off propagation delay	8	—	94	125		V_S = 500V/600V
t_{sd}	Shutdown propagation delay	9	—	110	140		V_S = 500V/600V
t_r	Turn-on rise time	10	—	25	35		
t_f	Turn-off fall time	11	—	17	25		
MT	Delay matching, HS & LS (IR2110)	—	—	—	10		
	turn-on/off (IR2113)	—	—	—	20		

Static Electrical Characteristics

V_{BIAS} (V_{CC}, V_{BS}, V_{DD}) = 15V, T_A = 25°C and V_{SS} = COM unless otherwise specified. The V_{IN}, V_{TH} and I_{IN} parameters are referenced to V_{SS} and are applicable to all three logic input leads: HIN, LIN and SD. The V_O and I_O parameters are referenced to COM and are applicable to the respective output leads: HO or LO.

Symbol	Definition	Figure	Min.	Typ.	Max.	Units	Test Conditions
V_{IH}	Logic "1" input voltage	12	9.5	—	—	V	
V_{IL}	Logic "0" input voltage	13	—	—	6.0		
V_{OH}	High level output voltage, V_{BIAS} - V_O	14	—	—	1.2		I_O = 0A
V_{OL}	Low level output voltage, V_O	15	—	—	0.1		I_O = 0A
I_{LK}	Offset supply leakage current	16	—	—	50	µA	V_B=V_S = 500V/600V
I_{QBS}	Quiescent V_{BS} supply current	17	—	125	230		V_{IN} = 0V or V_{DD}
I_{QCC}	Quiescent V_{CC} supply current	18	—	180	340		V_{IN} = 0V or V_{DD}
I_{QDD}	Quiescent V_{DD} supply current	19	—	15	30		V_{IN} = 0V or V_{DD}
I_{IN+}	Logic "1" input bias current	20	—	20	40		V_{IN} = V_{DD}
I_{IN-}	Logic "0" input bias current	21	—	—	1.0		V_{IN} = 0V
V_{BSUV+}	V_{BS} supply undervoltage positive going threshold	22	7.5	8.6	9.7	V	
V_{BSUV-}	V_{BS} supply undervoltage negative going threshold	23	7.0	8.2	9.4		
V_{CCUV+}	V_{CC} supply undervoltage positive going threshold	24	7.4	8.5	9.6		
V_{CCUV-}	V_{CC} supply undervoltage negative going threshold	25	7.0	8.2	9.4		
I_{O+}	Output high short circuit pulsed current	26	2.0	2.5	—	A	V_O = 0V, V_{IN} = V_{DD} PW ≤ 10 µs
I_{O-}	Output low short circuit pulsed current	27	2.0	2.5	—		V_O = 15V, V_{IN} = 0V PW ≤ 10 µs

Structure de IR2110 :



Bon maintenant nous allons définir les pins de notre microcontrôleur IR 2110 :

Tout d'abord, on a les broche 1 et 7 se sont les sorties du pilote pour le cote Haut et bas, et 9 et 13 représente les entres de commande numérique 15V.

La pin 9 qui représente VDD est la tension positive logique numérique d'une valeur de 3V jusqu'à 15V ce qui nous permet l'utilisation de la logique CMOS ou LS TTL.

CMOS (Complementary métal-oxide-semiconducteur) : Cet élément de RAM statique peut stocker, traiter et transmettre des données numériques et analogiques de façon simultanée.

Les entrées logiques sont compatibles avec sortie standard CMOS ou LSTTL, jusqu'à 3,3 V logique, Les délais de propagation sont adaptés pour simplifier l'utilisation dans les applications à haute fréquence.

La broche 10 ou HIN contrôle la sortie sur la broche 7 étiquetée HO, qui contrôle le MOSFET du côté HAUT. Cette entrée doit être modulée en largeur d'impulsion.

La broche 12 ou LIN contrôle le MOSFET côté BAS connecté à la broche 1 LO. Si on applique une tension de 10-15V sur la broche 1 il permet d'active le MOSFET cote Low.

La broche SD représente arrêt ou elle permet de désactive toutes les sorties lorsqu'elles sont élèves, on la connecte à la masse numérique.

La masse négative de l'alimentation numérique est connue comme la broche 13 ou VSS. Ceci est fréquemment connecté à la masse COM ou à la broche 2 de la tension HT de l'alimentation du moteur.

Les broches 8 et 14 ne sont pas connectées.

Les broches 10, 11 et 12 ont toutes des entrées de déclenchement de Schmitt (le Schmitt Declerator est un multivibrateur avec deux états stables, la sortie restant dans l'un des états stables jusqu'à ce qu'une nouvelle commande soit reçue. Lorsque le signal d'entrée s'active approximativement, l'état passe d'un état stable à un autre) et des résistances d'abaissement internes.

La broche 7 fournit la tension de commande de grille pour le MOSFET côté HAUT Q1. La broche 1 fournit la tension de commande de grille pour LOW MOSFET Q2.

La commande numérique est à gauche. Ce qui impose que le commun numérique VSS est séparé du commun d'alimentation du moteur COM.

Explication des composants :

Le condensateur C4 et C13 est chargé à travers une diode de VCC à 15V. Le transistor interne "A" commutera la tension du condensateur sur la porte Q1. C4 et C13 ne peut plus charger à partir de VCC tant que Q1 n'est pas éteint.

C'est la charge-décharge de C4 connectée aux bornes de Q1 qui maintient Q1 allumé.

Le condensateur C4 dans notre circuit est de valeur 0.22uF, Il est chargé à 15V à travers la diode D1. Dans notre cas, nous utilisant une diode 1N4007. Pour la commutation à grande vitesse, une diode haute vitesse UF4007. Et pour bloque le retour du courant.

Le condensateur se charge à 15 V entre VB et VS lorsque Q1 est éteint. VS est connecté à la connexion source Q1.

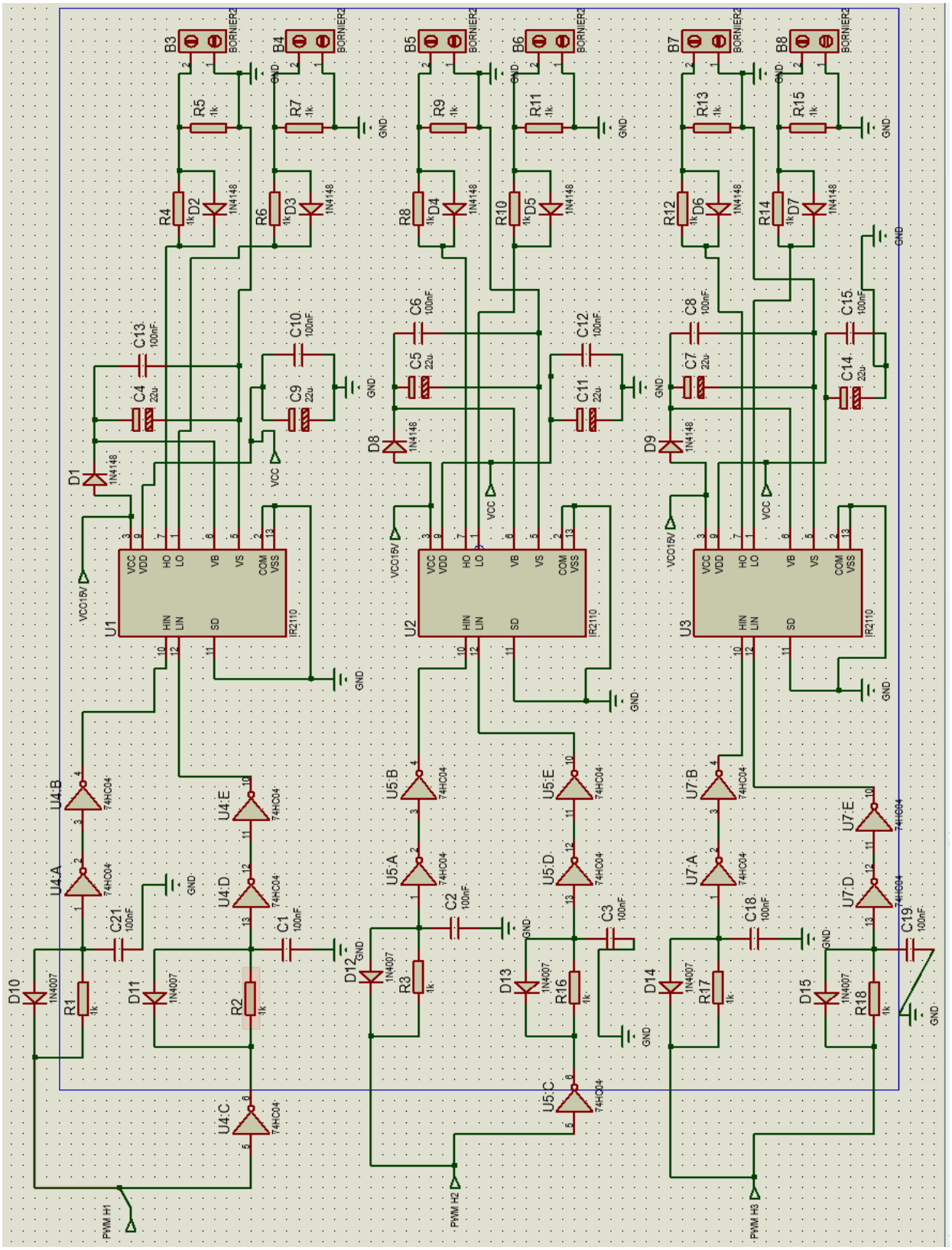
Le condensateur C9 et C10 sont des condensateurs de découplage pour éliminer la composant alternatif et les bruits pour assurer une tension continue de 5V a entré.

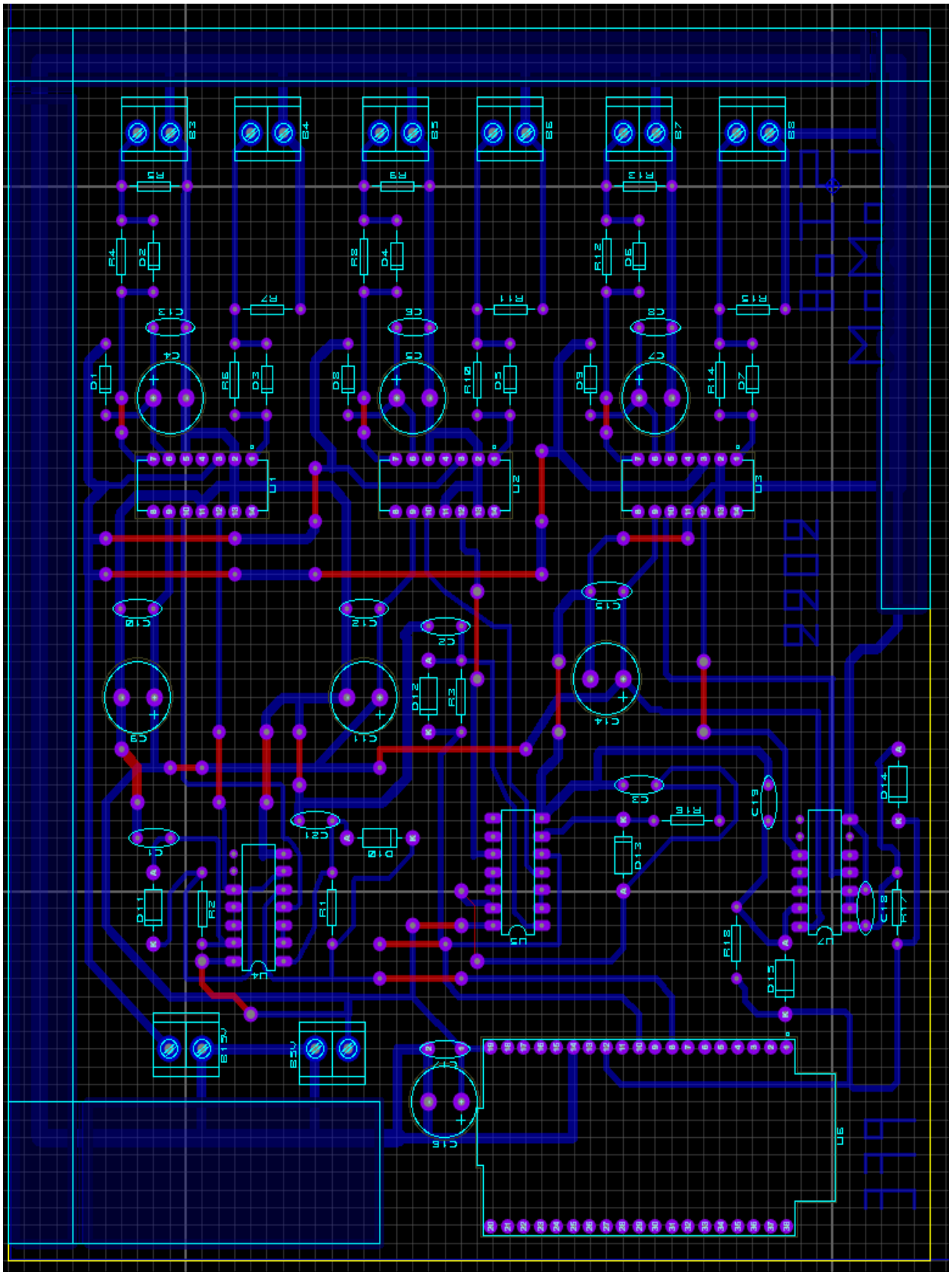
La résistance R5 et R7 sert à bloque les IGBT Q1 et Q2.

Bon en a fini avec notre circuit de pilotage maintenait en vas voir les résultats du Circuit de puissance « l'onduleur ».

Annexe B : Circuits électroniques

Schéma électrique du circuit de pilotage /« ISIS/ARES/PCB »





Vue en 3D du circuit de pilotage/Typon coté piste

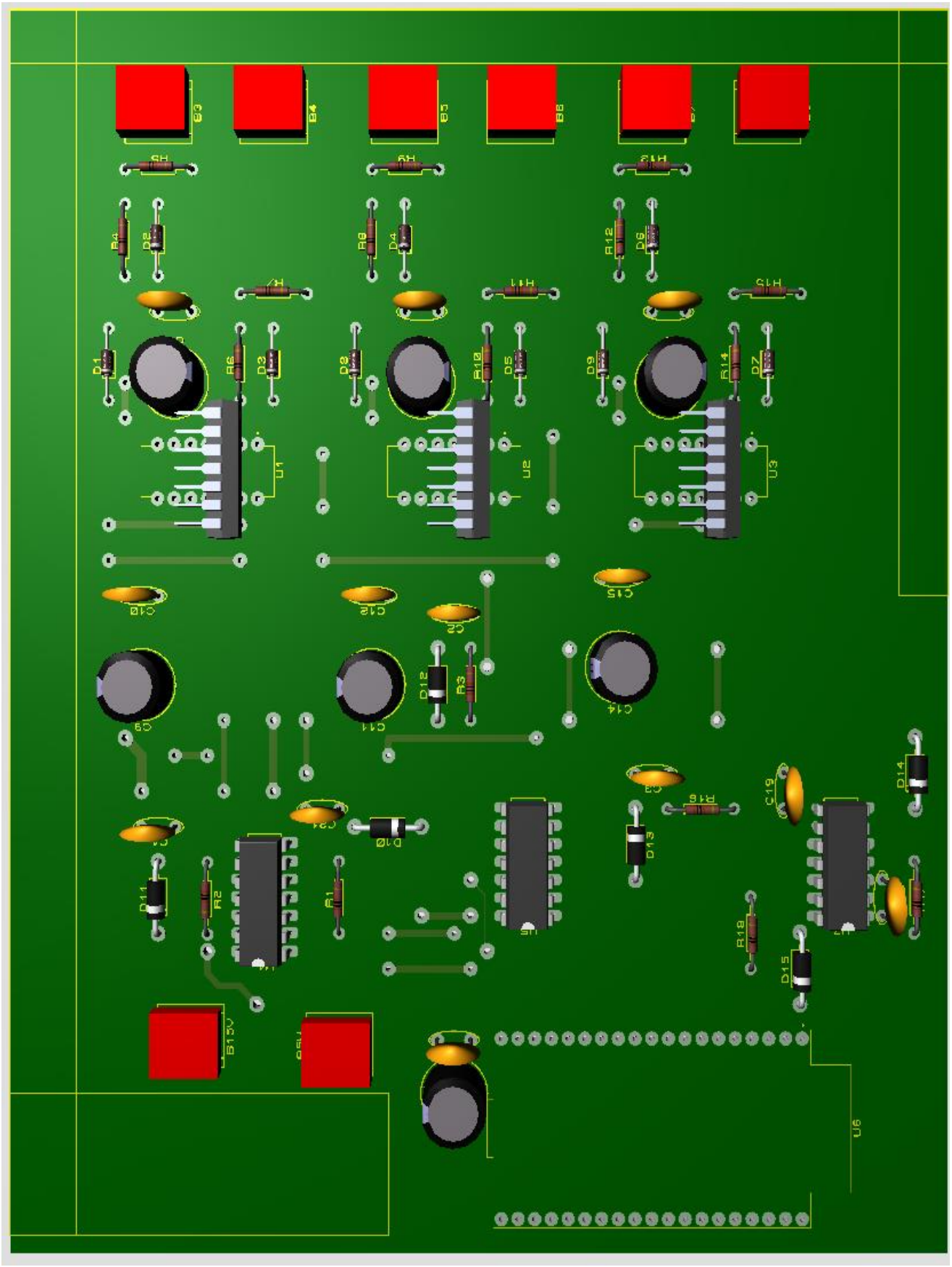
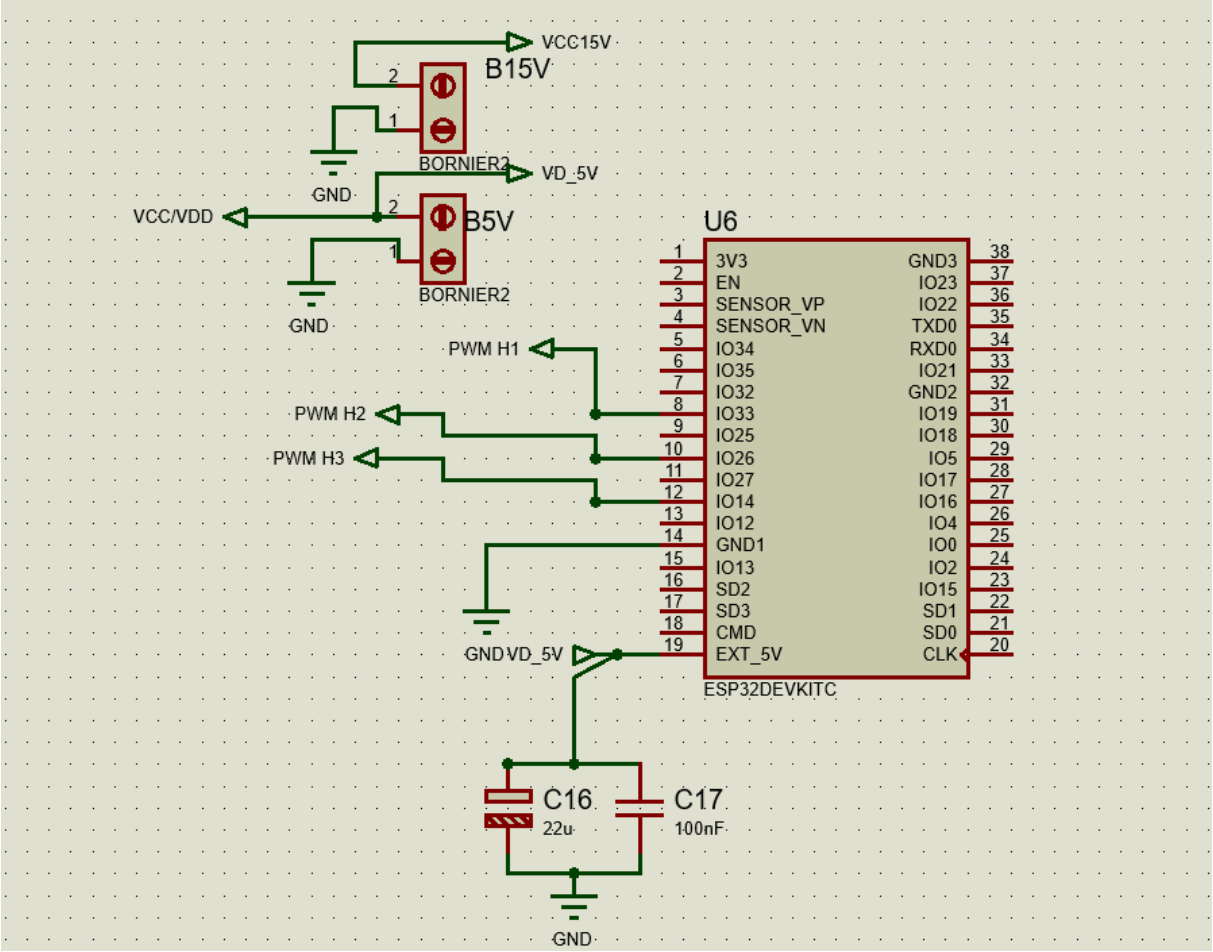


Schéma électrique du Microcontrôleur /« ISIS »



Annexe C

ESP-WROOM-32 PINOUT:

Les broches en vert peuvent être utilisées. Celles qui sont en jaune peuvent être utilisées, mais vous devez faire attention car elles peuvent avoir un comportement inattendu, notamment au démarrage. Les broches en rouge ne sont pas recommandées pour être utilisées comme entrées ou sorties.

Tableau 5 : ESP-WROOM-32 PINOUT

GPIO	Input	Output	Notes
0		**OK**	outputs PWM signal at boot
1	**TX PIN**	**OK**	Debug output at boot
2	**OK**	**OK**	connected to on-board LED
3	**OK**	**RX PIN**	HIGH at boot
4	**OK**	**OK**	
5	**OK**	**OK**	outputs PWM signal at boot
6	**X**	**X**	connected to the integrated SPI flash
7	**X**	**X**	connected to the integrated SPI flash
8	**X**	**X**	connected to the integrated SPI flash
9	**X**	**X**	connected to the integrated SPI flash
10	**X**	**X**	connected to the integrated SPI flash
11	**X**	**X**	connected to the integrated SPI flash
12	**OK**	**OK**	
13	**OK**	**OK**	outputs PWM signal at boot

14	**OK**	**OK**	outputs PWM signal at boot
15	**OK**	**OK**	
16	**OK**	**OK**	
17	**OK**	**OK**	
18	**OK**	**OK**	
19	**OK**	**OK**	
20	**OK**	**OK**	
21	**OK**	**OK**	
22	**OK**	**OK**	
23	**OK**	**OK**	
24	**OK**	**OK**	
25	**OK**	**OK**	
26	**OK**	**OK**	
27	**OK**	**OK**	
28	**OK**	**OK**	
29	**OK**	**OK**	
30	**OK**	**OK**	
31	**OK**	**OK**	
32	**OK**	**OK**	
33	**OK**	**OK**	
34	**OK**		
35	**OK**		Input only

36	**OK**		Input only
37	**OK**		Input only
38	**OK**		Input only

ملخص

نظرًا لأن المحركات غير المتزامنة هي قلب الصناعة ، حيث يتجاوز استهلاكها نصف الطاقة المنتجة على الأرض ، فمن الضروري معرفة كيفية التعامل مع هذه الآلات بدقة ووفقًا لاحتياجات كل ميدان

تحتاج هذه الأنواع من المحركات إلى محولات سرعة خاصة

هناك عدة أنواع من محولات السرعة ، والتي تعتمد أساسًا على قوة المحرك.

مع العلم أن مصداقية هذه الاخيرة زادت بشكل ملحوظ مع تطور إلكترونيات الطاقة

.في حالتنا ، سنركز على محولات السرعة للمحركات غير المتزامنة ثلاثية الطور

في هذا المشروع حاولنا تصميم محول سرعة متغير من خلال الجمع بين العديد من التقنيات للحصول على نتيجة مرضية .

Résumé

Les moteurs asynchrones étant le cœur de l'industrie, avec une consommation dépassant la moitié de l'énergie produite sur terre, il est impératif de savoir manipuler ces machines avec précision et en fonction des besoins de chaque domaine.

Ces types de moteurs ont besoin de variateur de vitesse a base d’onduleur.

Il existe plusieurs types de variateurs de vitesse, qui sont principalement en fonction de la puissance du moteur.

Il faut savoir que la fiabilité de ces derniers a augmenté de façon remarquable

Avec l'évolution de l'électronique de puissance, la fiabilité de ces.

Dans notre cas, nous nous concentrerons sur les variateurs de vitesse pour les moteurs asynchrones triphasés.

Nous avons essayé de concevoir un variateur de vitesse triphasé en combinant plusieurs technologies pour obtenir un résultat satisfaisant.

Abstract

Asynchronous motors being the heart of the industry, with a consumption exceeding half of the energy produced on earth, it is imperative to know how to handle these machines with precision and according to the needs of each field.

These types of motors require inverter-based variable speed drives.

There are several types of inverters, which are mainly based on the power of the motor.

It should be noted that the reliability of these has increased remarkably

With the evolution of power electronics, the reliability of these.

In our case, we will focus on variable speed drives for three-phase asynchronous motors.

We have tried to design a three-phase variable speed drive by combining several technologies to obtain a satisfactory result.

Mot clés

IGBT de puissance, IR2110, Flutter, Commande Scalaire, ESP32, Variateur de vitesse