

ECOLE DES HAUTES ETUDES D'INGENIERIE – OUJDA

Mémoire de Projet de Fin d'Année

FILIERE : Génie des Systèmes Industriels

Régulation de la vitesse d'un MCC en utilisant une plateforme DSP

Réalisé par :

M ABESSI Mohammed M MEZIANE Zakaria

M NESSATI Mohammed M NASSAR Nouh

M SLIMI Zakaria MME EL WALI Soukayna

Encadré par :

M. Mohammed CHAKER

Soutenance le 21/07/ 2025 devant le Jury :

M CHAKER Mohammed

M RAHMANI Driss

M AZZOUZI Tayeb

M TOUILI Samir

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce projet de fin d'année.

Nous remercions tout d'abord notre encadrant, **M. CHAKER**, pour sa disponibilité, ses conseils avisés, son suivi rigoureux et son soutien tout au long de ce travail. Ses remarques constructives nous ont permis de progresser méthodiquement et d'enrichir nos compétences dans les domaines techniques abordés.

Nous remercions également nos collègues de classe et camarades de projet pour leur esprit de collaboration, les échanges enrichissants, le partage de ressources et leur soutien moral.

Nos remerciements s'adressent également aux membres du jury, qui nous font l'honneur de participer à cette soutenance.

À toutes et à tous, merci.

Table des matières

INTRODUCTION GENERALE	9
CHAPITRE 1 : MODELISATION ET SIMULATION DU MCC ET DU HACHEUR 4Q	10
I. MODELISATION PAR BLOC DU MATHWORKS :.....	11
A. MODELISATION DU MOTEUR A COURANT CONTINU (MCC)	11
1. <i>Introduction et principes fondamentaux</i>	11
1.1 Structure de base.....	11
2. <i>Équations fondamentales du moteur CC</i>	11
2.1 Circuit électrique équivalent.....	11
2.2 Force contre-électromotrice.....	12
2.3 Couple électromagnétique	12
3. <i>Équation mécanique du moteur</i>	12
4. <i>Modèle d'état complet.....</i>	13
5. <i>Modèle Simulink du Moteur à Courant Continu (MCC)</i>	13
5.1. partie simulation :	13
5.2. Configuration du moteur.....	14
B. MODELISATION DU HACHEUR 4Q	14
1. <i>Introduction :</i>	14
2. <i>Modèle Simulink du hacheur 4Q.....</i>	15
C. COMMANDE D'UN MOTEUR A COURANT CONTINU VIA UN HACHEUR QUATRE QUADRANTS.....	16
II. MODELISATION A TRAVERS LES FONCTIONS DE TRANSFERT :	17
A. FONCTION DE TRANSFERT DU MCC :	17
1. <i>Partie électrique :</i>	17
2. <i>Partie mécanique :</i>	18
B. FONCTION DE TRANSFERT DU HACHEUR	18
C. COMMANDE EN BOUCLE OUVERTE D'UN MOTEUR A COURANT CONTINU VIA UN HACHEUR QUATRE QUADRANTS	19
D. COMPARAISON DES RESULTATS DES DEUX METHODES.....	20
CHAPITRE 2 : IMPLEMENTATION SUR DSP	21
I. INTRODUCTION	22
II. CARTE DE DEVELOPPEMENT :.....	23
1. <i>Caractéristique de la carte.....</i>	23
2. <i>Processeur TS320F28069M</i>	25
III. LOGICIELS NECESSAIRES :	26
IV. CONFIGURATION DES PERIPHERIQUES DU DSP :	27
1. <i>Configuration du fichier Simulink (.slx) :</i>	27
2. <i>Test du bloc GPIO :.....</i>	28
2.1 GPIO DO :.....	28
2.2 GPIO DI :.....	34
3. <i>Test du bloc ePWM :</i>	35

Remarque :	36
3.1 La fréquence :	36
3.2 Les entrées WA et WB :	36
3.3 Entraînement direct de la valeur du rapport cyclique.....	36
3.4 Les entrées de forçage :.....	37
4. Test des blocs ADC :	37
4.1 Caractéristiques :.....	37
4.2 Test :.....	38
4.3 Résultat :	39
V. CARTE DE PUISSANCE BOOSTXL-DRV8305EVM :	39
1. <i>Introduction</i> :	39
2. <i>Caractéristique</i>	40
3. <i>L'objectif</i>	41
VI. COMMANDE DU MOTEUR A COURANT CONTINUE VIA DSP	42
1. <i>Commande en boucle ouvert</i>	42
1.1 Simulation.....	42
1.2 Application.....	45
1.2.1 Rôle fonctionnel du circuit (avec tachymètre en entrée)	45
1.2.2 Matériels utiliser.....	47
1.2.3 Test	48
2. <i>Commande en Boucle Fermer</i> :	50
2.1 Identification de la fonction de transfert de système :.....	50
2.2 Dimensionnement du correcteur PI parallèle :	51
CHAPITRE 3 : RESULTATS FINAL (H4Q + MCC + CORRECTEUR PI)	54
I. APPLICATION	55
1. <i>Test sans charge</i>	55
2. <i>Test avec charge</i>	56
CONCLUSION	58

Liste des figures

Figure 1: Modèle Simulink du MCC.....	13
Figure 2: Fichier .m des constantes	14
Figure 3: Configuration du moteur.....	14
Figure 4: Modèle Simulink du hacheur 4Q	15
Figure 5: MCC commander par un hacheur 4Q	16
Figure 6: schéma de vitesse du MCC en rad/s	16
Figure 7: Paramètres de la charge (TL=0)	16
Figure 8: schéma de tension et du courant du hacheur 4Q.....	16
Figure 9 : Paramètres de la charge (TL=1)	17
Figure 10:schéma de vitesse du MCC en rad/s (TL=1)	17
Figure 11:Circuit électrique du MCC	17
Figure 12:Modélisation du MCC en boucle ouvert	18
Figure 13: Fonction de Transfert du Hacheur 4Q	18
Figure 14: commande du MCC en boucle ouverte via un Hacheur 4Q.....	19
Figure 15:système de commande du MCC en FCT et BLOC MATHWORKS	19
Figure 16: comparaison de la vitesse du MCC entre les deux systèmes	20
Figure 17: LAUNCHXL-F28069M Board Overview	24
Figure 18: Disposition et affectation des broches du microcontrôleur F28069M sur la carte LAUNCHXL-F28069M	25
Figure 19: Paramètres d'un fichier Simulink (hardware implementation).....	27
Figure 20: Paramètres d'un fichier Simulink (code generation)	28
Figure 21: test du bloc GPIO DO	28
Figure 22: Configuration du bloc GPIO DO	29
Figure 23: diagnostique de l'exécution d'un fichier Simulink	29
Figure 24: Emplacement du fichier.c généré	30
Figure 25: configuration de la cible	30
Figure 26: connexion de la cible	31
Figure 27: chargement du fichier.out	31
Figure 28: emplacement du fichier.out	31
Figure 29: recherche de la variable var	32
Figure 30: l'ajout de la variable var	32
Figure 31: l'exécution du programme	33
Figure 32: Résultat du test GPIO DO	33
Figure 33: Résultat du test GPIO DO	34
Figure 34: Test GPIO DI	34
Figure 35:Test du bloc ePWM.....	35
Figure 36: configuration du bloc ePWM	35

Figure 37: Résultats obtenus des signaux PWM sur oscilloscope	36
Figure 38: test du bloc ePWM	37
Figure 39: test du bloc ADC	38
Figure 40: configuration du bloc ADC.....	38
Figure 41: Les pins de sortie de la carte de puissance BOOSTXL-DRV8305EVM.....	40
Figure 42: les 6 MOSFET de la carte.....	41
Figure 43: fichier .m des constantes.....	42
Figure 44: montage de commande du moteur en boucle ouverte.....	42
Figure 45: configuration du bloc ePWM1.....	43
Figure 46: configuration du bloc ePWM2.....	43
Figure 47: bloc Fcn.....	44
Figure 48: bloc TIME PERIOD.....	44
Figure 49:circuit de protection et de régulation de tension	45
Figure 50: test du bloc ADC (valeur analogique)	47
Figure 51: MCC	47
Figure 52:Montage final en boucle ouverte.....	48
Figure 53: la vitesse en fonction de la tension	49
Figure 54: bloc de l'ADC.....	50
Figure 55: Réponse du système a un échelon	50
Figure 56:valeur de 3t visualiser sur l'oscilloscope	51
Figure 57: Fonction de Transfert en Boucle Ouverte de notre système.....	52
Figure 58: Fonction de Transfert en BF.....	52
Figure 59: Configuration du bloc pi	53
Figure 60: programme final.....	55
Figure 61: valeurs des constantes (vitesse = 100tr/m).....	55
Figure 62: valeurs des constantes (vitesse = 200tr/m).....	56
Figure 63:courant sans charge	57
Figure 64:courant avec charge	57
Figure 65: Montage expérimental du projet réalisé.....	57

Liste des tableaux

Tableau 1: Caractéristiques de la carte LAUNCHXL-F28069M.....	23
Tableau 2: Résultat du test GPIO DI.....	35
Tableau 3: les valeurs du test en BO.....	49

Introduction générale

Dans le cadre de notre formation en Génie des Systèmes Industriels, nous avons mené un projet visant à développer une solution de commande en vitesse d'un moteur à courant continu (MCC) à l'aide d'un hacheur quatre quadrants, piloté par un processeur de signal numérique (DSP). L'objectif principal est de mettre en œuvre un système capable de fonctionner dans les quatre quadrants du plan tension-courant, permettant ainsi le contrôle précis de la vitesse et du sens de rotation du moteur, tout en assurant une capacité de freinage régénératif. Pour cela, nous avons utilisé la carte LAUNCHXL-F28069M comme plateforme DSP, en combinaison avec la carte de puissance BOOSTXL-DRV8305EVM, permettant le pilotage des six MOSFETs nécessaires à la construction du hacheur. Ce projet a permis d'explorer plusieurs aspects essentiels : modélisation, simulation, implantation sur carte embarquée, acquisition de données en temps réel et mise en œuvre d'un correcteur PI pour assurer une régulation efficace. L'ensemble du travail a été mené de façon structurée, de la conception à la validation expérimentale.

Chapitre 1 : Modélisation et Simulation du MCC et du hacheur 4Q

I. Modélisation par bloc du MathWorks :

A. Modélisation du Moteur à Courant Continu (MCC)

1. Introduction et principes fondamentaux

Le moteur à courant continu (CC) est une machine électrique qui convertit l'énergie électrique en énergie mécanique. Son fonctionnement repose sur l'interaction entre un champ magnétique stationnaire et un champ magnétique créé par des courants circulant dans un enroulement mobile.

1.1 Structure de base

Un moteur CC classique comporte deux parties principales :

- Le stator : partie fixe qui crée le champ magnétique principal (par des aimants permanents ou des électroaimants)
- Le rotor (ou armature) : partie mobile constituée de bobinages dans lesquels circule le courant d'alimentation

La commutation du courant dans les différentes sections du bobinage rotorique est assurée par un système collecteur-balais.

2. Équations fondamentales du moteur CC

2.1 Circuit électrique équivalent

Pour un moteur CC, le circuit électrique équivalent simplifié est représenté par l'équation suivante :

$$V = R_a I_a + L_a \frac{dI_a}{dt} + E$$

ou

- R_a : la résistance d'induit
- L_a : l'inductance d'induit
- E : la force contre-électromotrice (proportionnelle à la vitesse)
- V : la tension d'alimentation
- I_a : le courant d'induit

2.2 Force contre-électromotrice

La force contre-électromotrice est proportionnelle à la vitesse de rotation du moteur :

$$E = K_e \omega_m$$

Où

K_e est la constante de f.c.é.m. [V·s/rad]

ω_m est la vitesse angulaire du moteur [rad/s]

2.3 Couple électromagnétique

Le couple électromagnétique développé par le moteur est proportionnel au courant d'induit:

$$T_e = K_t I_a$$

Où :

- T_e est le couple électromagnétique [N·m]
- K_t est la constante de couple [N·m/A]
- I_a est le courant d'induit [A]

Dans un moteur CC, les constantes K_e et K_t sont numériquement égales dans le système SI :

$$K_e = K_t = K$$

3. Équation mécanique du moteur

L'équation mécanique du moteur est donnée par :

$$J \frac{d\omega_m}{dt} + B\omega_m + T_L = T_e$$

Où :

- J est le moment d'inertie total (moteur + charge) [kg·m²]
- B est le coefficient de frottement visqueux [N·m·s/rad]
- T_L est le couple résistant (couple de charge) [N·m]
- T_e est le couple électromagnétique [N·m]

4. Modèle d'état complet

En combinant les équations électriques et mécaniques, on obtient un modèle d'état complet pour le moteur CC :

$$\frac{dI_a}{dt} = -\frac{R_a}{L_a} I_a - \frac{K}{L_a} \omega_m + \frac{V}{L_a}$$

$$\frac{d\omega_m}{dt} = \frac{K}{J} I_a - \frac{B}{J} \omega_m - \frac{T_L}{J}$$

Ce système d'équations différentielles de premier ordre décrit complètement la dynamique du moteur CC.

5. Modèle Simulink du Moteur à Courant Continu (MCC)

5.1. partie simulation :

Schéma Simulink représentant le modèle du MCC seul, avec les blocs pour la tension d'entrée, le courant, la vitesse et le couple résistant.

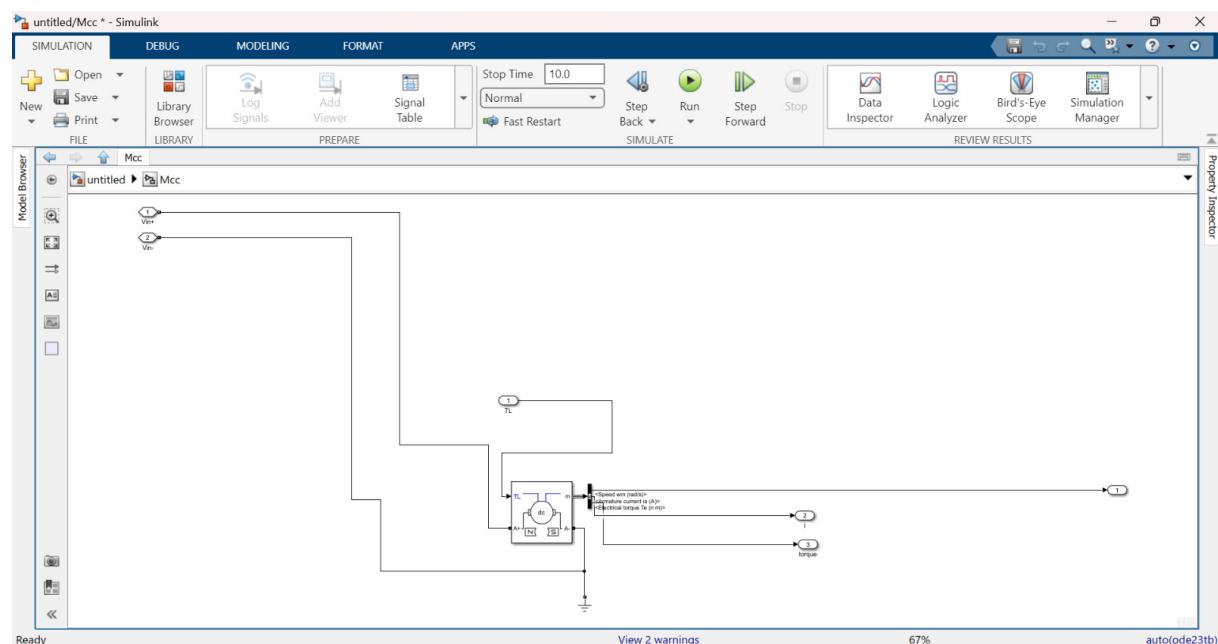


Figure 1: Modèle Simulink du MCC

Avant de passer au montage de la simulation on va déclarer toutes les valeurs des constantes qu'on va utiliser à la simulation de la commande du moteur DC, et voilà un fichier .m qui résume ces valeurs qui sont les mêmes utiliser dans la partie expérimentale qui suivra cette partie de simulation :

```

1      R = 2; % résistance en Ohm
2      L = 10e-3; % inductance en Henri
3      K = 0.1;
4      vdc = 100; % tension d'alimentation en Volt
5      Tmli = 1/2000; % période MLI en seconde
6      J = 2.5e-4; % coefficient d'énergie du MCC
7      fr = 1e-5; % coefficient de frottement visqueux
8      Te = L/R;
9      Tm = R*J/K^2;
10     Tl = 0;
11     Kp = 1;
12     Ti = 0;
13     Th = 1/(2*pi/Tmli);

```

Figure 2: Fichier .m des constantes

5.2. Configuration du moteur

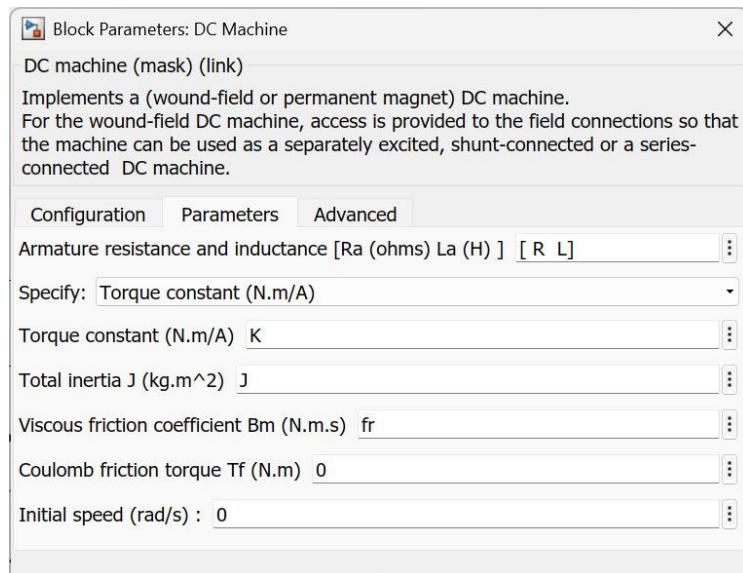


Figure 3: Configuration du moteur

B. Modélisation du Hacheur 4Q

1. Introduction :

Le hacheur quatre quadrants est un convertisseur DC/DC capable de fonctionner dans les quatre quadrants du plan tension-courant, permettant au MCC de fonctionner en moteur ou en générateur, avec rotation dans les deux sens (vitesse positive ou négative). Contrairement à un hacheur à un quadrant, le hacheur 4Q peut :

- Fournir une tension positive ou négative (V_{moy}).
- Gérer un courant positif ou négatif (I).

- Permettre le transfert d'énergie dans les deux directions (du réseau vers le MCC ou du MCC vers le réseau).

Le fonctionnement dans les quatre quadrants est décrit comme suit :

- **Quadrant 1** : Tension positive, courant positif (moteur, rotation avant).
- **Quadrant 2** : Tension positive, courant négatif (générateur, freinage en rotation avant).
- **Quadrant 3** : Tension négative, courant négatif (moteur, rotation arrière).
- **Quadrant 4** : Tension négative, courant positif (générateur, freinage en rotation arrière).

La tension moyenne de sortie du hacheur 4Q est contrôlée par modulation de largeur d'impulsion (MLI) et est donnée par :

$$V_{moy} = (2\alpha - 1) * V_{dc}$$

α est le rapport cyclique pouvant varier entre -1 et 1 (pour permettre une tension négative), et V_{dc} est la tension d'entrée.

2. Modèle Simulink du hacheur 4Q

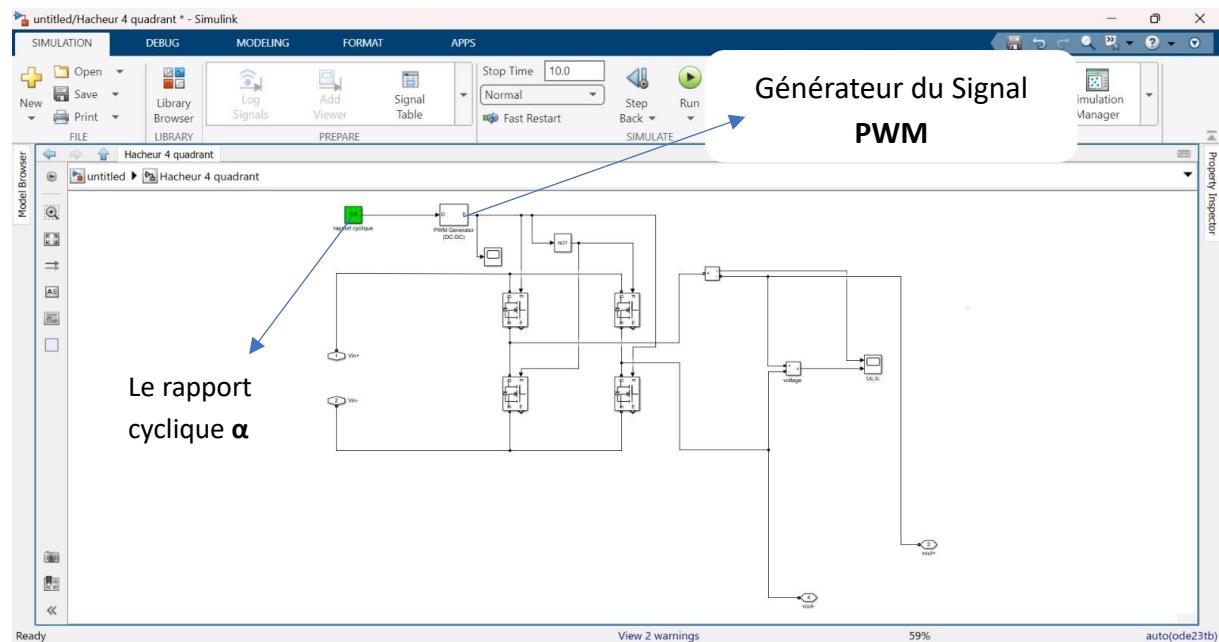


Figure 4: Modèle Simulink du hacheur 4Q

C. Commande d'un moteur à courant continu via un Hacheur quatre quadrants

Pour faire la commande du moteur, on va utiliser le montage suivant (Test avec $\alpha=0.9$) :

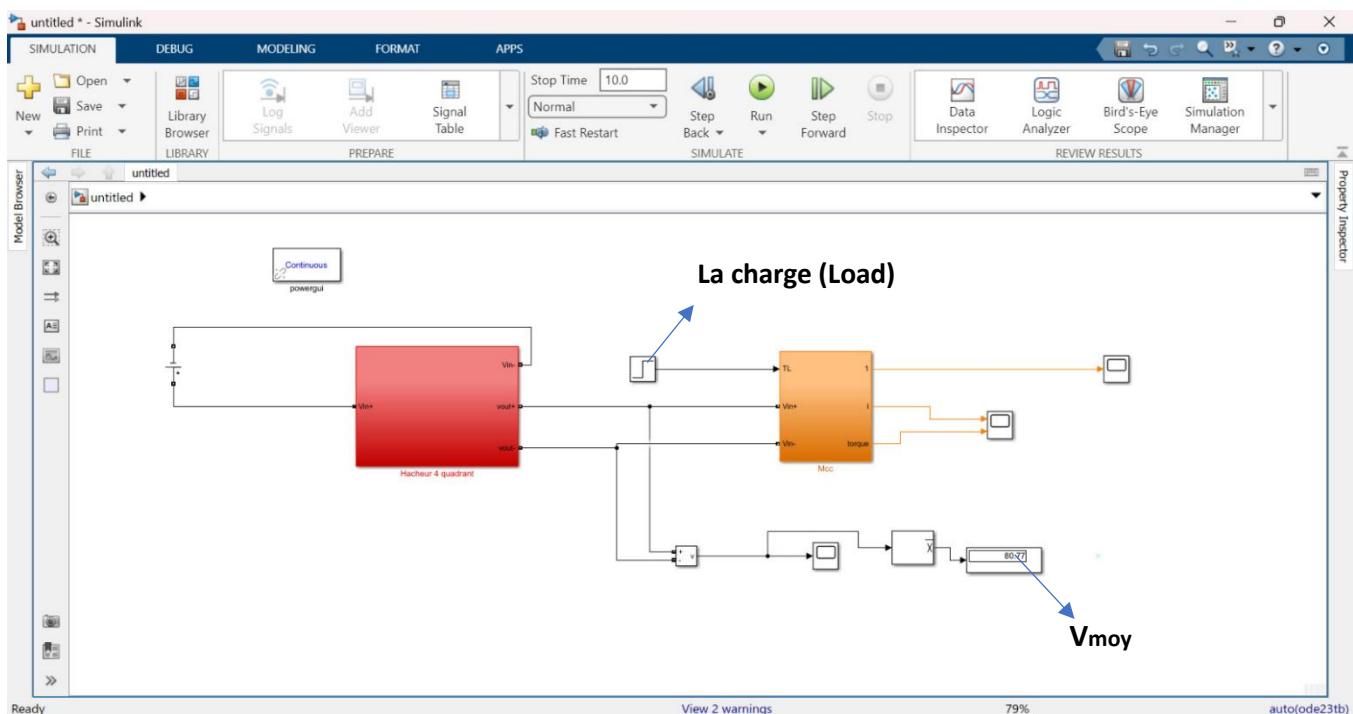


Figure 5: MCC commander par un hacheur 4Q

$$V_{moy} = (2\alpha - 1) * V_{dc} ; V_{dc} = 100V ; \alpha = 0.9$$

$$\text{AN } V_{moy} = (2 * 0.9 - 1) * 100 = 80V$$

- Pour TL (Load) = 0, on obtient les résultats suivants :

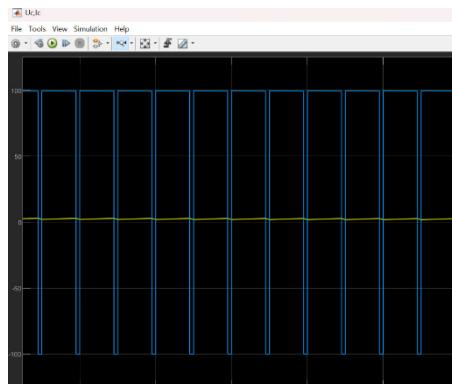


Figure 8: schéma de tension et du courant du hacheur 4Q

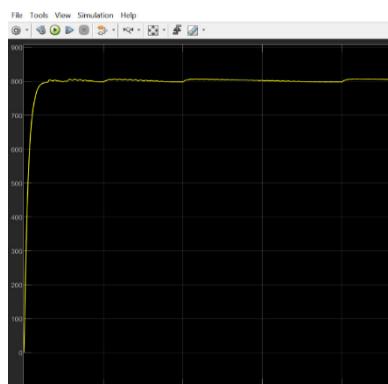


Figure 6: schéma de vitesse du MCC en rad/s

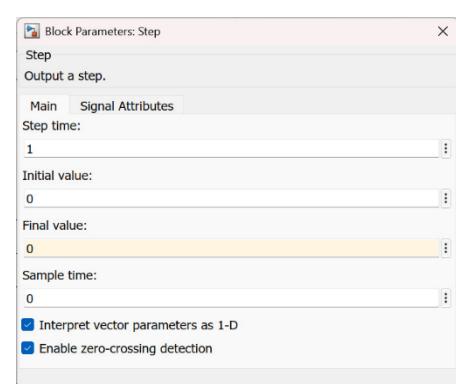


Figure 7: Paramètres de la charge (TL=0)

- Pour **TL (Load) = 1**, on obtient les résultats suivants :

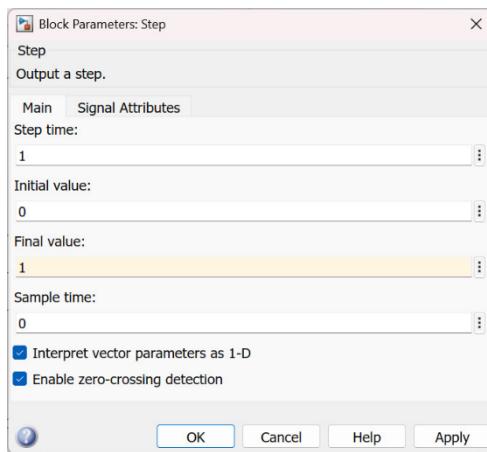


Figure 9 : Paramètres de la charge (TL=1)

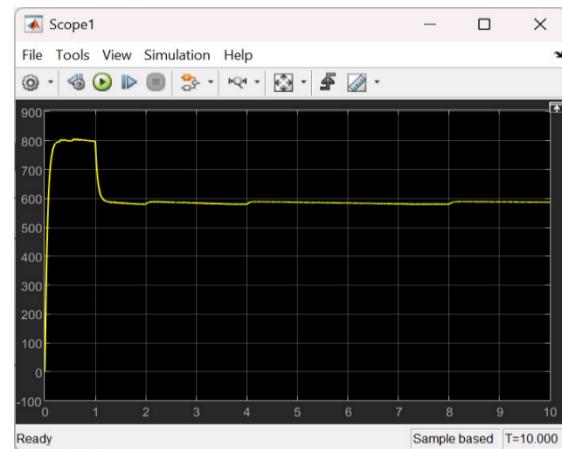


Figure 10: schéma de vitesse du MCC en rad/s (TL=1)

Le hacheur quatre quadrants n'est pas affecté par la charge, donc il n'y a pas de changement pour Vmoy.

II. Modélisation à travers les fonctions de transfert :

A. Fonction de Transfert du MCC :

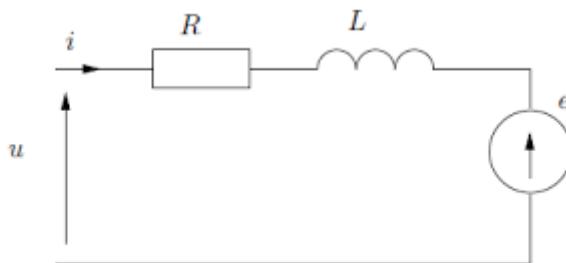


Figure 11:Circuit électrique du MCC

1. Partie électrique :

Circuit électrique et application de la loi des mailles

$$1. \quad u(t) = Ri(t) + L \cdot [di(t)/dt] + e_m(t)$$

Ainsi on lui appliquant la transformée de Laplace

$$2. \quad I(p) = \frac{U(p) - E_m}{L_p + R}$$

La Fonction de transfert de la partie électrique est :

$$3. \quad G_e(p) = \frac{I_m(p)}{U_m(p) - E_m(p)}$$

On aura alors :

$$4. \quad G_e(p) = \frac{1}{R + Lp}$$

2. Partie mécanique :

De manière identique au circuit électrique, on procède ici à l'application de la Relation Fondamentale de la Dynamique RFD:

$$5. \sum F_{EXT} = j \frac{d\omega(t)}{dt}$$

Les deux forces à considérer sont : **la force du couple moteur appliquée par la machine et notée $Cm(t)$** et **la force de couple résistant appliquée par la charge notée $Cr(t)$** .

On aura par ailleurs,

$$6. Cm(t) - Cr(t) = j \frac{d\omega(t)}{dt}$$

- Avec j : le moment d'inertie des parties tournantes.

En remplaçant la force du couple résistant appliquée par la charge $Cr(t)$ par son expression, on aboutit à :

$$7. Cm(t) - f\omega(t) = j \frac{d\omega(t)}{dt}$$

- Avec f le coefficient du frottement visqueux.

En appliquant la transformée de Laplace, on obtient finalement **la Fonction de transfert de la partie mécanique :**

$$8. G_m(p) = \frac{\omega(p)}{Cm(p)} = \frac{\omega(p)}{jP\omega(p) + f\omega(p)} = \frac{1}{f+jP}$$

Maintenant on peut facilement modéliser notre système sous la forme du schéma bloc fonctionnel ci-dessous :

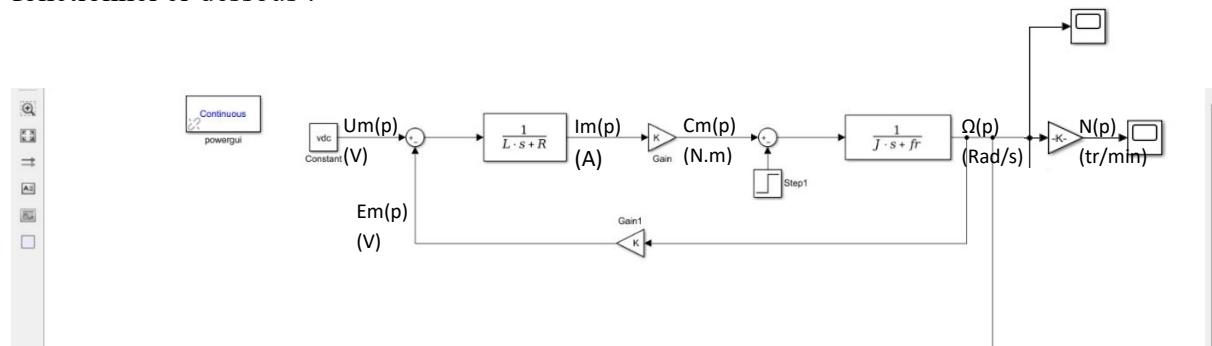


Figure 12: Modélisation du MCC en boucle ouverte

B. Fonction de Transfert du hacheur :



Figure 13: Fonction de Transfert du Hacheur 4Q

C. Commande en boucle ouverte d'un moteur à courant continu via un Hacheur quatre quadrants :

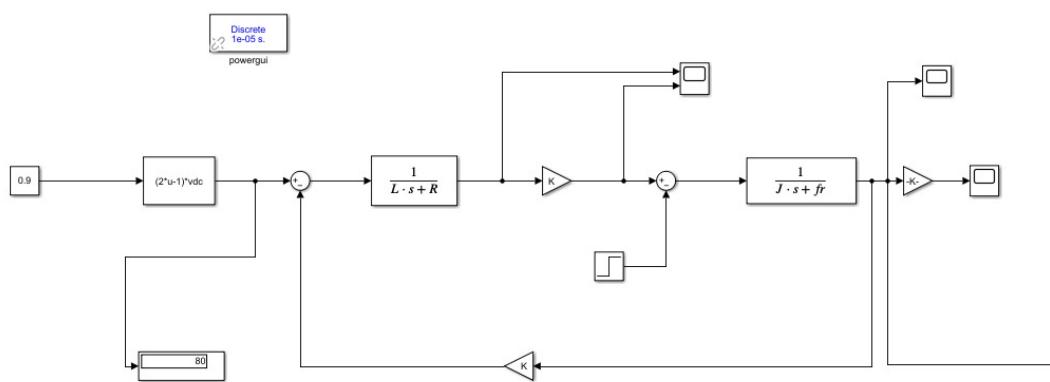


Figure 14: commande du MCC en boucle ouverte via un Hacheur 4Q

Pour valider la précision du modèle basé sur les fonctions de transfert, nous l'avons comparé aux résultats obtenus à partir du système implémenté avec les blocs Simulink de MathWorks.

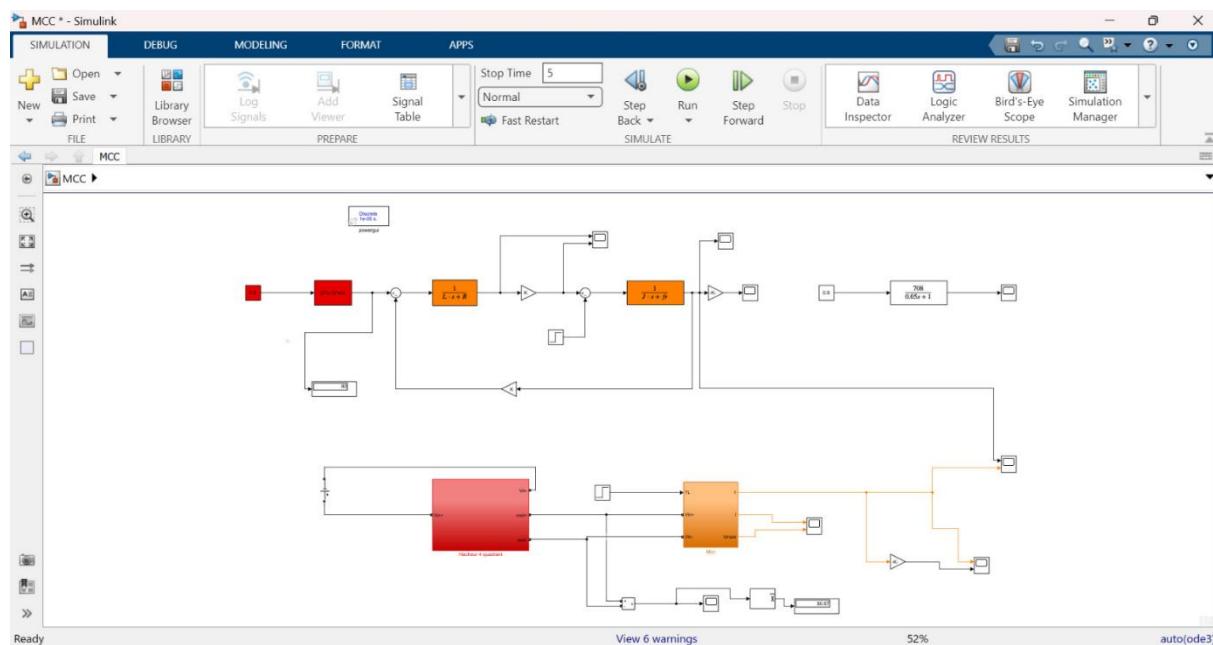


Figure 15:système de commande du MCC en FCT et BLOC MATHWORKS

D. Comparaison des Résultats des Deux Méthodes

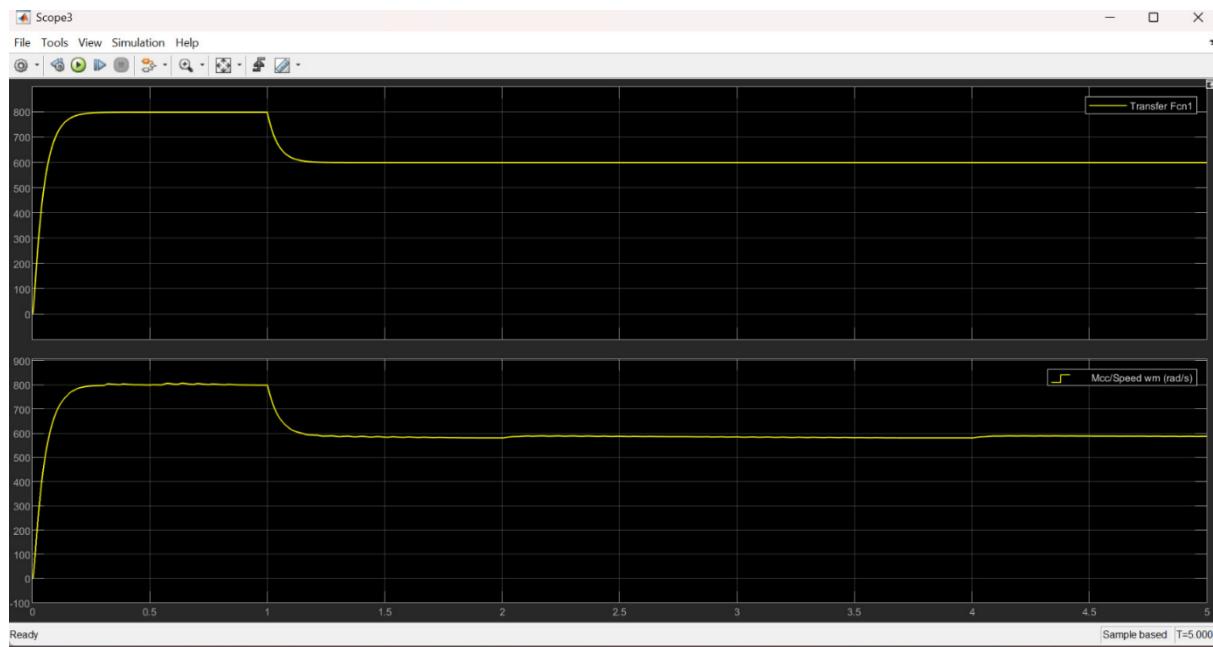


Figure 16: comparaison de la vitesse du MCC entre les deux systèmes

Les simulations effectuées avec les deux approches – le modèle physique basé sur le système implémenté avec les blocs Simulink de MathWorks du MCC et du hacheur 4Q et le modèle basé sur la fonction de transfert ont produit des résultats identiques. Les courbes de vitesse et de courant obtenues à partir des blocs Simulink physiques et celles issues de la fonction de transfert montrent une correspondance parfaite pour les mêmes conditions d’entrée ($V_{dc}=100$, $\alpha=0.9$, $TL=1$, et paramètres du MCC). Cette cohérence confirme la validité des deux modèles et la précision de la linéarisation utilisée pour la fonction de transfert. Les différences négligeables, si présentes, peuvent être attribuées à des approximations numériques dans Simulink.

Chapitre 2 : Implémentation sur DSP

I. Introduction

Dans le cadre de la simulation et du contrôle des systèmes électromécaniques, tels que le moteur à courant continu (MCC) alimenté par un hacheur quatre quadrants, l'utilisation d'une carte à processeur de signal numérique (DSP) joue un rôle central. Une carte DSP est un dispositif électronique spécialisé conçu pour exécuter des calculs complexes en temps réel, particulièrement adaptés au traitement des signaux numériques dans les applications de contrôle-commande. Grâce à leur architecture optimisée, les DSP offrent une grande rapidité d'exécution pour les algorithmes de régulation, tels que les boucles PID, PI et PD, les modulations de largeur d'impulsion (MLI), ou les transformations de signaux, tout en garantissant une précision élevée.

Dans ce projet, la carte DSP est envisagée comme une plateforme pour implémenter les algorithmes de commande simulés sous MATLAB/Simulink, notamment pour la régulation de la vitesse du MCC via le hacheur 4Q. Les cartes DSP, comme celles de Texas Instruments (par exemple, la série C2000), intègrent des périphériques avancés : convertisseurs analogique-numérique (CAN), générateurs de MLI, et interfaces de communication (UART, CAN, SPI). Ces fonctionnalités permettent une interaction efficace avec les capteurs (courant, vitesse) et les actionneurs (hacheur) du système réel. En outre, la programmation des DSP via des environnements comme Code Composer Studio ou MATLAB Embedded Coder facilite le passage de la simulation à l'implémentation pratique.

L'introduction d'une carte DSP dans ce projet vise à valider expérimentalement les résultats obtenus par simulation, en assurant une commande robuste et adaptable aux conditions réelles. Cette approche illustre l'importance des DSP dans la transition entre la modélisation théorique et les applications industrielles, offrant une solution performante pour le contrôle des systèmes dynamiques complexes.

II. Carte de développement :

1. Caractéristique de la carte

La carte DSP utilisée dans notre projet est « LAUNCHXL-F28069M C2000 LaunchPad », c'est une carte développée par la société Texas-instrument. Cette carte a pour caractéristiques techniques :

Référence	LAUNCHXL-F28069M
Fabricant	Texas Instruments
Catégorie	Carte d'évaluation
Architecture	TI C2000
Série	C2000 Picolo
Technologie	32-bit MCU
Type d'outils	Kit d'évaluation
Tension de fonctionnement	3.3V-5V
Tension de fonctionnement minimale	3.3 V
Tension de fonctionnement maximale	5 V

Tableau 1: Caractéristiques de la carte LAUNCHXL-F28069M

Le LaunchPad C2000 Piccolo, modèle LAUNCHXL-F28069M, est une carte de développement complète et à faible coût pour les microcontrôleurs Piccolo F2806x de Texas Instruments et la technologie InstaSPIN. Le kit LAUNCHXL-F28069M comprend tout le matériel et les logiciels nécessaires pour développer des applications basées sur le microprocesseur F2806x.

Le LaunchPad est basé sur le composant F28069M, qui est le plus complet de la gamme, et permet facilement aux utilisateurs de migrer vers des versions moins coûteuses des F2806x une fois les besoins du projet définis. Il propose un outil d'émulation JTAG intégré, permettant une interface directe avec un PC pour faciliter la programmation, le débogage et l'évaluation. En plus de l'émulation JTAG, l'interface USB fournit une connexion série UART (Universal Asynchronous Receiver/Transmitter) entre le dispositif F2806x et l'ordinateur hôte.

Les fonctionnalités du LaunchPad LAUNCHXL-F28069M C2000 comprennent :

- Interface de programmation et de débogage USB via une sonde de débogage isolée XDS100v2 à haute vitesse, avec connexion USB/UART
- Composant F28069M complet permettant une migration facile vers des versions plus économiques
- Deux LED utilisateur
- Bouton poussoir de réinitialisation du dispositif
- Broches du dispositif facilement accessibles pour le débogage ou pour y connecter des cartes d'extension personnalisées
- Bibliothèque InstaSPIN en mémoire ROM, permettant l'implémentation des solutions InstaSPIN-MOTION et InstaSPIN-FOC
- Deux interfaces codeur en quadrature 5 V
- Interface CAN avec transceiver intégré
- Interrupteurs de sélection du mode de démarrage

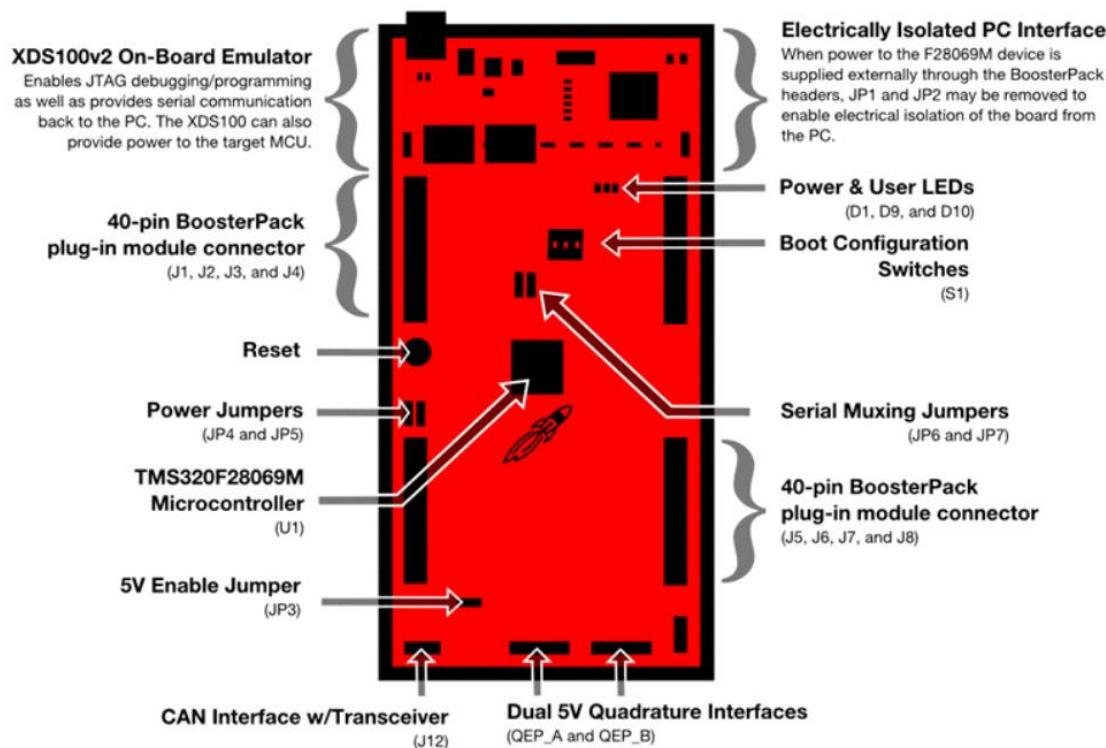


Figure 17: LAUNCHXL-F28069M Board Overview

Afin de mieux comprendre l'interconnexion possible avec des périphériques externes et d'exploiter pleinement les capacités de la carte LAUNCHXL-F28069M, il est essentiel d'avoir une vue d'ensemble des broches du microcontrôleur F28069M ainsi que de leurs fonctionnalités.

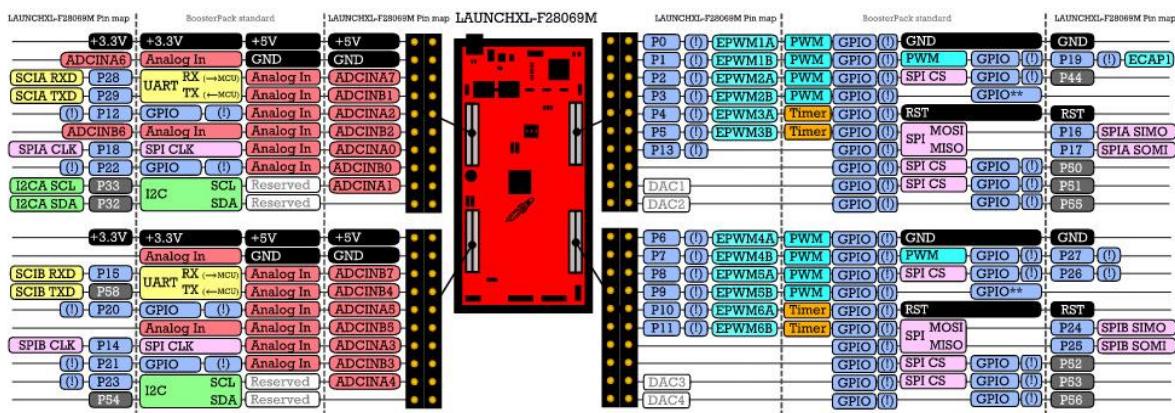


Figure 18: Disposition et affectation des broches du microcontrôleur F28069M sur la carte LAUNCHXL-F28069M

La section suivante présente les broches exploitées dans le projet, accompagnées de leur rôle et de leur configuration.

- **EPWM (Enhanced Pulse Width Modulation)** : utilisées pour la génération de signaux de commande, notamment dans les applications de pilotage moteur.
- **ADC (Analog-to-Digital Converter)** : exploitées pour la conversion de signaux analogiques en données numériques, par exemple pour la lecture de capteurs.
- **GPIO (General Purpose Input/Output)** : utilisées comme entrées ou sorties numériques polyvalentes, permettant l'échange de signaux simples avec des périphériques externes.

2. Processeur TS320F28069M

TS320F28069M est un microcontrôleur inclus dans « InstaSPIN-Motion Librairie ». Ce processeur intègre un cœur 32 bits fonctionnant jusqu'à 90 MHz, ainsi qu'une unité FPU (Floating Point Unit) facilitant les calculs numériques rapides et précis.

Le TMS320F28069M se distingue également par l'intégration en ROM de la bibliothèque InstaSPIN™, qui permet la mise en œuvre simplifiée d'algorithmes avancés de commande moteur tels que InstaSPIN-FOC et InstaSPIN-MOTION. Il dispose de nombreux périphériques intégrés, notamment des convertisseurs analogique-numérique (ADC) haute résolution, des modules de modulation de largeur d'impulsion (ePWM), des interfaces de communication série (I2C, SPI, SCI, CAN), ainsi que des broches GPIO configurables.

Ce microcontrôleur dispose des caractéristiques suivantes :

- 14 canaux PWM, dont 8 canaux PWM haute résolution
- 16 canaux ADC (convertisseurs analogique-numérique)
- 0 canal DAC
- 45 broches GPIO (entrées/sorties numériques)
- 2 canaux QEP (encodeur en quadrature)
- Mémoire Flash jusqu'à 256 Ko
- Mémoire RAM jusqu'à 100 Ko
- Mémoire ROM OTP (One-Time Programmable) de 2 Ko
- Fréquence d'horloge allant jusqu'à 90 MHz
- Architecture Harvard pour la séparation des bus instructions et données
- Unité de calcul en virgule flottante (FPU) intégrée
- Capacité à exécuter du code indépendamment du cœur CPU principal

III. Logiciels nécessaires :

- MATLAB
- Code Composer Studio version 11.0.0
- ControlSUITE
- Drivers (texasinstrumentsc2000soc, texasinstrumentsc2000, C2000Ware_5.04.00.00)

Le F28069M LaunchPad est pris en charge à la fois dans Code Composer Studio et Energia. Cela dépend de notre préférence d'outils, nous pouvons installer l'un ou l'autre ou les deux. Mais dans notre cas, nous avons choisi d'installer le CCS (CCSv11.0.0) avec le controlSUITE3.4.9.

Code Composer Studio

Code Composer Studio (CCS) est un environnement de développement intégré (IDE) développé par Texas Instruments, spécialement conçu pour programmer, déboguer et compiler des microcontrôleurs et processeurs embarqués de la famille TI, tels que les MSP430, C2000, et les processeurs ARM Cortex.

CCS offre une interface complète regroupant un éditeur de code, un compilateur, un débogueur et des outils d'analyse. Grâce à son support natif des architectures TI, il facilite la création d'applications embarquées efficaces et optimisées.

Les principales fonctionnalités de Code Composer Studio incluent :

- Édition et gestion avancée du code source en C, C++ et assembleur
- Compilation optimisée pour les microcontrôleurs TI
- Débogage en temps réel via connexion JTAG/SWD

- Analyse des performances et gestion de la consommation d'énergie
- Intégration avec des bibliothèques et outils TI comme InstaSPIN ou MotorWare

IV. Configuration des Périphériques du DSP :

1. Configuration du fichier Simulink (.slx) :

Pour toutes les différentes applications qui viennent, on doit toujours faire cette configuration du fichier.slx d'application.

En cliquant sur le bouton configuration  situé en haut de la fenêtre du fichier.slx la fenêtre suivante apparaîsse :

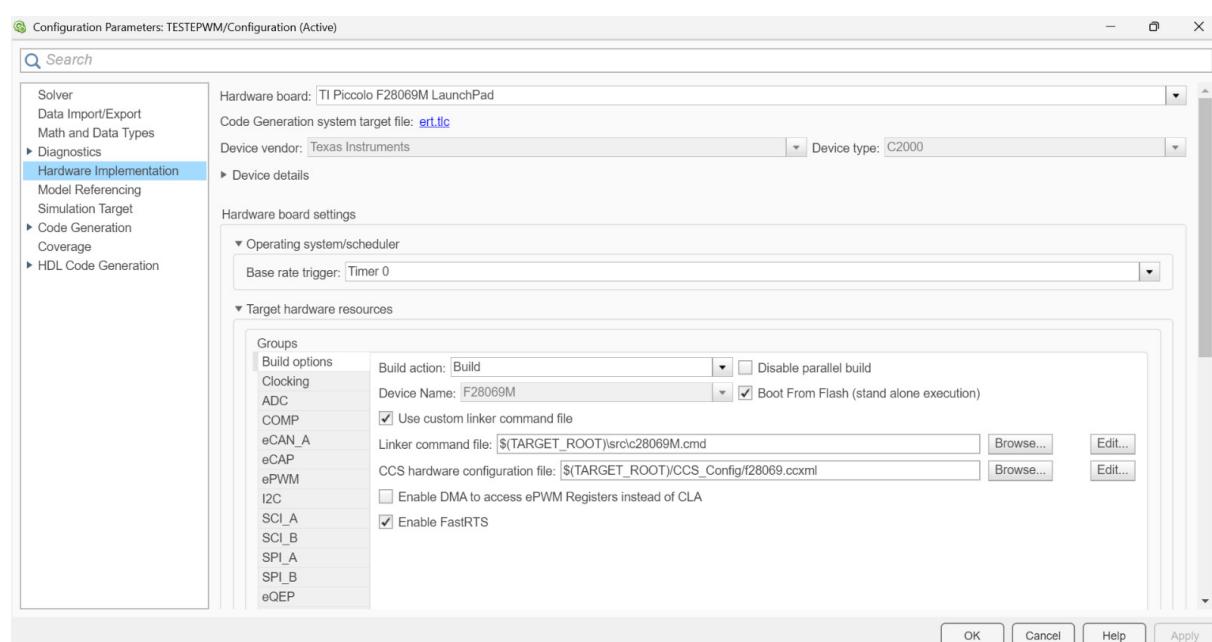


Figure 19: Paramètres d'un fichier Simulink (hardware implementation)

On clique sur « hardware implémentation » et on remplit les données comme elles sont fixées dans la figure 19.

On passe au « code génération » et on remplit les données comme elles sont tapées sur la figure suivante :

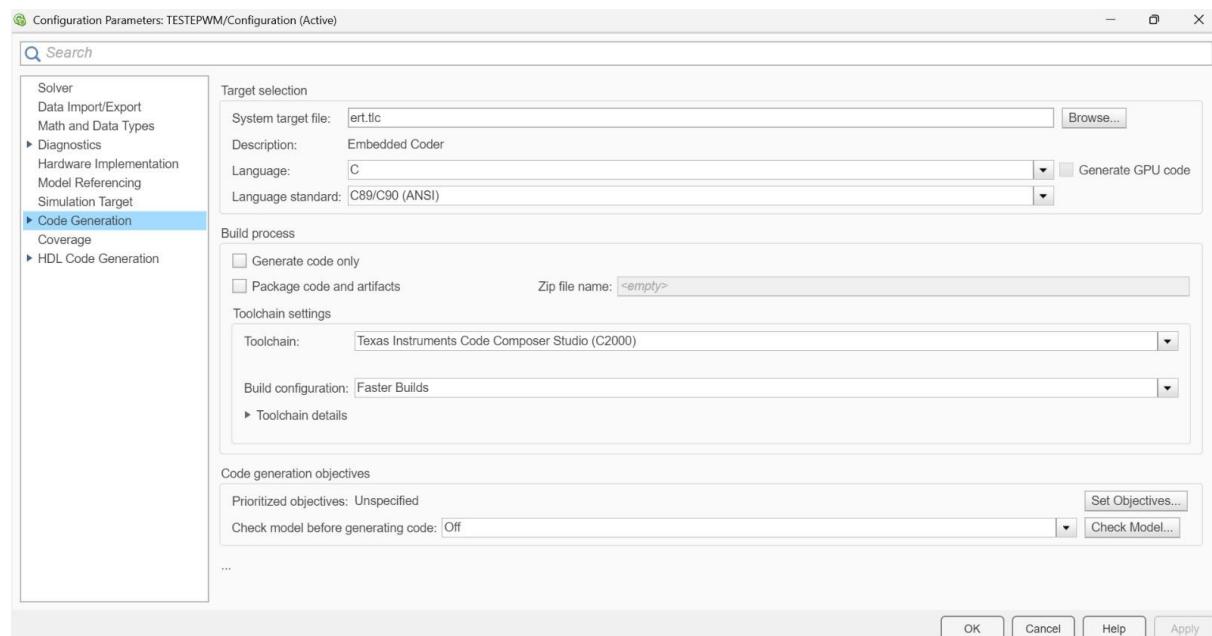


Figure 20: Paramètres d'un fichier Simulink (code generation)

2. Test du bloc GPIO :

Ce test a pour objectif d'apprendre à configurer les blocs GPIO (Processor General-Purpose Input/Output).

2.1 GPIO DO :

Dans ce test on va configurer un GPIO comme sortie (DO : Digital Output) qui va générer la valeur de la constante ‘VAR’ et qui sera afficher à l'aide d'un oscilloscope.

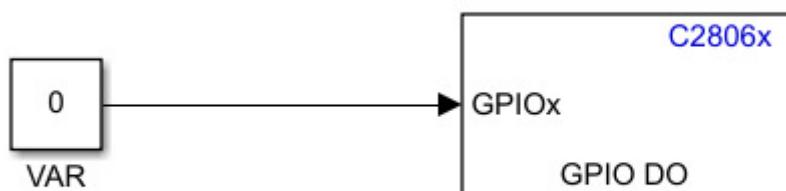


Figure 21: test du bloc GPIO DO

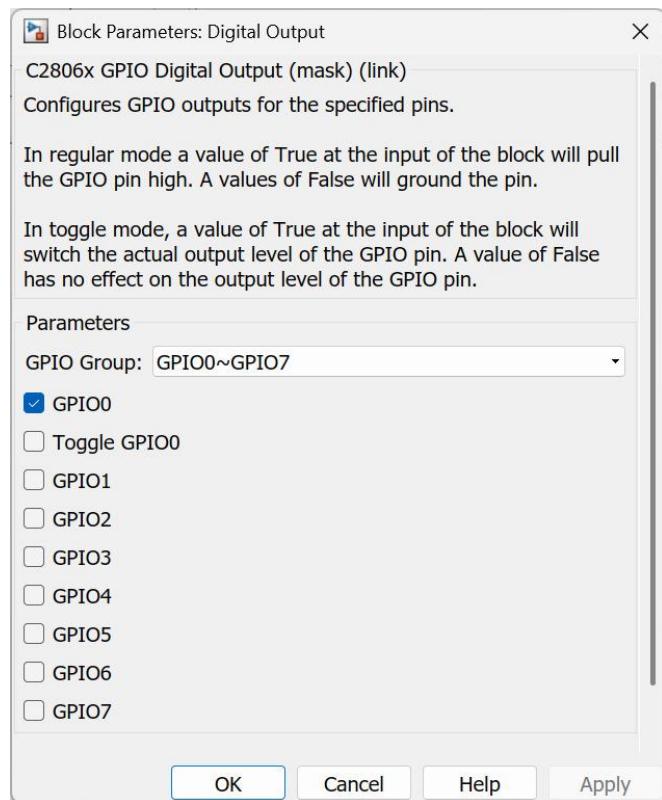


Figure 22: Configuration du bloc GPIO DO

Après on clique sur le bouton déployer  situer en haut de la fenêtre du fichier simulink. On peut suivre l'enchaînement des étapes en cliquant sur « voir diagnostique » situé en bas de la fenêtre :

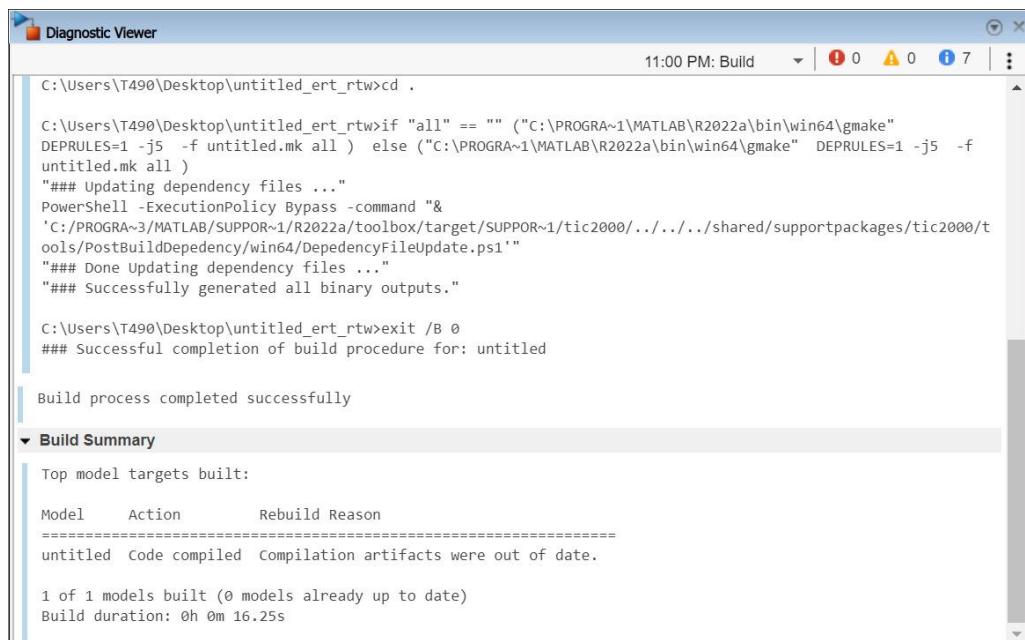


Figure 23: diagnostique de l'exécution d'un fichier Simulink

Après la terminaison de toutes les étapes, elle sort la fenêtre (figure 24) qui nous informe que le code est bien généré et qui nous indique l'emplacement du fichier.c :

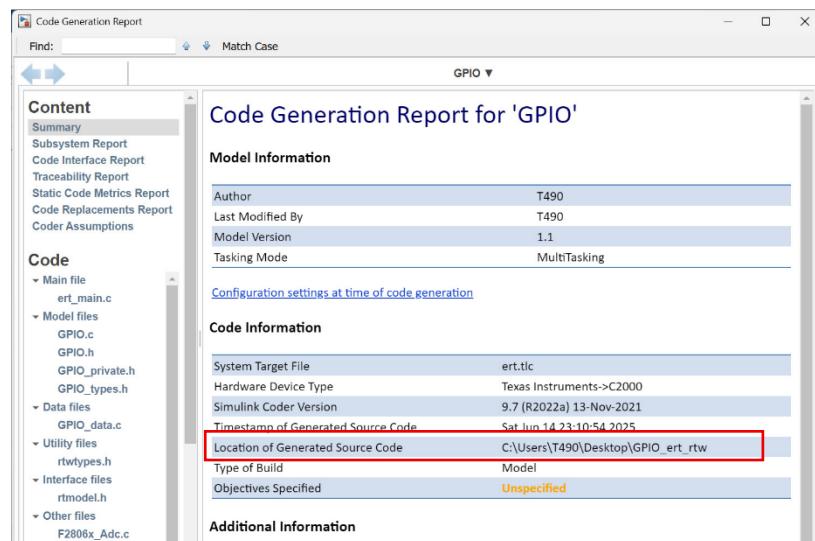


Figure 24: Emplacement du fichier.c généré

Passant à CCS :

Après l'ouverture du CCS 11.0.0 on clique double clic sur « user defined », par la suite en clique avec bouton droit sur « F28029M.ccxm », ensuite on clique sur « lanch selected configuration » :

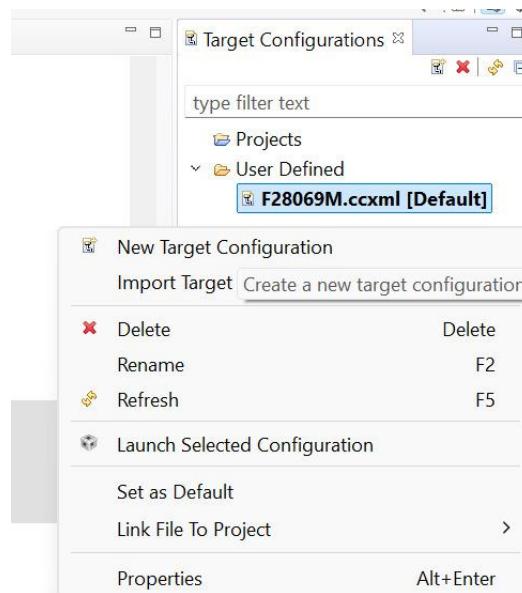


Figure 25: configuration de la cible

Après on connecte notre carte DSP en cliquant avec bouton droit sur « Texas instrument.. », ensuite on clique sur « connect target » :

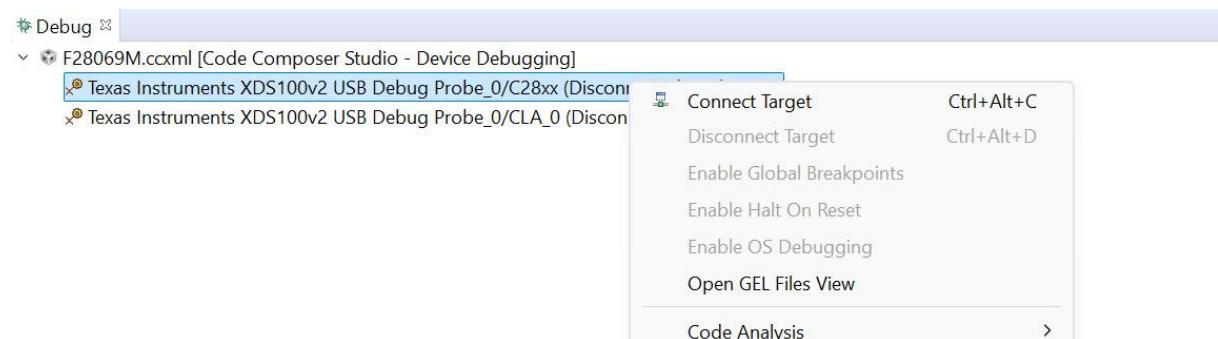


Figure 26: connexion de la cible

Par la suite on fait ‘ Load ‘ à notre fichier.out :



Figure 27: chargement du fichier.out

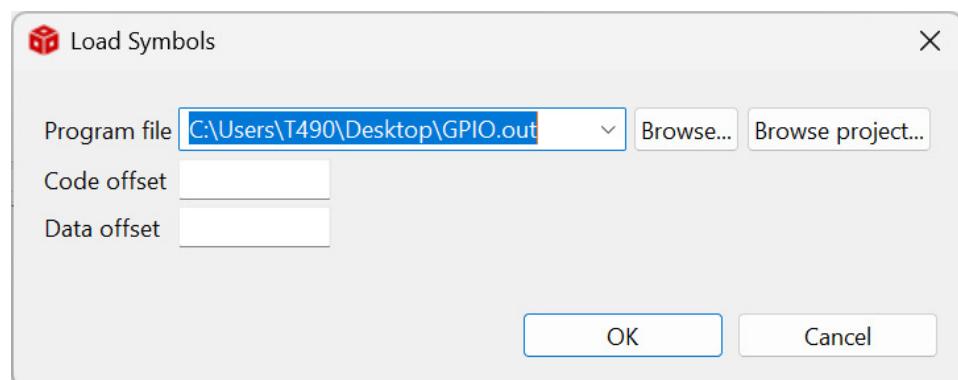


Figure 28: emplacement du fichier.out

Si on a besoin du fichier.c (comme le cas pour ce test) on peut l'ouvrir en cliquant sur « file » → « open file » → emplacement du fichier → ouvrir.

Dans notre cas on va utiliser le fichier.c pour sortir la variable correspondante à notre constante « var » dans la fenêtre expression dans CCS :

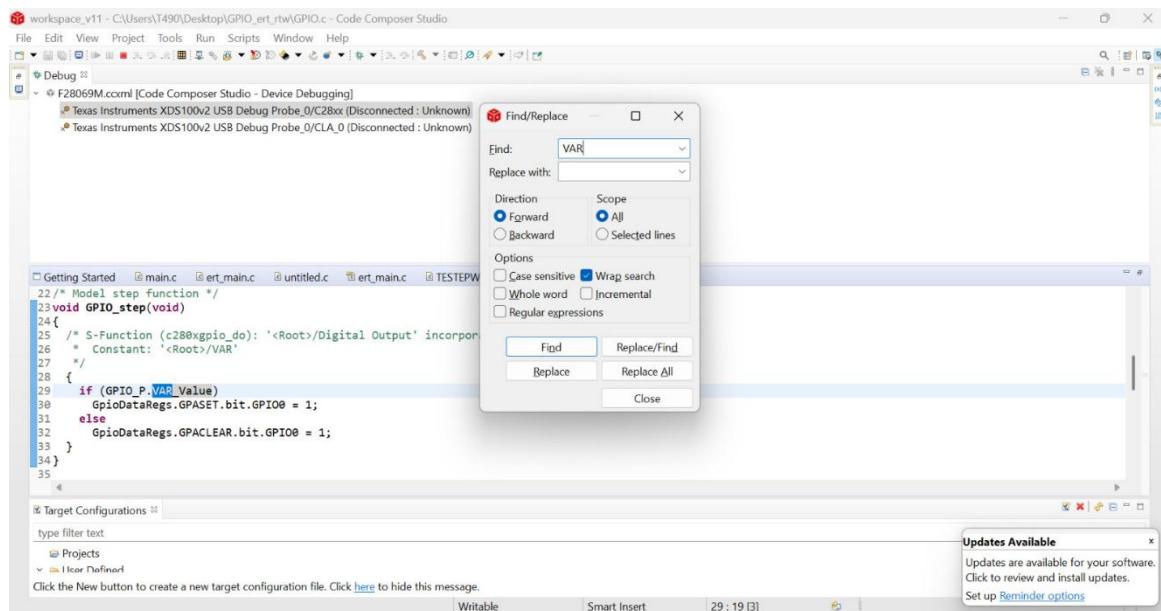


Figure 29: recherche de la variable var

Pour l'ajouter à nos expressions on la sélectionne et on clique sur bouton droit, ensuite on clique sur « Add Watch Expression » :

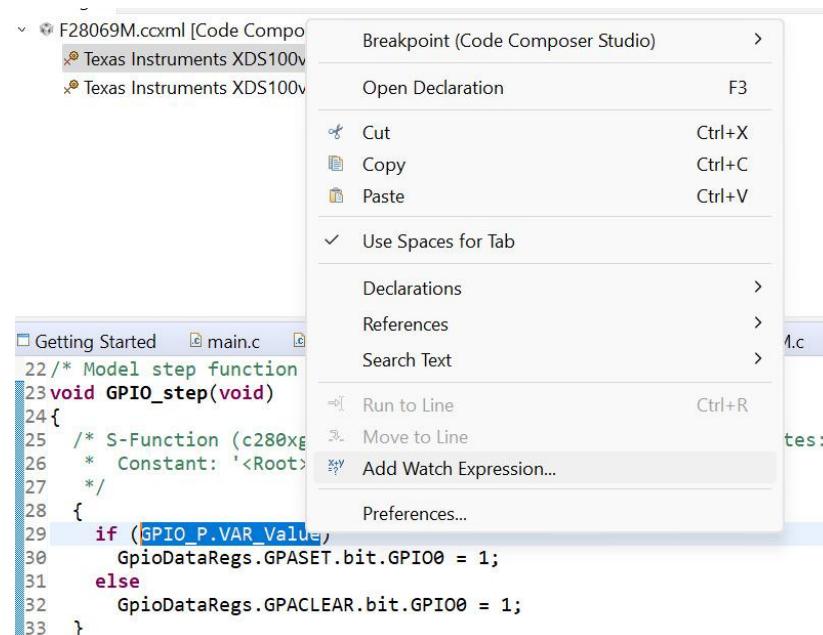


Figure 30: l'ajout de la variable var

Par la suite on revient à « debug » et on clique sur « Run »

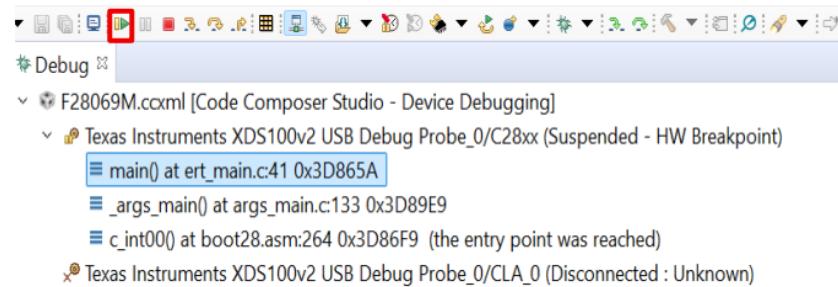


Figure 31: l'exécution du programme

Résultat du test :

- Pour VAR=0 :

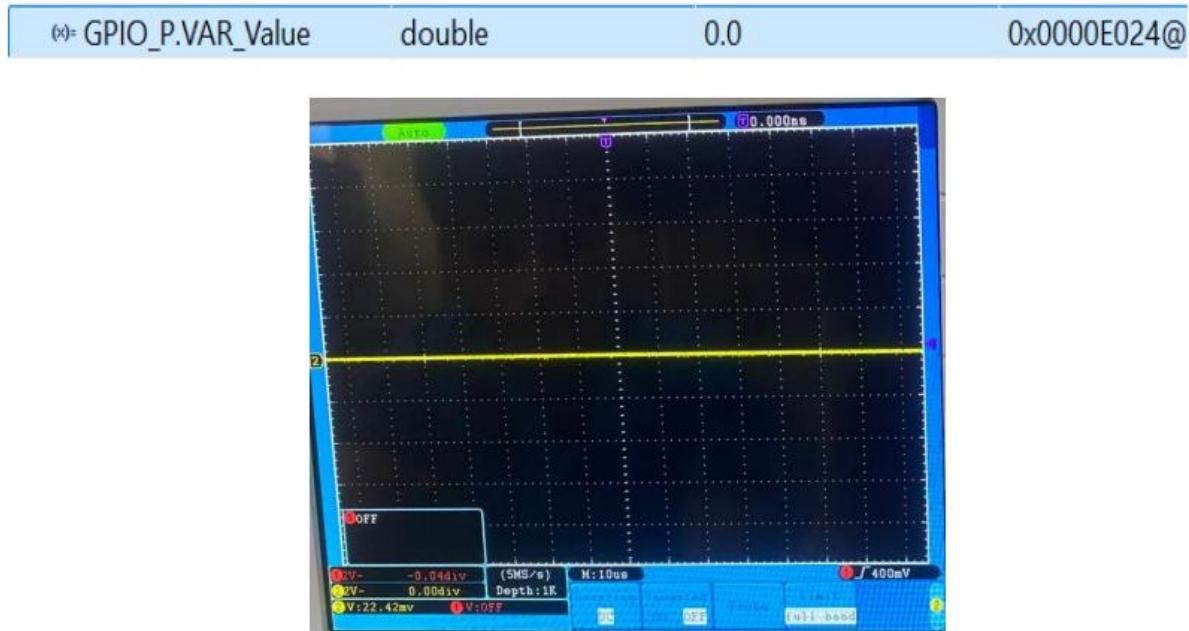


Figure 32: Résultat du test GPIO DO

- Pour VAR=2(DO=3.3V)

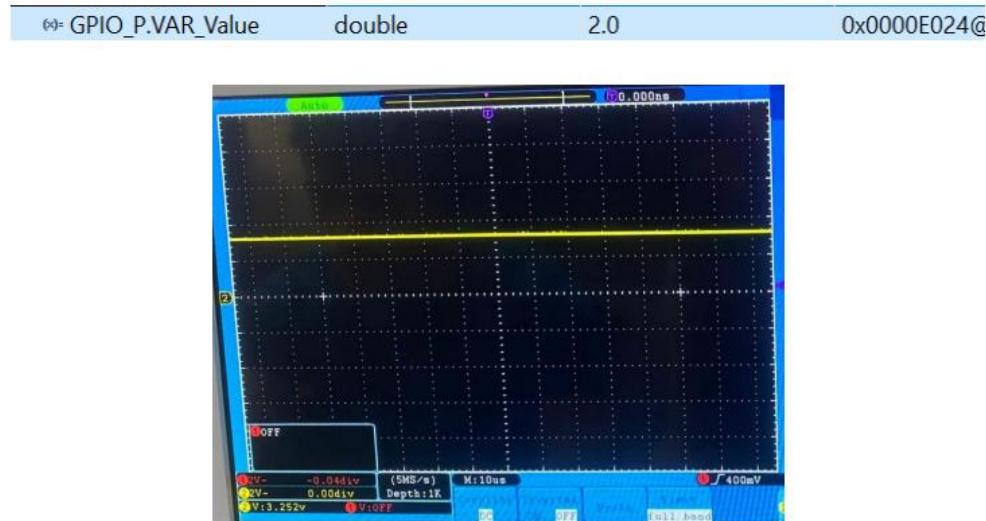


Figure 33: Résultat du test GPIO DO

En changeant la valeur de ‘test2’ d’après CCS dans la fenêtre ‘Expressions’ on conclut que :

Pour VAR = 0 le signal de sortie vaut zéro

Pour VAR ≠ 0 (positive ou négative) le signal de sortie vaut 3.3V.

2.2 GPIO DI :

Pour faire ce test on va faire une boucle; faire connecter le pin du GPIO DI avec l’un des pins suivants (1, 22, 41, 62) et le générer à l'aide d'un GPIO DO.



Figure 34: Test GPIO DI

Et on a les résultats suivants :

Pins utilisés	Signal d'entrée	Signal de sortie
22 / 62	0V	0V
1 / 41	3.3V	3.3V
Pin GPIO non connecté	-	0V

Tableau 2: Résultat du test GPIO DI

3. Test du bloc ePWM :

Ce test a pour objectif d'apprendre à configurer les blocs ePWM.

Dans ce test on va configurer un bloc ePWM pour générer un signal modulé en largeur d'impulsion qui sera affiché à l'aide d'un oscilloscope.

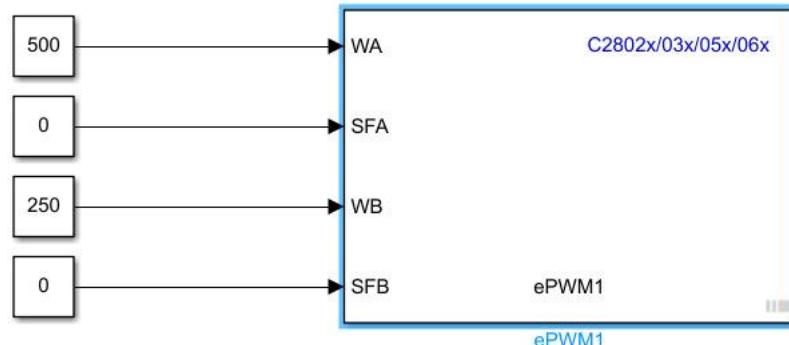


Figure 35:Test du bloc ePWM

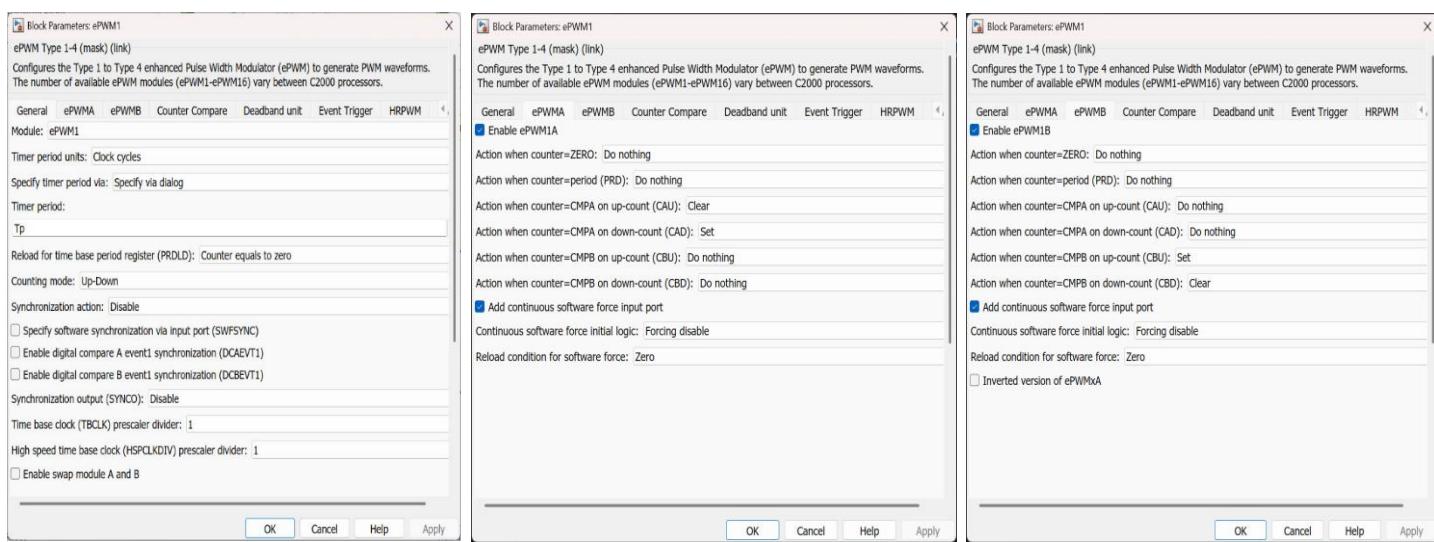


Figure 36: configuration du bloc ePWM

À la fin du paramétrage enregistrer le fichier .slx et puis déployer le programme sur le matériel, passer au CCS et compiler le fichier (.out) (run) et on a les résultats suivants :

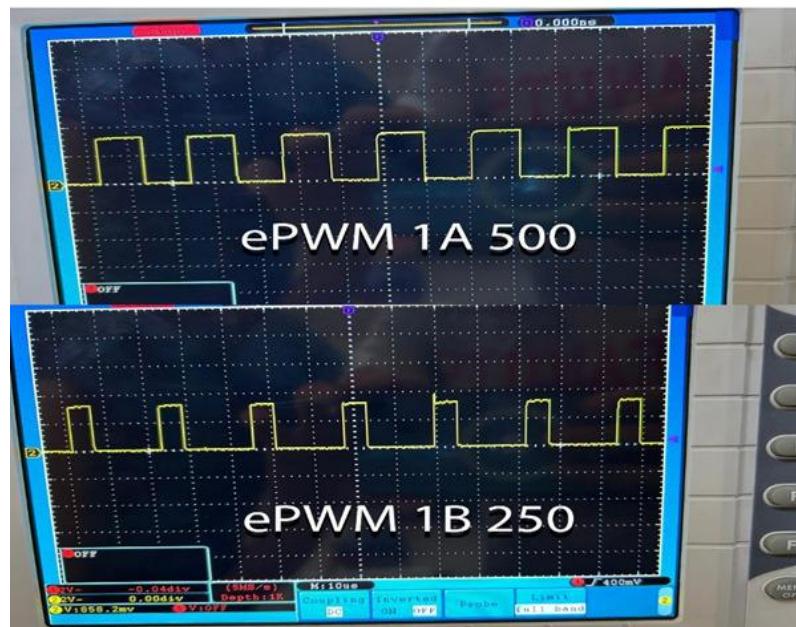


Figure 37: Résultats obtenus des signaux PWM sur oscilloscope

Remarque :

- Un dépassement a eu lieu au niveau de tensions observées
- La fréquence de la PWM pour le mode Up-Down égale à 45KHz pour un Timer_period égal à 1000.
- La valeur de Timer period est limitée entre 0 et 65535.
- Dans ce test on a pris Timer period = 1000

3.1 La fréquence :

La fréquence de la ePWM vaut 45MHz pour un Timer_period égal à 1.

3.2 Les entrées WA et WB :

On voit bien que la valeur du rapport cyclique (α) vaut 50% pour le signal A et vaut 25% pour le signal B, donc la valeur de α est calculée de la façon suivante :

$$\alpha = \frac{w}{\text{timer period}}$$

3.3 Entrainement direct de la valeur du rapport cyclique

Pour entrer directement la valeur du rapport cyclique souhaité on va multiplier note entrée w avec un gain égal à la valeur du Timer periode fixé dans le bloc ePWM (dans notre cas Timer period = 1000) :

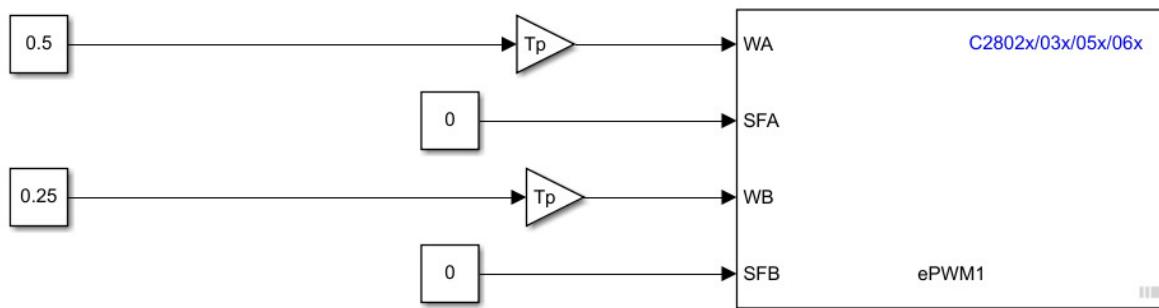


Figure 38: test du bloc ePWM

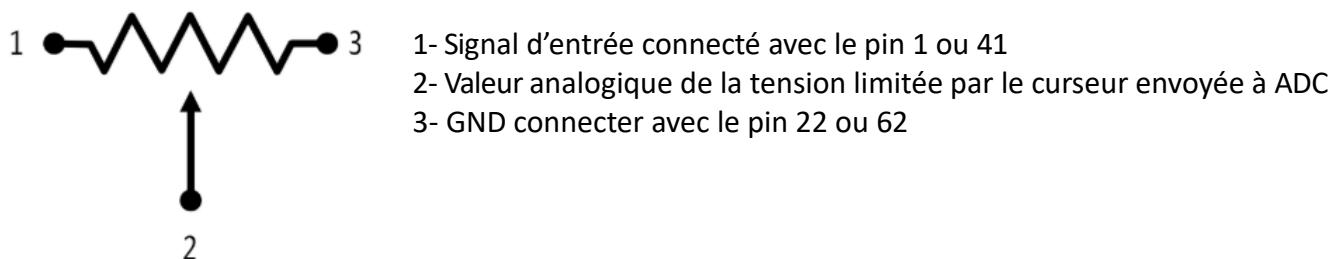
3.4 Les entrées de forçage :

L'entrée de forçage a pour objet de forcer une valeur pour la PWM :

- SFA=0 mode de fonctionnement normale
- SFA=1 la sortie est forcer à prendre la valeur zéro
- SFA=2 la sortie est forcer à prendre la valeur max=3.3v

4. Test des blocs ADC :

Ce test a pour objectif d'apprendre à configurer les blocs ADC.



Ce composant permet d'envoyer une tension analogique vers l'ADC (Convertisseur Analogique-Numérique) de la carte. L'ADC a pour rôle de convertir un signal analogique continu en une valeur numérique exploitable par le microcontrôleur.

4.1 Caractéristiques :

- La conversion numérique est effectuée sur **12 bits**, ce qui donne une valeur maximale de $2^{12} = 4096$.
- La plage de tension d'entrée admissible s'étend de **0 à 3,3 V**.

- La fréquence d'échantillonnage correspond à l'horloge du processeur : **CPUCLK = 90 MHz.**
- La conversion suit la logique suivante :
 - Si la tension d'entrée ≤ 0 V → **valeur numérique = 0**
 - Si $0 < \text{tension d'entrée} < 3,3$ V → **valeur numérique = $(4096 \times \text{Vin}) / 3,3$**
 - Si la tension d'entrée $\geq 3,3$ V → **valeur numérique = 4095**

4.2 Test :

On relie le curseur du potentiomètre avec le pin 27 (ADCINA0), le pôle 1 avec le pin 1 ou 41 (3.3V) et le pôle 3 avec le pin 22 ou 62 (GND) et on configure le bloc ADC comme suivant :



Figure 39: test du bloc ADC

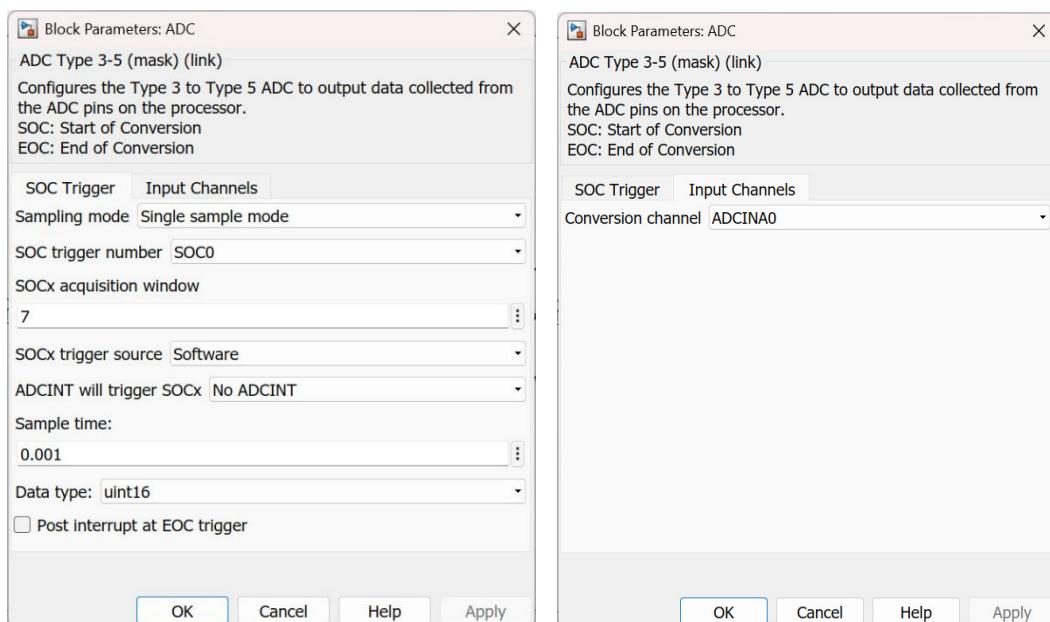


Figure 40: configuration du bloc ADC

4.3 Résultat :

Pour input = 3.3V :

- Pour input = 3.3V :

<code>(*) GPIO_B.adc1</code>	unsigned int	4054	0x0000E01E@
------------------------------	--------------	------	-------------

- Pour input = 1V :

<code>(*) GPIO_B.adc1</code>	unsigned int	1307	0x0000E01E@
------------------------------	--------------	------	-------------

- Pour input = 0V :

<code>(*) GPIO_B.adc1</code>	unsigned int	4	0x0000E01E@
------------------------------	--------------	---	-------------

V. Carte de puissance BOOSTXL-DRV8305EVM :

1. Introduction :

La carte d'extension BOOSTXL-DRV8305EVM est un module développé par Texas Instruments pour la commande de moteurs triphasés, en particulier les moteurs BLDC (Brushless DC) et PMSM (Permanent Magnet Synchronous Motor). Elle est conçue pour s'interfacer facilement avec les cartes de développement LaunchPad™, notamment celles basées sur les microcontrôleurs C2000. Cette carte intègre le pilote de moteur DRV8305, qui assure la commande de six MOSFETs pour former un onduleur triphasé ou un hacheur multi-quadrants, ainsi que la mesure de courant, la régulation de tension et diverses protections matérielles. Elle constitue un outil complet et compact pour le prototypage et le développement de systèmes de commande moteur performants.

L'architecture de la carte est indiquée dans la figure ci-dessous :

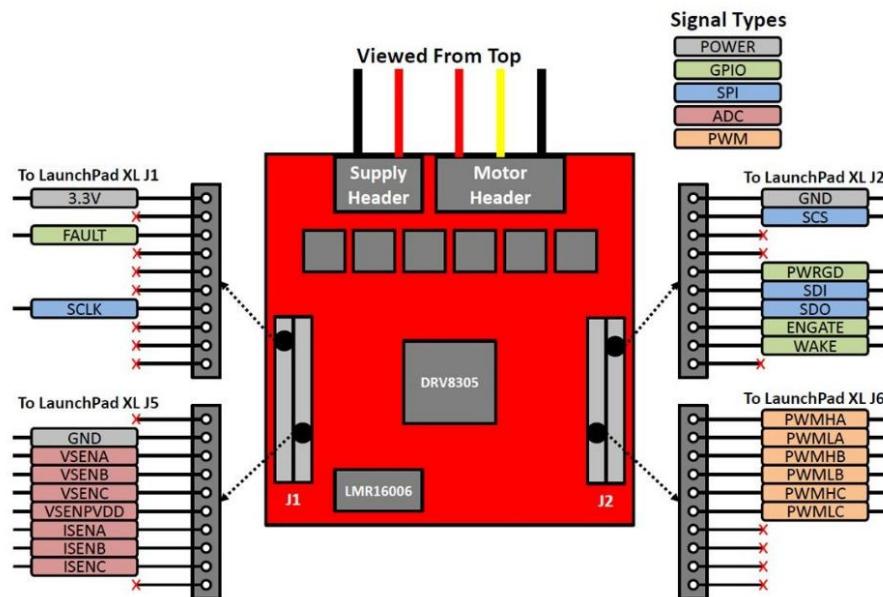


Figure 41: Les pins de sortie de la carte de puissance BOOSTXL-DRV8305EVM

2. Caractéristique

- **Pilote de moteur intégré : DRV8305**
 - Contrôle de 3 demi-ponts pour moteurs triphasés
 - Interface SPI pour la configuration et le diagnostic
 - Intègre des amplificateurs de courant shunt (3 canaux)
 - Régulateur buck intégré 3,3 V / 1,5 A
- **MOSFETs de puissance :**
 - 6 MOSFETs N-channel intégrés
 - Commande d'un moteur jusqu'à 45 V et 10 A (en continu ou en pointe)
- **Tension d'alimentation :** de 6,7 V à 45 V
- **Capteurs de courant :**
 - Mesure sur les 3 phases via résistances shunt
 - Amplification analogique intégrée pour une lecture précise
- **Protections intégrées :**
 - Contre les surintensités (OC), surtensions (OV), sous-tensions (UV)
 - Protection thermique (TSD) et détection de court-circuit
- **Compatibilité :**
 - Interface BoosterPack™ compatible avec les LaunchPads TI
 - Idéal pour une utilisation avec les microcontrôleurs C2000 (ex : F28069M)
- **Utilisation logicielle :**
 - Compatible avec les environnements MotorWare, InstaSPIN-FOC, et Code Composer Studio
 - Permet le développement de stratégies de commande complexes comme les hacheurs quatre quadrants, le PWM symétrique, et le pilotage vectoriel

3. L'objectif

L'objectif de l'utilisation de la carte BOOSTXL-DRV8305EVM dans ce projet est de mettre en œuvre un hacheur quatre quadrants destiné à la commande d'un moteur à courant continu. Cette carte intègre six transistors MOSFET de puissance configurables, ce qui la rend parfaitement adaptée à la réalisation d'un pont complet bidirectionnel, élément central du hacheur quatre quadrants. Grâce à cette architecture, il est possible de contrôler le sens de rotation du moteur ainsi que le flux de puissance dans les deux directions (motoring et régénératif), permettant ainsi un contrôle complet de la vitesse et du freinage. L'intégration de cette carte facilite également la génération des signaux PWM, la mesure des courants et la gestion des protections, contribuant à une commande fiable et performante du moteur.

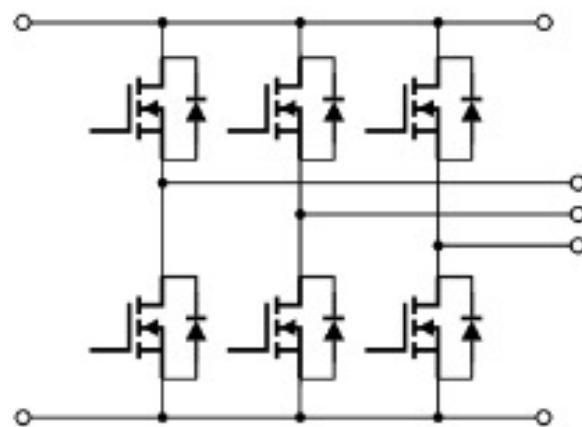


Figure 42: les 6 MOSFET de la carte

Dans notre implémentation du hacheur quatre quadrants, seuls quatre des six MOSFETs présents sur la carte BOOSTXL-DRV8305EVM seront exploités pour former les deux branches du pont en H nécessaire à la commande du moteur à courant continu. Les deux MOSFETs restants seront maintenus désactivés (forcés à l'état bas) afin de ne pas interférer avec le fonctionnement du hacheur. Le pilotage des transistors est assuré à l'aide des blocs ePWM du microcontrôleur C2000, permettant une génération précise et synchronisée des signaux PWM pour contrôler efficacement les quatre quadrants du fonctionnement du moteur.

VI. Commande du moteur à courant continu via DSP

1. Commande en boucle ouverte

1.1 Simulation

Dans cette partie, nous allons assurer la commande du moteur à courant continu (MCC) à l'aide d'un processeur de signal numérique (DSP), tout en bénéficiant de la carte BOOSTXL-DRV8305EVM pour la génération des signaux de commande.

Avant de passer au montage de la simulation on va déclarer toutes les valeurs des constantes qu'on va utiliser à la simulation de la commande du moteur DC, et voilà un fichier .m

```

1      Fmli = 10e3 ; % fréquence MLI en Hertz
2      Tp = 45e6 /Fmli; % TIME PERIOD
3      Vdc = 20; % tension d'alimentation en Volt

```

Figure 43: fichier .m des constantes

Pour faire la commande du moteur, on va utiliser le montage suivant :

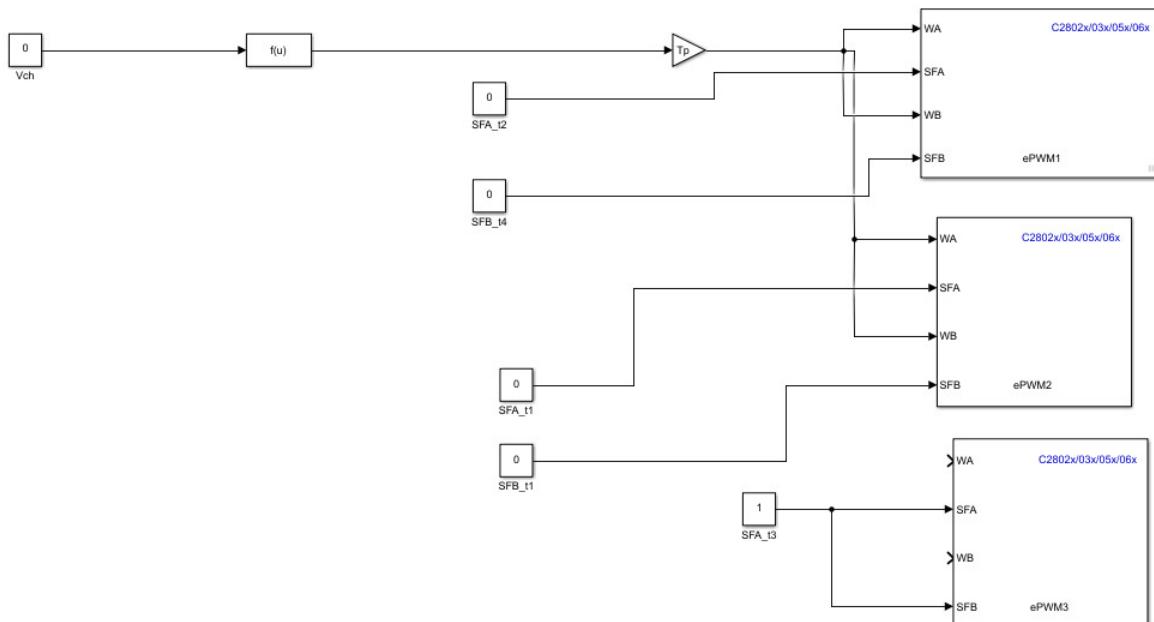


Figure 44: montage de commande du moteur en boucle ouverte

Les blocs ePWM sont configurés comme suite :

- ePWM1 :

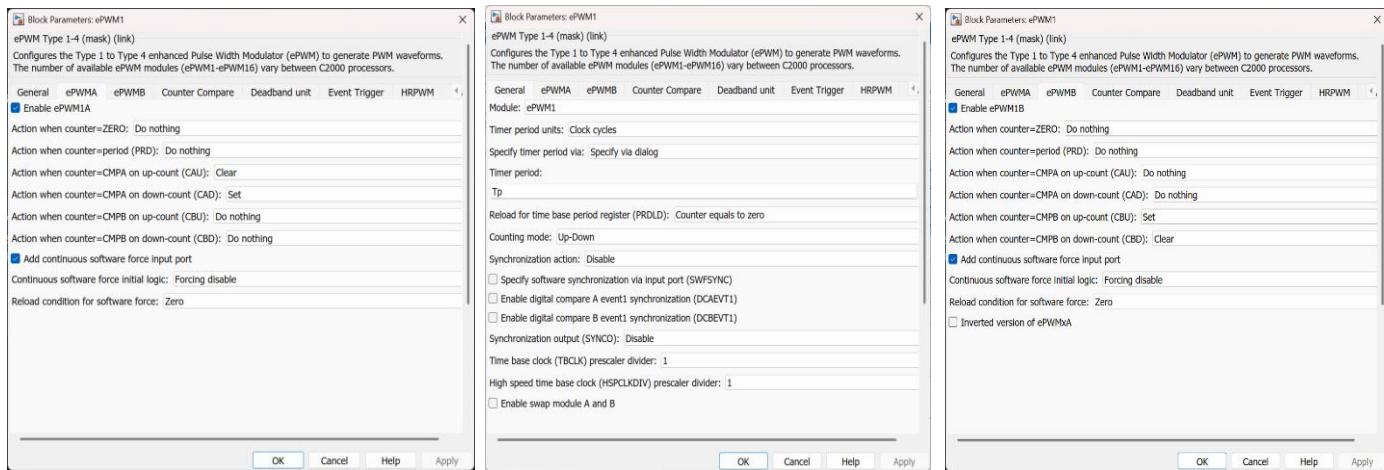


Figure 45: configuration du bloc ePWM1

- ePWM2 :

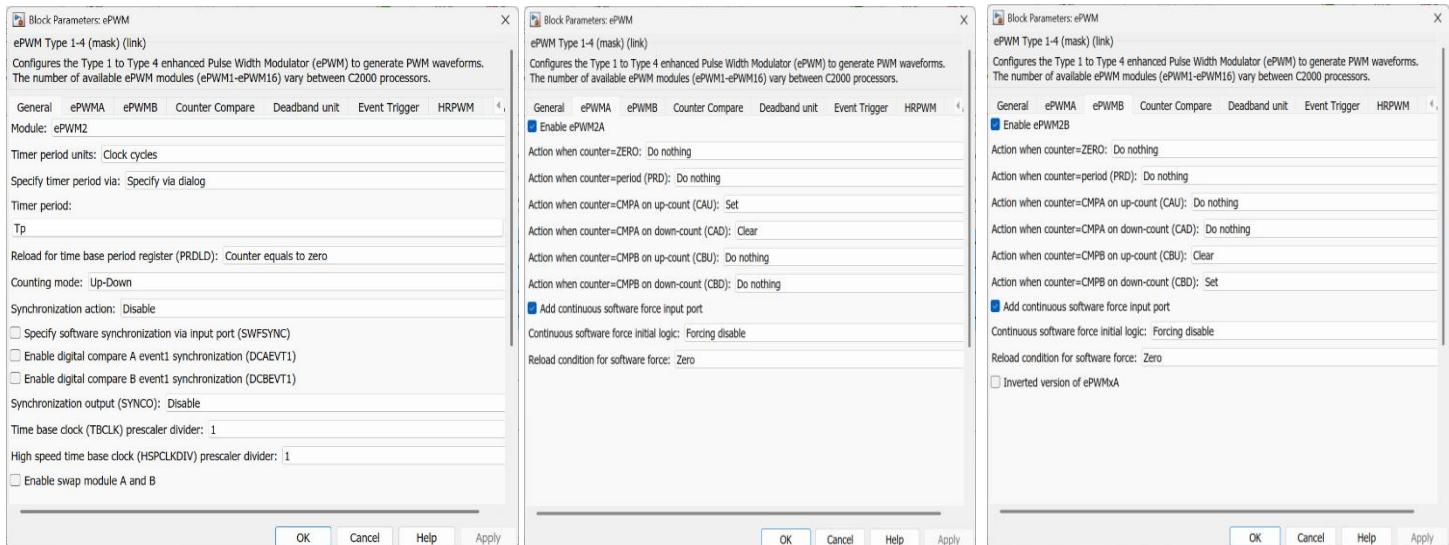


Figure 46: configuration du bloc ePWM2

On va forcer les ePWM1 et ePWM2 à fonctionner en état normal par les constantes **SFA_t2**, **SFB_t4**, **SFA_t1** et **SFB_t1** c'est-à-dire on met **SFA=SFB=0**, alors que le bloc ePWM3 est forcé à l'arrêt par la constante **SFA_t3** (**SFA=SFB=1**) c'est-à-dire on force le troisième bras des transistors à être en état bas pour ne pas perturber le fonctionnement du hacheur.

Pour un hacheur 4 quadrant on a :

$$V_{ch} = (2\alpha - 1) * V_{dc}$$

Avec :

- $V_{ch} = V_{moy}$: Valeur moyenne de la tension désirée aux bornes de la charge ;
- $V_{dc} = 20V$: Tension d'alimentation ;
- α : Rapport cyclique.

$$\text{Alors : } \alpha = \frac{V_{ch}}{2 * V_{dc}} + 0,5$$

Ce qui explique l'ajout du bloc **Fcn** pour le calcul du rapport cyclique α :

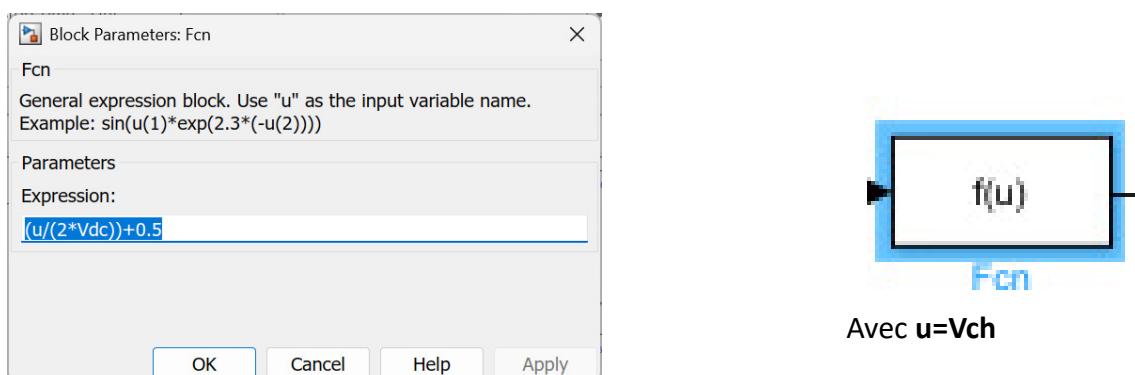


Figure 47: bloc Fcn

Avec $u = V_{ch}$

La valeur entrée dans WA et WB des blocs est la même et elle est calculée par la relation :

$$WA = WB = \alpha * Time\ Period$$

Alors $\alpha = \frac{WA}{TIME\ PERIOD}$ ce qui explique l'ajout du bloc :

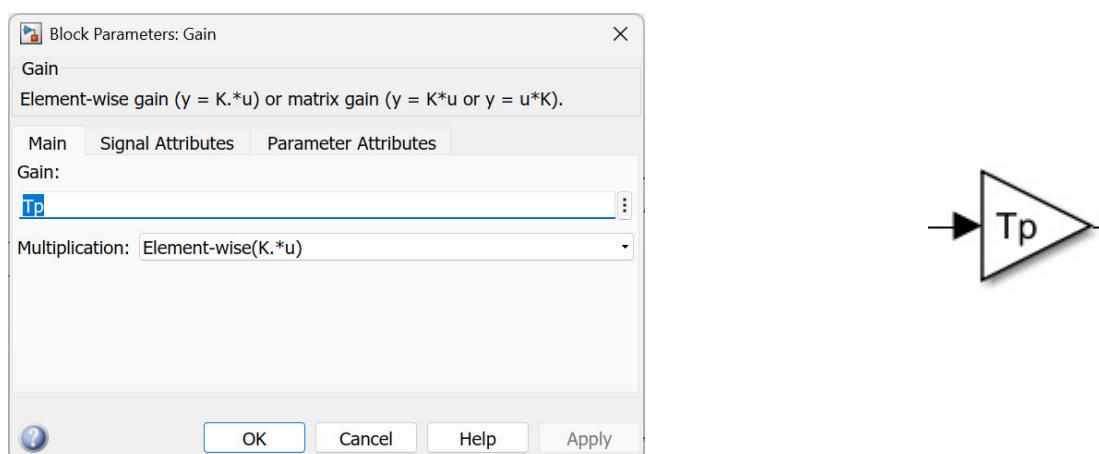


Figure 48: bloc TIME PERIOD

1.2 Application

Afin d'adapter le signal issu du tachymètre du moteur à courant continu (MCC) à la carte de commande numérique (DSP), nous avons réalisé un circuit de conditionnement basé sur un suiveur de tension suivi d'une protection par diode Zener. En effet, la carte DSP ne tolère pas des tensions supérieures à 3,3 V sur ses entrées analogiques. Ce circuit permet donc non seulement de stabiliser le signal du tachymètre sans le charger, grâce à l'amplificateur opérationnel en suiveur, mais aussi de le limiter à un maximum de 3,3 V via une diode Zener, évitant ainsi tout risque de détérioration de la DSP.

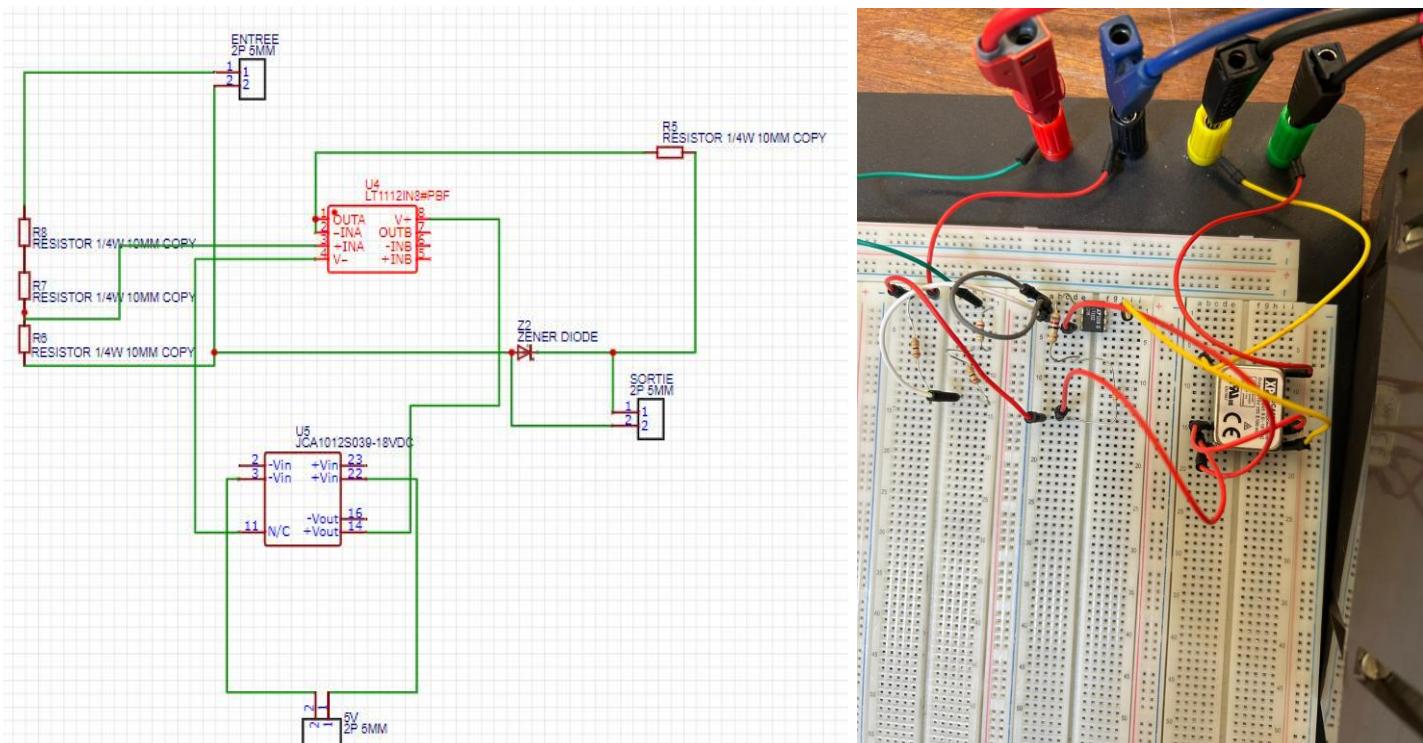


Figure 49:circuit de protection et de régulation de tension

1.2.1 Rôle fonctionnel du circuit (avec tachymètre en entrée)

a. Entrée : Signal tachymétrique

- La borne ENTREE reçoit une tension continue proportionnelle à la **vitesse du moteur**.
- Ce signal est ensuite **abaissé** via un **pont diviseur de tension** composé de R6(20KOhm), R7(20KOhm), R8(20KOhm). Ce traitement permet de :
 - Sert à **adapter** la tension du tachymètre à une plage adéquate pour le traitement.
 - Valeur de diviseur $V_{dV} = \frac{R_6}{R_6 + R_7 + R_8} * V_{in} = \frac{20}{60} * V_{in} = \frac{1}{3} * V_{in}$

b. Amplificateur opérationnel (U4 – LT1112 en suiveur)

- Le montage **suiveur** signifie que :
 - **Vout = Vin**
 - L'entrée positive (+IN) reçoit la tension issue du pont diviseur.
 - La sortie est **reliée directement à l'entrée négative (-IN)**.
- **Pourquoi utiliser un suiveur ?**
 - Il permet de **séparer les étages** : il ne charge pas la source (tachymètre) grâce à sa **très haute impédance d'entrée**.
 - Il fournit une **tension identique à l'entrée, mais avec une capacité de courant plus élevée** pour alimenter l'étage suivant.
 - Cela garantit une **transmission fidèle et stable** du signal de vitesse vers la suite du circuit.

c. Convertisseur DC-DC (JCA0312D03)

- Il fournit l'alimentation symétrique (+V et -V) nécessaire au fonctionnement de l'ampli op.
- Permet un fonctionnement stable même si la source d'alimentation est asymétrique (ex. +5 V seulement à l'entrée).

d. Résistance R5 + Diode Zener Z2

- **R5 limite le courant** vers la Zener.
- La **diode Z2 protège** contre les surtensions en limitant la tension de sortie à une valeur sûre (dans notre cas 3.3V).
- Elle agit comme **clamp** si la tension dépasse sa tension de claquage → cela protège notre carte DSP.

e. Résumer de fonctionnement du circuit :

Ce circuit permet de **conditionner et stabiliser la tension issue d'un tachymètre** connecté à un moteur à courant continu.

La tension tachymétrique est d'abord **abaissée par un pont diviseur**, puis **transmise fidèlement via un suiveur de tension (amplificateur opérationnel LT1112)**. Le suiveur joue un rôle d'**adaptation d'impédance** et de **stabilisation du signal**, assurant que la charge en aval ne perturbe pas la lecture de la vitesse. Enfin, une **diode Zener couplée à une résistance limite** la tension à la sortie, assurant la **protection contre les surtensions** pour les composants sensibles en aval.

La sortie de ce circuit est reliée à l'entrée ADC (Analog to Digital Converter) de la carte DSP afin de permettre la lecture en temps réel de la tension délivrée par le tachymètre.

Et grâce à l'utilisation du bloc ci-dessous on peut lire la valeur du tachymètre en volt :

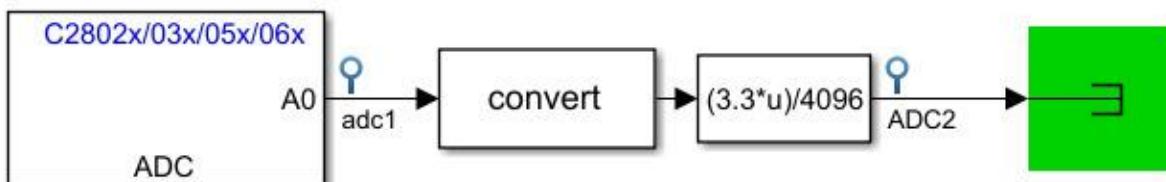


Figure 50: test du bloc ADC (valeur analogique)

1.2.2 Matériels utiliser

- **DC POWER SUPPLY (0-32V ; 0-3A)** : Tension d'alimentation de la carte DSP
- **ALIMENTATION STABILISEE (JEULIN +15 -15)** : alimentation de JCA0312D03
- **OSCILLOSCOPE**
- **MOTEUR A COURANT CONTINUE**

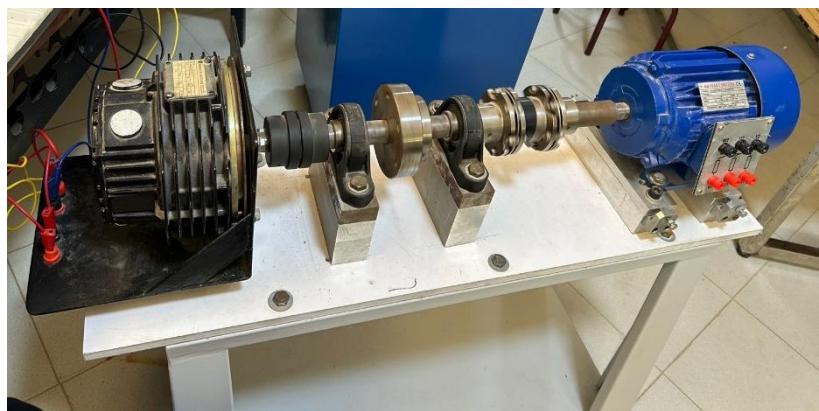


Figure 51: MCC

1.2.3 Test

Dans cette phase, nous allons implémenter le montage présenté dans la figure 44, en l'associant avec celui de la figure 50, et les transférer vers notre carte DSP à l'aide de l'environnement de développement Code Composer Studio (CCS). Cette étape sera réalisée en suivant rigoureusement les instructions détaillées dans les figures 19, 20, 23 à 31 du document de référence. L'objectif est de piloter le hacheur à travers une tension d'entrée définie selon notre choix, permettant ainsi de commander efficacement le moteur à courant continu (MCC). Ce montage nous offrira également la possibilité de lire en temps réel les valeurs du tachymètre, ainsi que la vitesse du moteur.

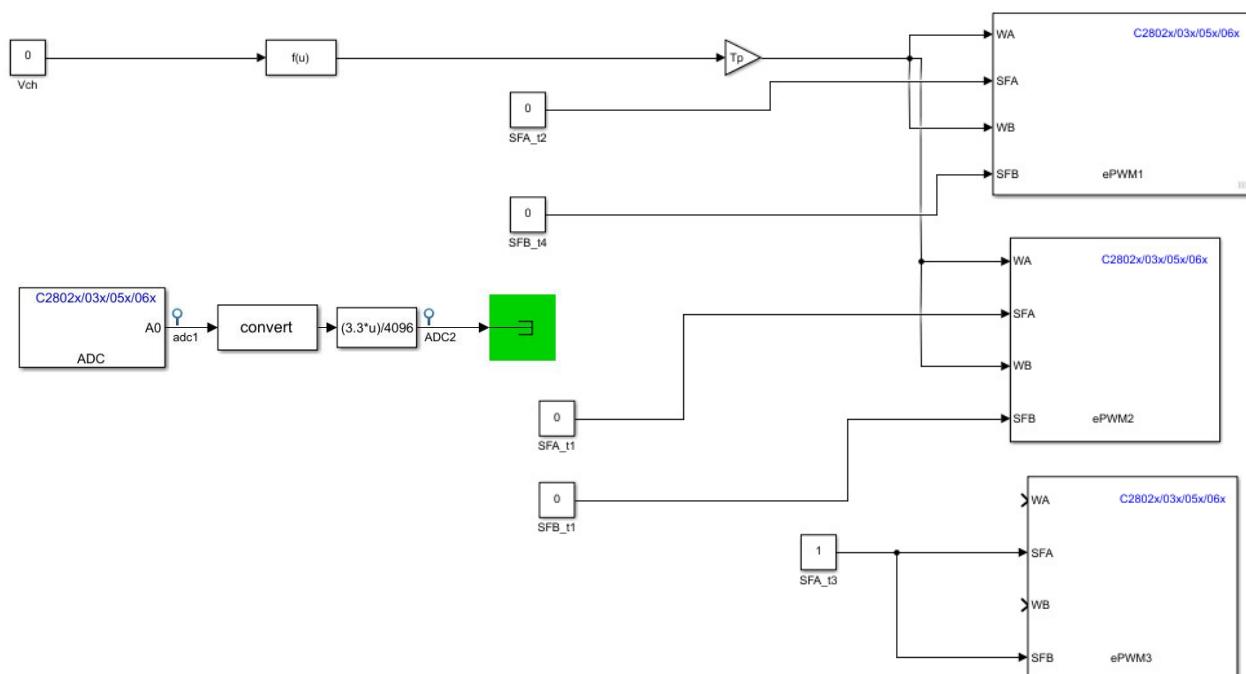


Figure 52:Montage final en boucle ouverte

En reçoit les valeurs suivantes :

Vch	VTachymètre (ADC2)	Vitesse (tr /min)
2	0.07	9
3	0.12	29
4	0.19	45.5
5	0.28	72
6	0.35	96
7	0.42	119.6
8	0.50	144
9	0.57	168
10	0.67	187
11	0.74	216
12	0.82	246
13	0.92	277.5

Tableau 3: les valeurs du test en BO

Dans cette étape, nous allons procéder à la lecture de la vitesse du moteur à l'aide de l'ADC (SPEED), sans recourir à l'utilisation du tachymètre. La valeur mesurée sera affichée dans l'environnement Code Composer Studio (CCS) afin d'assurer un suivi en temps réel. Pour établir une relation fiable entre la tension lue par l'ADC et la vitesse réelle du moteur, nous avons exploité les données du tableau 3 en les représentant sous forme de courbe linéaire à l'aide de l'outil Curve Fitter de MATLAB. Cette approche nous permet de visualiser la tendance des mesures, de modéliser le comportement de la vitesse en fonction de la tension, et surtout de déterminer le gain reliant les deux grandeurs, indispensable pour une estimation précise de la vitesse via l'ADC.

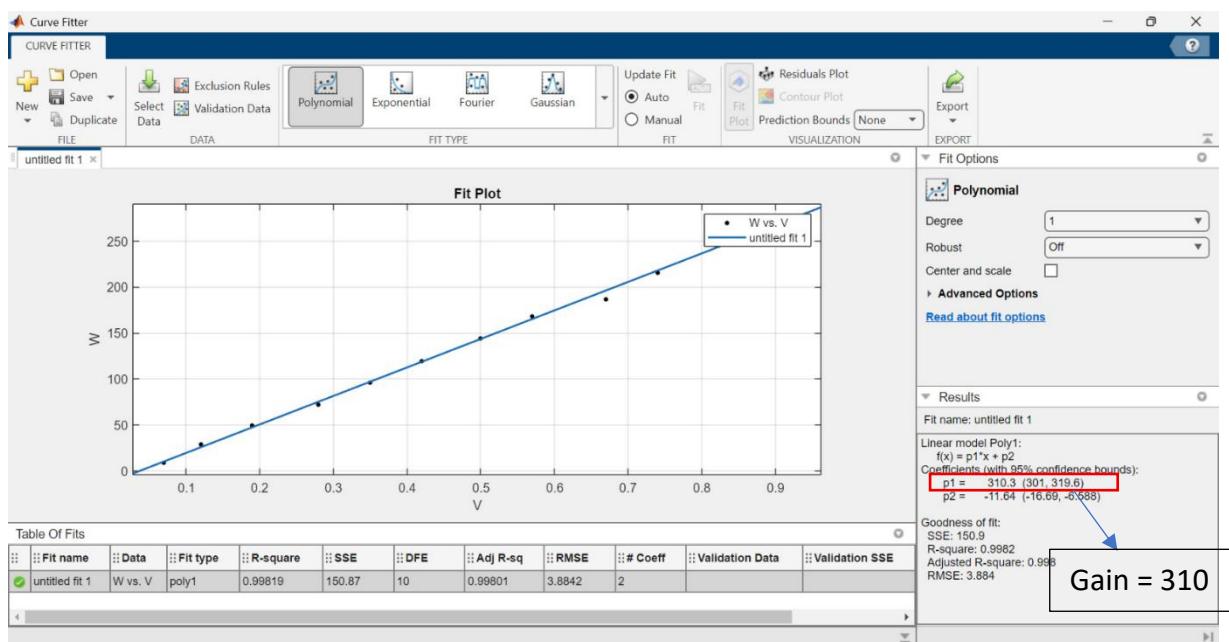


Figure 53: la vitesse en fonction de la tension

Donc le bloc de l'ADC devient :

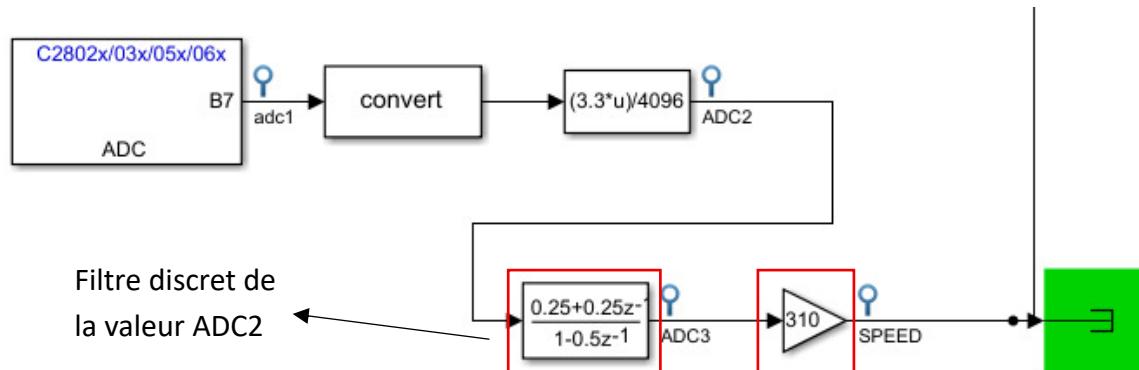


Figure 54: bloc de l'ADC

2. Commande en Boucle Fermer :

2.1 Identification de la fonction de transfert de système :

Pour déterminer la fonction de transfert du système, nous avons appliqué un signal en échelon à l'entrée du montage et observé la réponse temporelle en sortie.



Figure 55: Réponse du système à un échelon

La courbe obtenue (figure 55) présente un comportement caractéristique d'un système du premier ordre, ce qui nous permet de l'approximer comme tel.

- Donc la fonction de transfert de système :

$$H = \frac{G}{1 + \tau m P}$$

Pour déterminer la valeur du gain statique G , nous avons utilisé les données du tableau 3, en l'estimant à l'aide de la relation suivante :

$$G = \frac{\text{Sortie}}{\text{Entrée}} = \frac{\Delta \text{Speed}}{\Delta V_{ref}}$$

$$\text{1er cas : } G = \frac{187 - 168}{10 - 9} = 19 \quad , \quad \text{2ème cas : } G = \frac{144 - 119,5}{8 - 7} = 24,5$$

$$\text{3ème cas : } G = \frac{96 - 72}{6 - 5} = 24 \quad , \quad \text{4ème cas : } G = \frac{49,5 - 29}{4 - 3} = 20,5$$

- Donc :

$$G_{moy} = \frac{19 + 24,5 + 24 + 20,5}{4} = 22$$

Et pour déterminer la constante de temps τm , on applique la méthode des 3τ , qui permet d'identifier visuellement ce paramètre à partir de la courbe de réponse à un échelon.

D'après la figure 55, on a : $3\tau = 0.950 \text{ s} \longrightarrow \tau m = 0.32 \text{ s}$

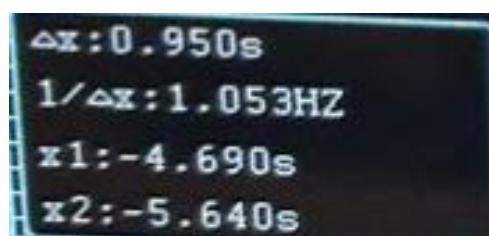


Figure 56:valeur de 3τ visualiser sur l'oscilloscope

2.2 Dimensionnement du correcteur PI parallèle :

Afin de réguler la vitesse du moteur à courant continu (MCC) et de maintenir une valeur constante même en présence de perturbations, telles qu'une variation de charge, nous avons opté pour l'utilisation d'un correcteur PI (Proportionnel-Integral) parallèle. Ce type de correcteur permet non seulement de réduire l'erreur statique, mais également d'assurer une réponse dynamique stable et rapide.

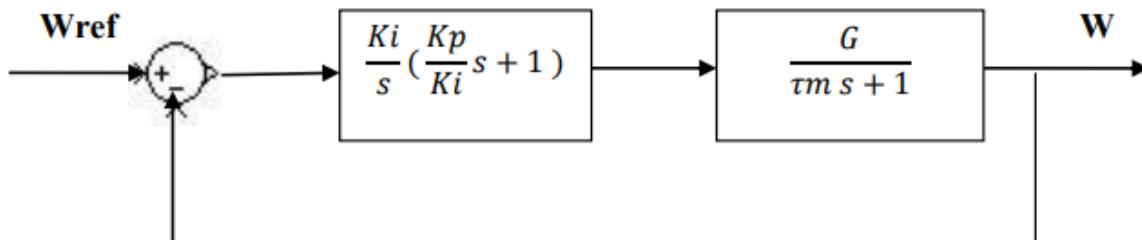


Figure 57: Fonction de Transfert en Boucle Ouverte de notre système

Pour déterminer les gains du correcteur on va appliquer la méthode d'élimination du pole dominant de la fonction de transfert de notre système : On cherche d'avoir une fonction de transfert du système en boucle fermée du 1er ordre avec un gain statique égale à 1 pour que la sortie soit égale à la consigne en régime permanent et avec une constante du temps qui dépend des gain proportionnel et intégral K_p et K_i pour qu'on puisse contrôler la rapidité de notre système :

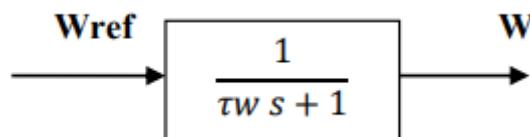


Figure 58: Fonction de Transfert en BF

- La fonction de transfert en boucle ouvert est : $\text{FTBO} = \frac{Ki}{s} \left(\frac{Kp}{Ki} s + 1 \right) * \frac{G}{\tau_m s + 1}$
- Pour avoir $\text{FTBF} = \frac{1}{\tau_w s + 1}$:
 - On va prendre : $\frac{Kp}{Ki} = \tau_m$ et $Ki = \frac{1}{G * \tau_w}$
 - Alors $\text{FTBO} = \frac{1}{G * \tau_w s} (\tau_m s + 1) * \frac{G}{\tau_m s + 1}$
 - $\text{FTBO} = \frac{1}{\tau_w s}$
- D'où la fonction de transfert en boucle fermée est :

$$\text{FTBF} = \frac{1}{\frac{1}{\tau_w s}} \longrightarrow \text{FTBF} = \frac{1}{\tau_w s + 1}$$

Donc les gains de notre correcteur PI sont :

$$K_p = \frac{\tau_m}{G * \tau_w} \quad \text{et} \quad K_i = \frac{1}{G * \tau_w}$$

On a choisi de prendre le temps de réponse du système avec correcteur égal au temps de réponse du système sans correcteur, c'est-à-dire $\tau_w = \tau_m$ alors dans notre cas, les gains proportionnels et intégrale vont être égale à :

$$K_p = \frac{\tau_m}{G * \tau_w} = 0.04545$$

$$K_i = \frac{1}{G * \tau_w} = 0.142045$$

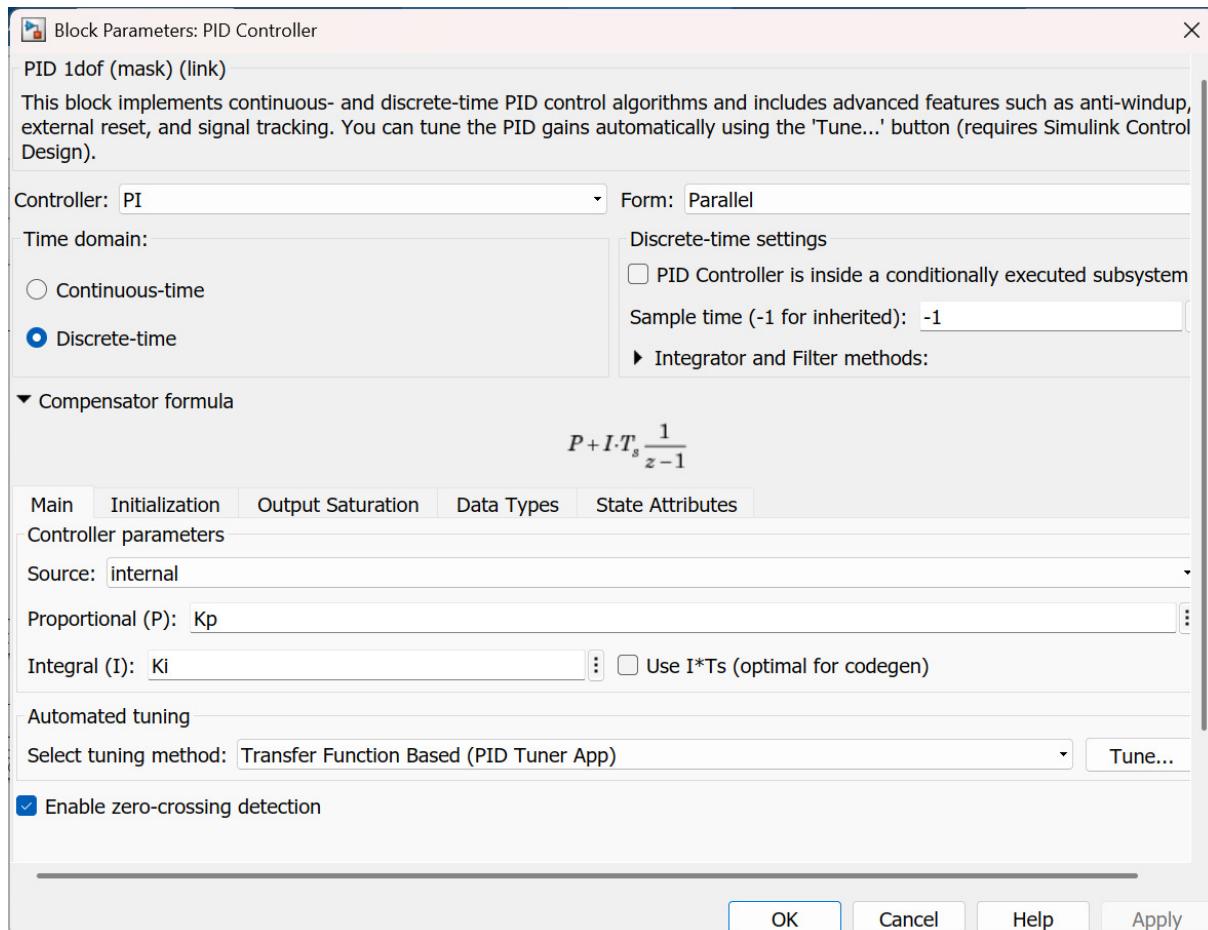


Figure 59: Configuration du bloc pi

Chapitre 3 : Résultats final (H4Q + MCC + CORRECTEUR PI)

I. Application

Pour récapituler l'ensemble des étapes réalisées dans notre projet, nous avons effectué un test final en téléchargeant le programme de la boucle fermée vers la DSP. À ce stade, le système est entièrement opérationnel, et il est désormais possible de saisir directement la valeur de vitesse souhaitée dans l'interface CCS. Le correcteur PI se charge ensuite d'ajuster automatiquement le signal de commande afin de maintenir cette vitesse, quelles que soient les conditions de charge appliquées au moteur. Ce test final permet de valider l'efficacité de notre système de régulation basé sur une commande en boucle fermée.

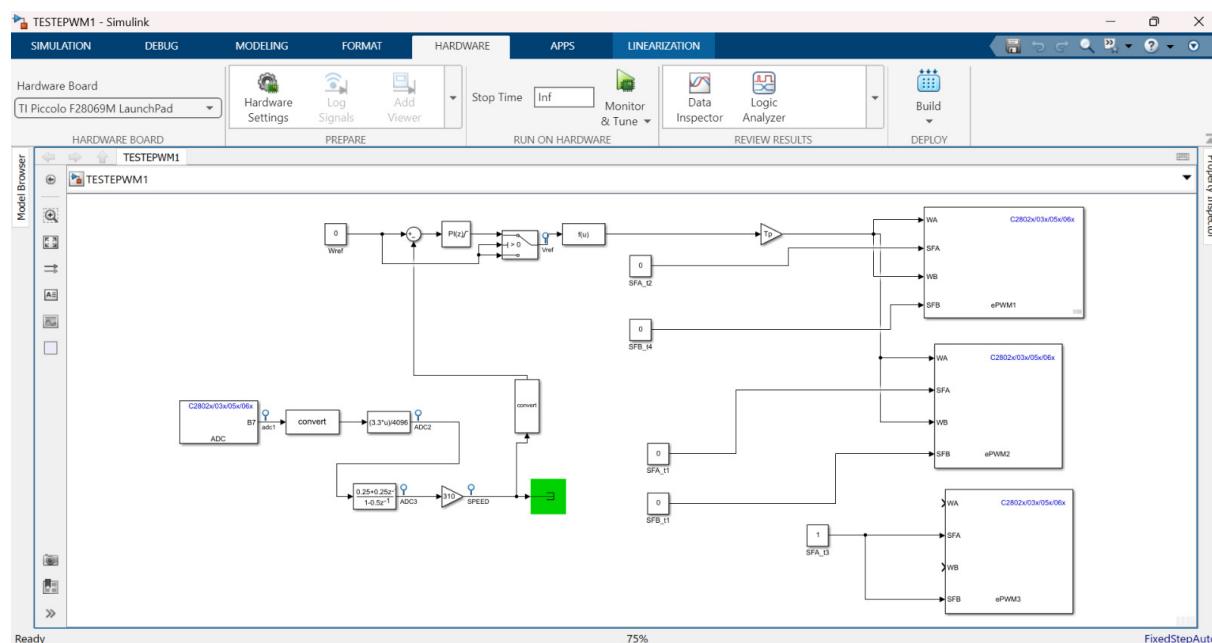


Figure 60: programme final

1. Test sans charge

(*)= TESTEPWM1_B.ADC3	float	0.318044305	0x0000E064@Data
(*)= TESTEPWM1_B.ADC2	float	0.335156232	0x0000E062@Data
(*)= TESTEPWM1_B.adc1	unsigned int	416	0x0000E068@Data
(*)= TESTEPWM1_B.SPEED	float	102.665924	0x0000E066@Data
(*)= TESTEPWM1_B.Vref	double	5.47996378	0x0000E060@Data
(*)= TESTEPWM1_P.Wref_Value	double	100.0	0x0000E00E@Data

Figure 61: valeurs des constantes (vitesse = 100tr/m)

- **Wref** : vitesse du MCC souhaitée (dans ce cas Wref = 100 tr/min)
- **Vref** : tension moyenne du H4Q régler par le PI afin d'être compatible avec Wref
- **adc1** : valeur numérique du tachymètre
- **ADC2** : valeur analogique du tachymètre (en Volt)
- **ADC3** : ADC2 filtrer
- **SPEED** : valeur de la vitesse à travers l'ADC (figure 54)

Deuxième test

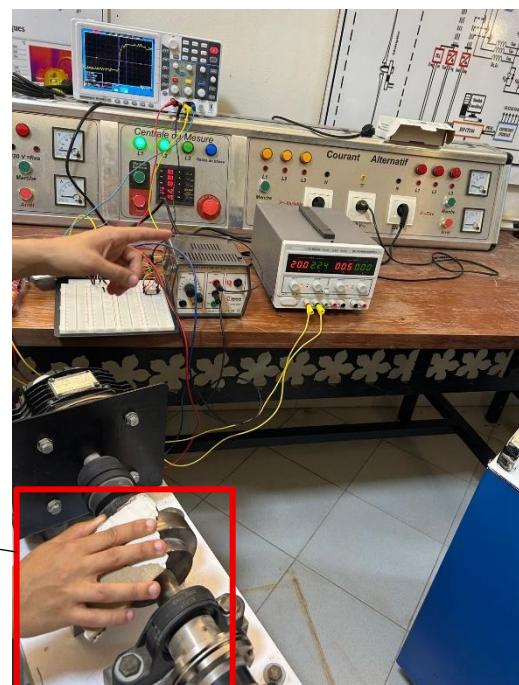
TESTEPWM1_B.ADC3	float	0.677818179	0x0000E064@Data
TESTEPWM1_B.ADC2	float	0.655810535	0x0000E062@Data
TESTEPWM1_B.adc1	unsigned int	806	0x0000E068@Data
TESTEPWM1_B.SPEED	float	204.195145	0x0000E066@Data
TESTEPWM1_B.Vref	double	9.64780903	0x0000E060@Data
TESTEPWM1_P.Wref_Value	double	200.0	0x0000E00E@Data

Figure 62: valeurs des constantes (vitesse = 200tr/m)

2. Test avec charge

Lorsque la charge est appliquée au moteur, nous observons que le système de commande réagit automatiquement en ajustant la grandeur de sortie. Plus précisément, le courant augmente afin de compenser l'effort supplémentaire demandé par la charge, ce qui permet de maintenir la vitesse de rotation constante.

Dans notre projet, le freinage appliqué au moteur en tant que charge est réalisé de manière manuelle. Ce freinage mécanique est exercé directement sur l'arbre du moteur afin de simuler une variation de charge réelle.



Charge



Figure 63:courant sans charge



Figure 64:courant avec charge

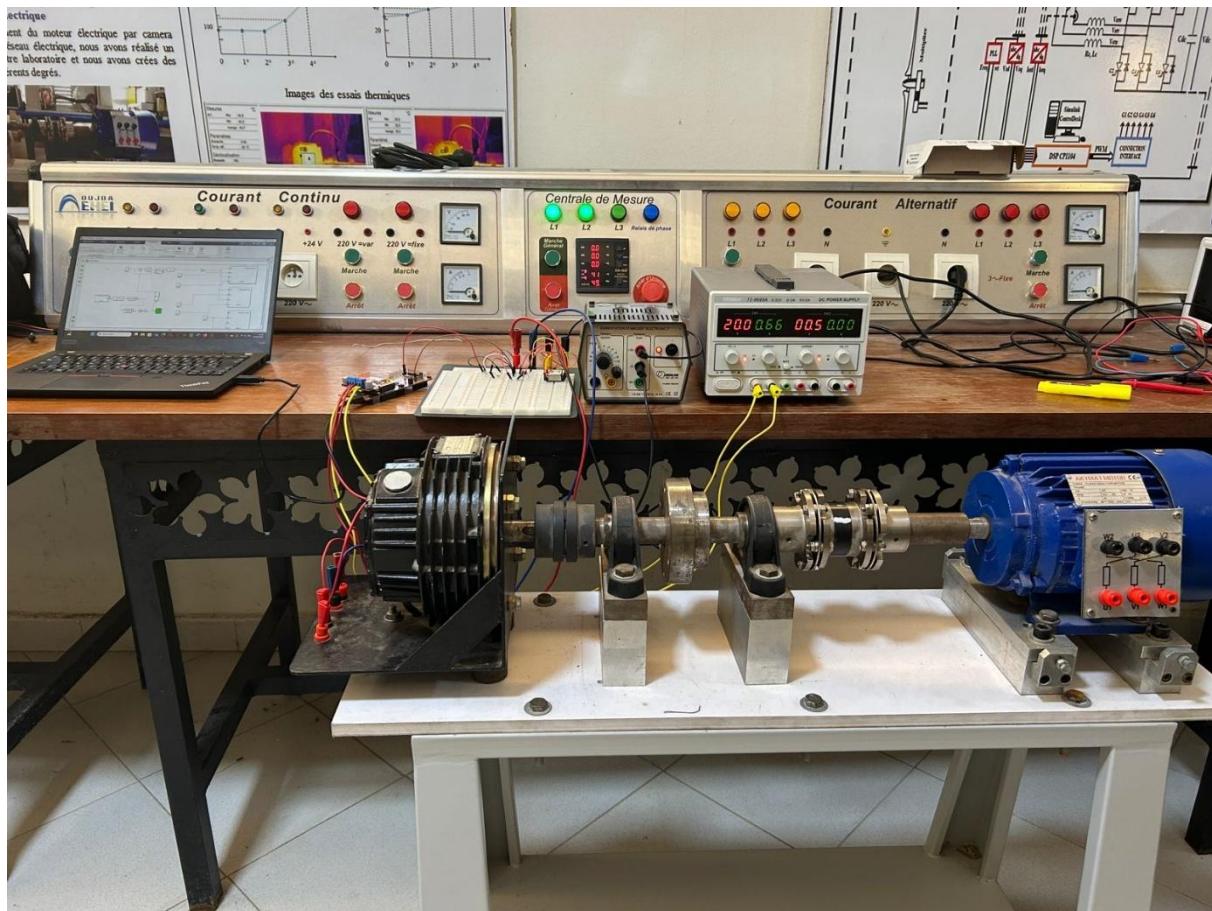


Figure 65: Montage expérimental du projet réalisé

Conclusion

Ce projet nous a permis de concevoir et de réaliser un système complet de régulation de la vitesse pour un moteur à courant continu, en s'appuyant sur une approche mêlant modélisation, simulation et implémentation pratique. Grâce à l'utilisation du hacheur quatre quadrants et d'une carte DSP performante, nous avons pu mettre en place une commande robuste, capable d'adapter la vitesse du moteur aux consignes fixées, même en présence de perturbations telles qu'une variation de la charge.

Les résultats obtenus en boucle ouverte et en boucle fermée ont confirmé la validité des modèles théoriques et la justesse du correcteur PI dimensionné. Ce travail a également illustré l'importance de l'interface entre les environnements de simulation (Simulink/MATLAB) et les plateformes matérielles (Code Composer Studio), facilitant la transition vers des applications industrielles concrètes.

En somme, ce projet a renforcé nos compétences techniques et méthodologiques, tout en soulignant le rôle essentiel des systèmes embarqués dans le pilotage de dispositifs électromécaniques modernes. Il constitue ainsi une base solide pour de futurs développements dans le domaine du contrôle-commande.