

RAPPORT BE AUTOMOBILE

BOURLOT Xavier, DEVAUD Emma, 5 ESPE, Binôme numéro uno, for the win

18 janvier 2021



Table des matières

1	Introduction	1
2	Partie 1 : Analyse de la commande U/f	1
2.1	Questions d'introduction	1
2.2	Commande U/F	4
2.2.1	Simulation Matlab	4
2.2.2	Simulation sur le moteur virtuel	6
2.2.3	Mise en place de la commande sur le prototype de démonstration	6
3	Mise au point de la commande FOC	7
3.1	Simulation de la commande FOC	8
3.1.1	Schéma bloc sous Simulink	8
3.1.2	Détermination des paramètres des correcteurs PI	8
3.1.3	Résultats de la simulation sous Simulink	10
3.2	Test sur carte micro contrôleur	11
3.2.1	Implémentation du code et simulations sur le moteur virtuel	11
3.2.2	Tests sur le moteur réel	12
4	Analyse Safety	13
4.1	Description de la défaillance, conséquences et safe states	13
4.2	Mécanismes de détection des défaillances	14
4.3	Algorithme de détection des défaillances	15
5	Conclusion	17

1 Introduction

Dans la première partie de ce BE automobile, après avoir répondu à des questions sur le fonctionnement général du module "Main Inverter", nous avons simulé et implémenté une commande U/F. Ensuite, pour améliorer les performances du système nous avons mis en place une commande plus complexe : Field Oriented Control (FOC). Enfin, nous avons réalisé une analyse safety se rapportant à un cas de défaillance particulier.

Vous pouvez retrouver toutes la documentation ainsi que notre code via le lien suivant :
https://github.com/reivax-boucoi/BE_Auto

2 Partie 1 : Analyse de la commande U/f

2.1 Questions d'introduction

(1) Quelles sont les rôles des boucles dites fast-loop et slow-loop du programme ? Quelles sont leurs périodes d'exécution ? Comment sont-elles déclenchées ?

- ☞ Fast loop : acquisition des courants de phase et calcul de la commande pwm, à 120us, triggered par l'ISR de l'adc
- ☞ Slow loop : mise à jour de la consigne de vitesse à 1.20ms. Appelée une fois sur dix dans l'interruption fast loop grâce à une variable cpt_rSlow_loop interne.

(2) La mesure des courants de phase, de l'angle et de la vitesse rotor sont-elles indispensables au bon fonctionnement de la commande U/f ? Pourquoi ?

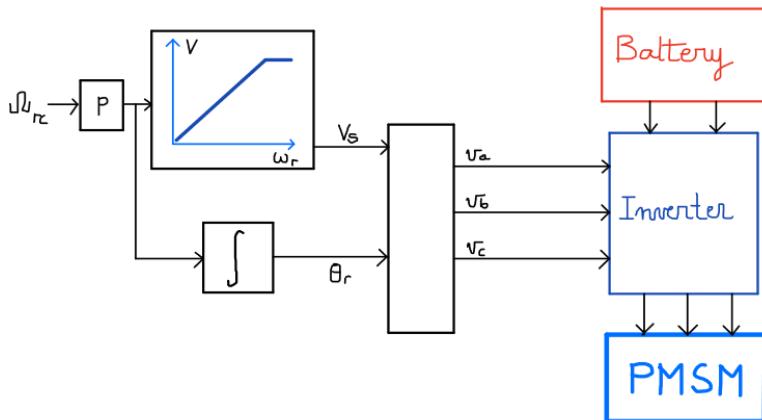


FIGURE 1 – Principe de la commande U/F (schéma G.Garcia)

- ☞ La commande U/F est une commande en boucle ouverte, elle exploite une constante du moteur $K=U/F$ pour imposer une tension de phase. Donc ces mesures ne sont pas indispensables.
- ☞ Cependant, la mesure de l'angle moteur aide pour la transformation inverse de Park, au lieu d'estimer l'angle de manière logicielle. Elle peut également permettre de mieux gérer la phase d'alignement du rotor avec le champ statorique (différence d'angle au démarrage).

(3) Que représentent les commandes U_d et U_q ? Pourquoi cherche-t-on à forcer la commande U_d à 0 ? Peut-on réellement le garantir ? De quel(s) paramètre(s) dépend U_q ?

- ☞ Ce sont les amplitudes des tensions dans le plan de Park.
- ☞ On cherche à maximiser la composante tangentielle (U_q), et à minimiser la composante radiale (U_d) pour obtenir un couple maximal sur le rotor en alignant le vecteur courant du stator (qui produit le champ magnétique) sur le vecteur couple du rotor.

☞ Étant en boucle ouverte aucune surveillance n'est effectuée de ce fait on ne peut rien garantir.

☞ Les paramètres dont dépend Uq sont les suivants :

- résistance interne du bobinage
- l'inductance cyclique
- vitesse de rotation du rotor
-

(4) Les rapports cycliques des commandes PWM appliquées sur les trois phases sont-ils mis à jour simultanément ? Si oui, pourquoi et comment ?

☞ Ils sont mis à jour simultanément, pour éviter des mauvaises configurations entre les bras de l'onduleur, et donc des transitoires importants. Les valeurs de PWM sont préchargées dans des registres (Config_PWM0_SignalParams) avant d'être chargés simultanément sur les 3 sous-modules (LoadPWM0_Params).

(5) Comment les courants de phase sont-ils mesurés ? Quels composants sont employés ? Quelle tension est lue par le microcontrôleur lorsqu'aucun courant de phase ne circule ?

☞ Avec des résistances de shunt placées sur les bras de l'onduleur, on génère une tension proportionnelle au courant de phase, amplifiée par un ampli d'instrumentation et numérisée par les ADCs du microcontrôleur.

☞ Quand $I=0$ la valeur de tension lue est VREF/2 qui correspond à la tension du MCU (MCU_VCC) divisée par deux.

(6) Quel est le rôle de la phase de calibration ? Quelles valeurs de rapports cycliques sont imposés aux commandes PWM des trois phases ? Est-ce une valeur adéquate ?

☞ Le rôle de la phase de calibration est de supprimer les offsets des mesures de courant (critique pour éviter tout courant DC dans la machine, i.e. pertes).

☞ Pour éviter qu'il y ait un courant constant qui circule dans la machine, on impose 50% de rapport cyclique aux commandes des PWM, soit une tension de phase nulle.

☞ Oui car avec cette commande aucun courant ne circule dans les résistances de shunt.

(7) Est-ce que les 3 courants de phase sont lus ? Pourquoi ? Quel événement déclenche la lecture des courants ? Comment la synchronisation entre la lecture des courants et cet événement est-elle assurée ?

☞ Non

☞ Seuls 2 courants de phase sont mesurés, car on peut déterminer mathématiquement l'un des courants à partir des deux autres, car $I_a + I_b + I_c = 0$.

☞ La commutation de la sortie PWM déclenche le trigger CTU qui permet de fixer un délai avant d'amorcer l'ADC. Ainsi, la mesure en courant est toujours synchronisée avec la commutation, ce qui permet d'éviter de mesurer des phénomènes transitoires, et d'obtenir une valeur de courant moyen réel (pas d'aliasing).

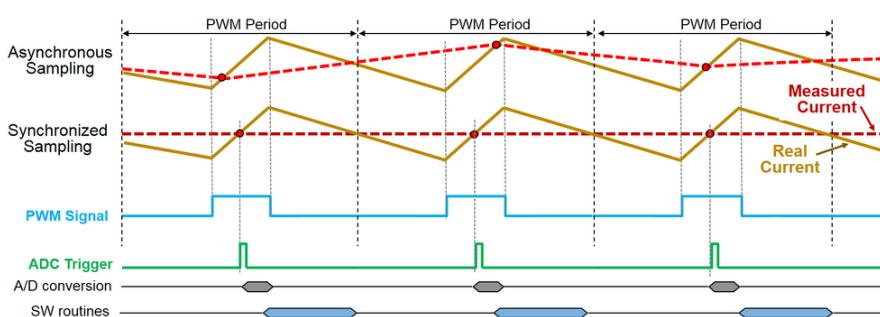


FIGURE 2 – Mesure de courant synchronisée (image NXP)

(8) Pourquoi deux ADC indépendants sont-ils utilisés pour assurer la lecture des courants de phase ?

Pour avoir une lecture simultanée des courants : cohérence de phase entre les mesures, et précision temporelle de l'acquisition par rapport au cycle PWM.

(9) Comment les courants de phase sont-ils mesurés sur la carte gate driver MOTOR GD ? Quelle contrainte le système de mesure des courants de phase impose-t-il sur l'alignement de la PWM ?

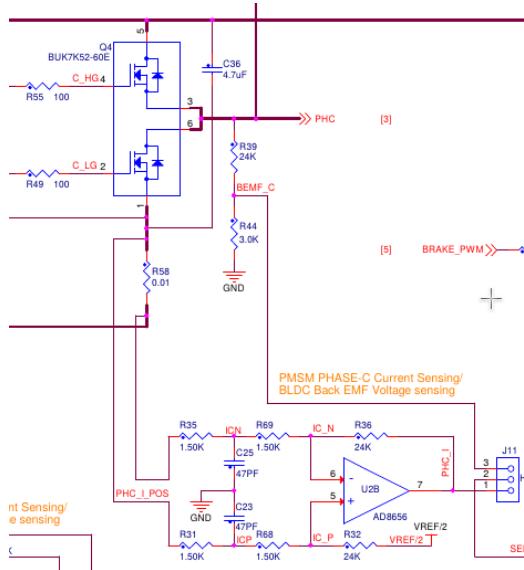


FIGURE 3 – Schéma de mesure d'un courant de phase

Les courants de phase sont mesurés sur la partie inférieure de chaque bras de l'onduleur, grâce à une résistance de shunt et un amplificateur différentiel (cf Fig.3).

Cette configuration implique que la mesure de courant doit être effectuée lorsque la partie inférieure du bras est ON, donc de manière synchrone à la PWM.

(10) Expliquez brièvement le principe et l'intérêt de la Space Vector Modulation (SVM).

Le rôle de la SVM est de produire les PWM qui commandent les trois phases de l'onduleur à partir des commandes en tension U_α et U_β issues de la transformée de Park inverse. Cette modulation particulière permet d'obtenir des orientations de champ plus fines qu'en commande bipolaire classique, et de maximiser le couple pour une tension d'alimentation donnée (3rd harmonic injection).

L'onduleur est considéré comme une machine à états pouvant atteindre 8 états différents. De ce fait le principal intérêt réside dans le fait que cette méthode permet de calculer directement la commande pour les trois phases. Au contraire des techniques de modulation bipolaire et unipolaire qui considèrent chacune des phases séparément.

(11) Quelle est la différence entre l'angle rotor mécanique et électrique ? Lequel intervient dans la transformation de Park ?

Le $\Omega = \frac{\omega}{p}$ correspond à l'angle mécanique du rotor. Ici dans notre cas le moteur possède 3 paires de pôles.

L'angle qui intervient dans la transformation de Park est mécanique l'angle entre le rotor et le champ statorique.

(12) Quel type de capteur acquiert la position et la vitesse du rotor ? Quel(s) sont les rôles du circuit PGA411 ?

Le capteur utilisé est de type résolveur (cf Fig.4).

Le circuit est capable de générer un sinus pour exciter le résolveur, et de mesurer les retours sin et cos afin de déterminer mathématiquement l'angle et la vitesse du rotor.

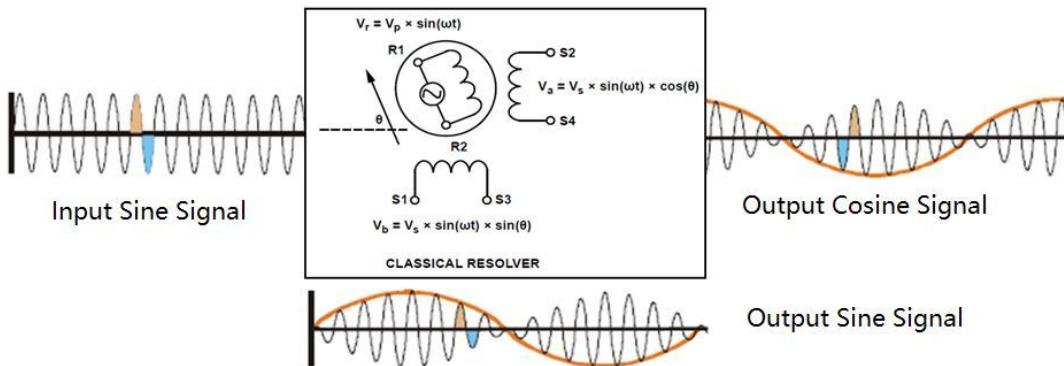


FIGURE 4 – Schéma de principe d'un résolveur

(13) Pourquoi n'acquierte-on pas directement la vitesse rotor à partir de celle que pourrait renvoyer directement le PGA411 ?

☞ Cette information n'est pas fiable pour de faibles vitesses, à cause de la grande résolution nécessaire dans ce cas. Il est alors préférable d'intégrer manuellement l'angle sur une période plus longue pour obtenir une bonne résolution sur la vitesse.

(14) Entrez les paramètres de la LUT reliant la vitesse rotor à atteindre et la tension de commande. Détaillez brièvement votre démarche pour les déterminer ?

☞ cf 2.2.1

2.2 Commande U/F

2.2.1 Simulation Matlab

Le modèle fourni sous Matlab permet simuler le comportement du banc d'essai en charge, d'observer les courants et tensions de phases, ainsi que les paramètres mécaniques (couple et vitesse). En particulier, on peut définir graphiquement la courbe de correspondance tension/vitesse de rotation (cf graphe Fig.1). Pour définir la relation, on procède de la manière suivante :

1. Choix d'une valeur de $K_v > 2.2V/kRPM$ valeur min selon la documentation du moteur (tenir compte des frottements). On a donc la pente de la partie linéaire de la courbe.
2. Convertir les volts en rapport cyclique (100% \Leftrightarrow Valim)
3. Calculer la vitesse pour laquelle la tension d'alimentation est saturée : $Vitesse_{max} = \frac{Valim * 1000RPM}{K_v}$, et saturer la sortie à 100% de rapport cyclique au delà.
4. Déterminer la tension min à appliquer au moteur pour vaincre les frottements. Calculer la vitesse de rotation à l'intersection du seuil bas et de la droite de K_v , et saturer à la tension min la commande pour les vitesses inférieures ou égales.

Graphiquement, le résultat est observable sur la figure suivante :

Nous avons réalisé des simulations à partir du modèle simulink les résultats sont disponibles ci-dessous :

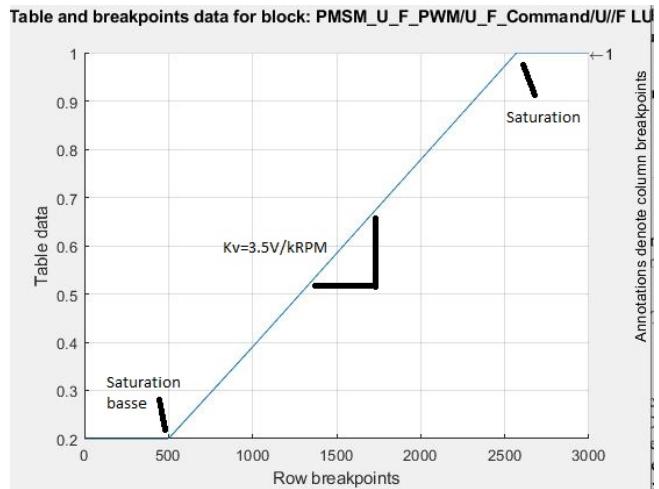


FIGURE 5 – Visualisation de la LUT sous matlab

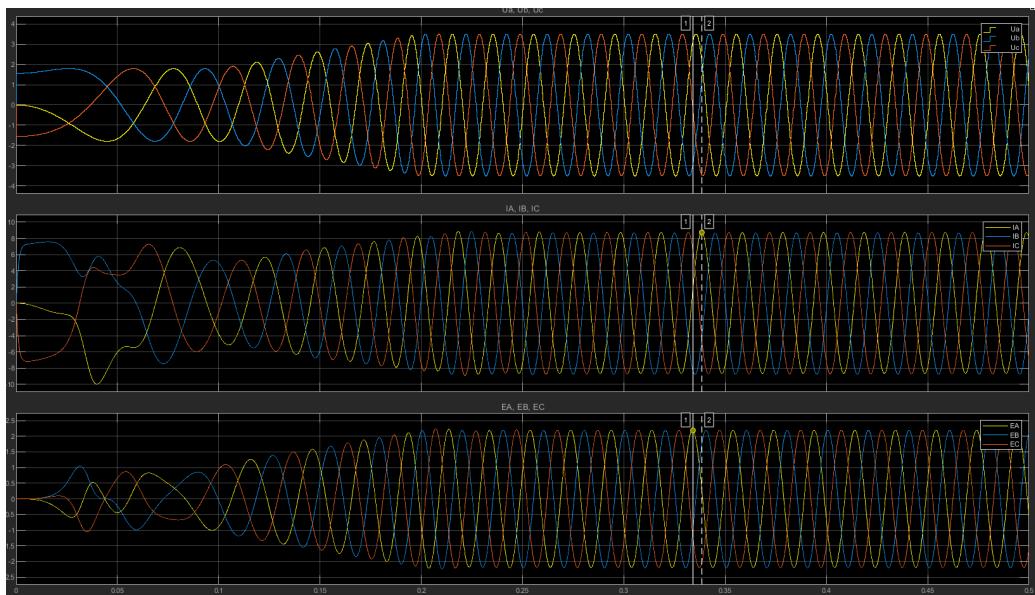


FIGURE 6 – Visualisation des courants, des tensions et de la fem pour une consigne à 1000 RPM

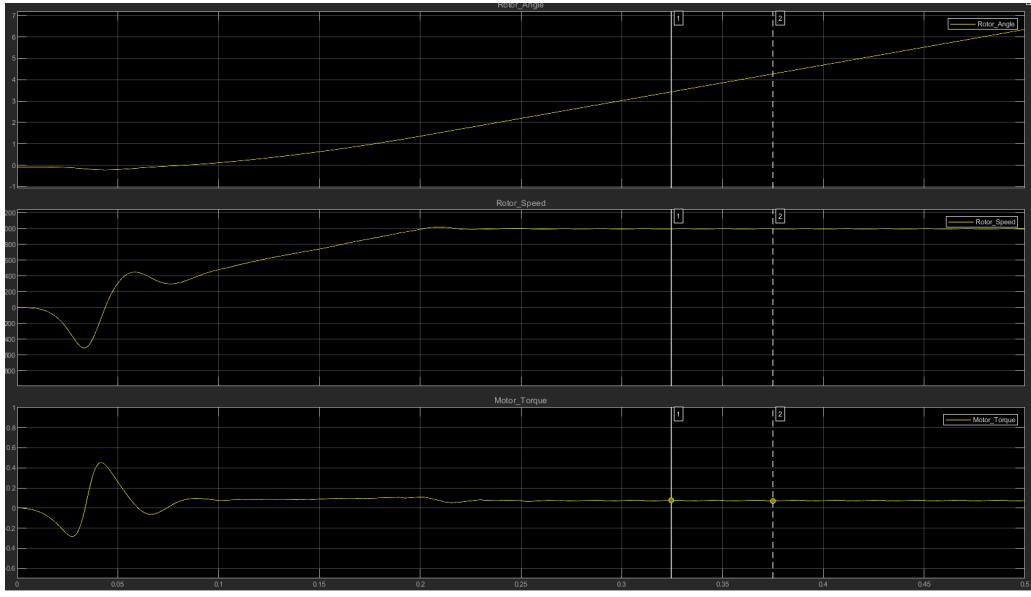


FIGURE 7 – Visualisation des grandeurs mécaniques : l'angle, vitesse et couple du rotor

2.2.2 Simulation sur le moteur virtuel

Une fois nos simulations validées sous Simulink, et avant de tester notre commande sur le moteur réel, nous avons implantée notre commande sur un "moteur virtuel". Nous avons utilisé S32 Design Studio ainsi que kit de développement DEVKIT-MPC5744P pour simuler le comportement du moteur. Pour observer les résultats, l'application FREEMASTER nous a permis de récupérer les différents signaux importants et de les afficher comme sur un oscilloscope.

Nous avons effectué plusieurs tests avec des valeurs de consigne différentes :

+/- 30 RPM

+/- 1000 RPM

+/- 3000 RPM

Cela nous a permis de vérifier que les phases et la sortie évoluent correctement avant de tester sur le matériel. La simulation fonctionnant correctement pour les vitesses plus élevées ainsi que pour les vitesses négatives, nous avons mis en place la commande sur le banc moteur réel.

Ci dessous les résultats sous FREEMASTER pour une commande de 30 RPM.

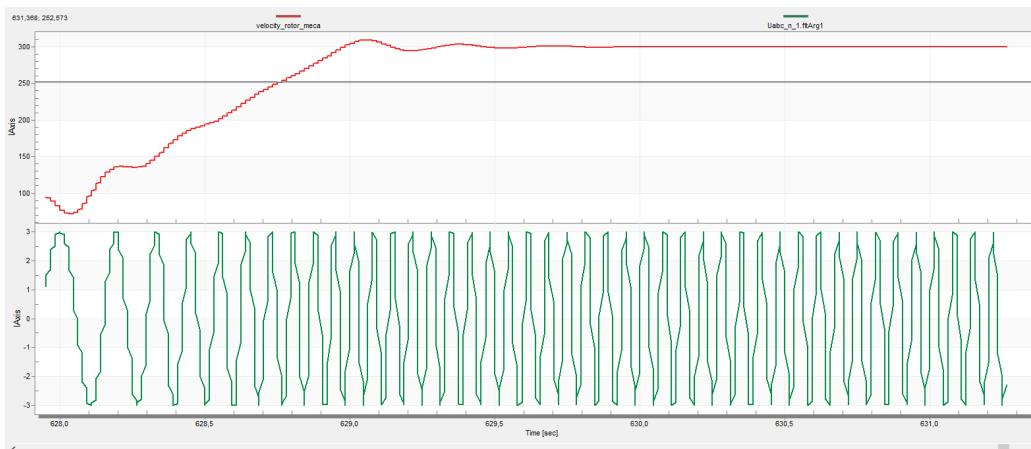


FIGURE 8 – Vitesse du rotor et phase UA du moteur virtuel.

2.2.3 Mise en place de la commande sur le prototype de démonstration

Le comportement du moteur en simulation étant vérifié (présence des phases sinusoïdales), il est alors possible de tester le code sur machine réelle.

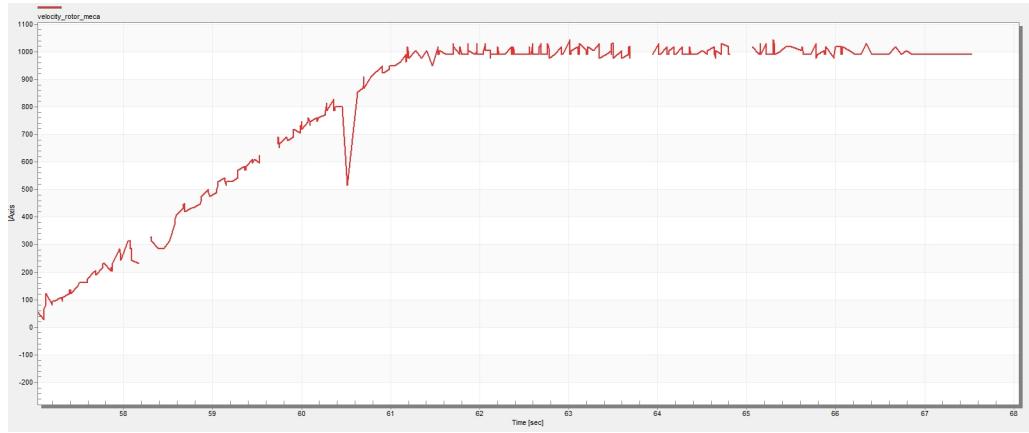


FIGURE 9 – Réponse en vitesse à une rampe de 1000RPM, $K_v = 3.5V/kRPM$

Le réponse en vitesse correspond bien à la consigne et à la simulation. Le courant de phase affiché ici n'est pas représentatif à cause d'un faible taux d'échantillonage.

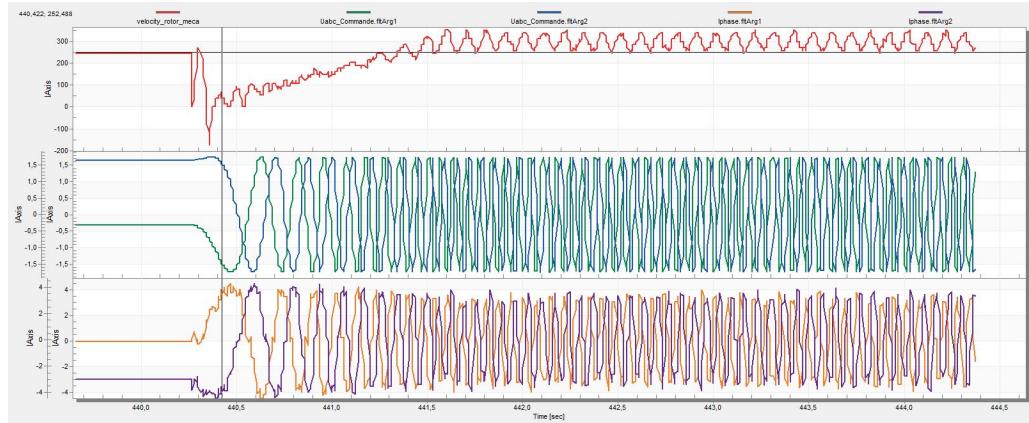


FIGURE 10 – Réponse en vitesse à une rampe de 300RPM, $K_v = 3.5V/kRPM$

On observe sur les relevés à vitesses faibles une anomalie périodique de la vitesse, correspondant probablement à des défauts de régularité de l'entrefer, causant une asymétrie du champ et donc du couple produit.

Vitesse (RPM)	30	300	1000
Erreur de vitesse & stabilité	oscillations periodiques (+/-100%)	faibles oscillations periodiques autour de 300	léger bruit mais consigne suivie
Courant phase (A)	4.1	3.8	8.2

Les vitesses d'exécution des boucles :

- Fast loop = $42\mu s$ soit 35% du temps disponible ($120\mu s$)
- Slow loop = $82\mu s$ soit 7% du temps disponible (1.2ms)

3 Mise au point de la commande FOC

On se propose d'implémenter la commande Field Oriented Control (cf schéma suivant).

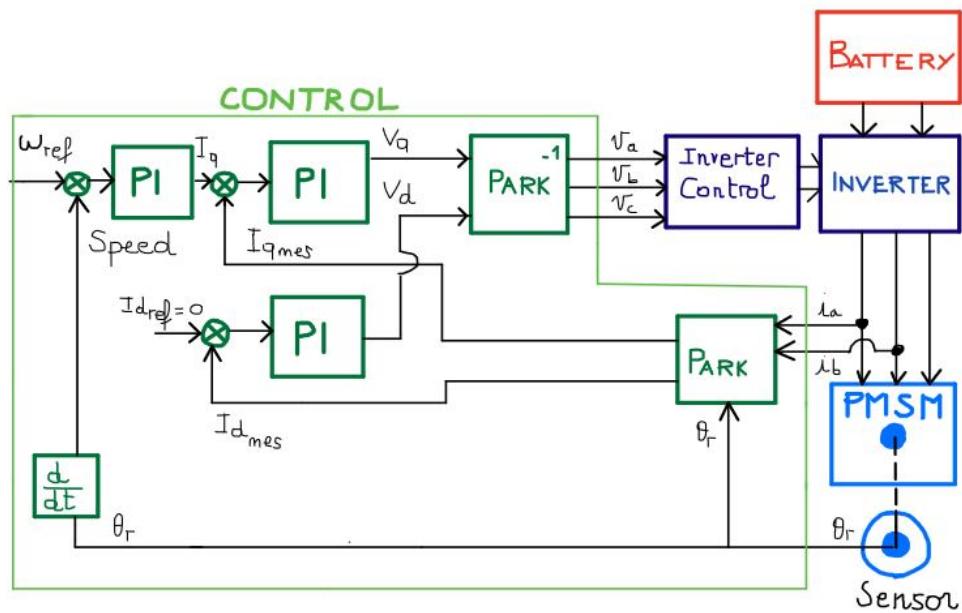


FIGURE 11 – Schéma principe commande FOC (courtesy G.Garcia)

3.1 Simulation de la commande FOC

3.1.1 Schéma bloc sous Simulink

Dans un premier temps nous avons implémenté les trois blocs correspondant aux correcteurs PI issus de la bibliothèque GFLIB. Ensuite, pour effectuer la régulation il est nécessaire de faire une transformée de Park qui permet d'obtenir les courants I_q et I_d . À partir de I_a et I_b (I_c étant calculé avec la relation : $I_a + I_b + I_c = 0$), le bloc "transformée de Clark" nous donne I_{α} et I_{β} . Il suffit de récupérer le sinus et le cosinus (bibliothèque GFLIB) de l'angle du rotor et de l'injecter avec I_{α} et I_{β} dans le bloc "transformée de Park".

Ci-dessous notre schéma bloc de la commande FOC sous Simulink :

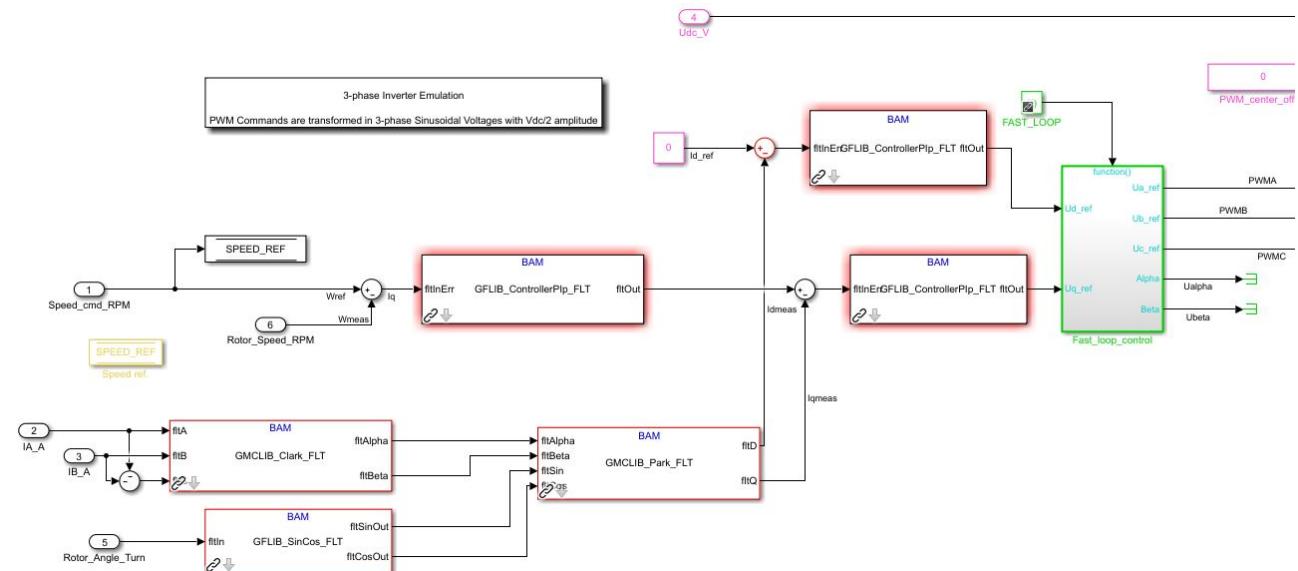


FIGURE 12 – Schéma Simulink commande FOC

3.1.2 Détermination des paramètres des correcteurs PI

CALCULS DES PARAMÈTRES DU PI RÉGULANT LE COURANT

Afin de déterminer les paramètres du correcteur de régulation de courant, on ouvre la boucle, et on simule une impulsion de tension triphasée en entrée du moteur. On observe la réponse en courant pour déterminer le retard, la pente et la valeur de gain.

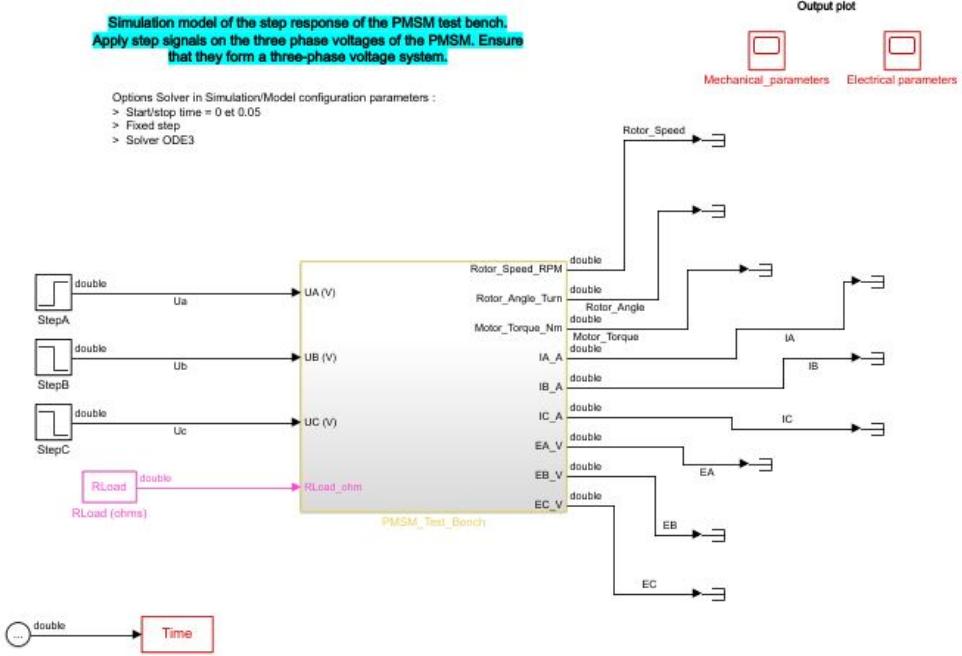


FIGURE 13 – Schéma Simulink boucle ouverte

On peut alors exploiter la méthode de Ziegler-Nichols en boucle ouverte pour déterminer les paramètres du correcteur :

$$\begin{cases} K_p = 0.9 \frac{\Delta m}{LR} = 2.3V/A \\ K_i = \frac{K_p}{\tau_i} = \frac{K_p}{3.3L} = 5.8V/A/ms \end{cases} \quad (1)$$

avec

$$\begin{cases} R = 16A/ms(\text{pente}) \\ L = 120\mu s(\text{retard}) \\ \Delta m = 5V(\text{entre}) \end{cases} \quad (2)$$

Les valeurs ainsi obtenues sont cependant souvent trop agressives pour notre système, on choisira donc de prendre une fraction de ces valeurs.

CALCULS DES PARAMÈTRES DU PI RÉGULANT LA VITESSE

Pour le correcteur PI régulant I_q en fonction de la vitesse de rotation du rotor, nous avons utilisé la méthode de Ziegler-Nichols en boucle fermée. Cette méthode consiste à tout d'abord annuler l'action intégrale du correcteur. Ensuite, on augmente l'action proportionnelle jusqu'à ce que des oscillations apparaissent. On relève K_u qui correspond au gain maximal et T_u la période des oscillations.

$$\begin{cases} T_u = 7.35ms \\ K_u = 0.11A/(rad/s) \end{cases} \quad (3)$$

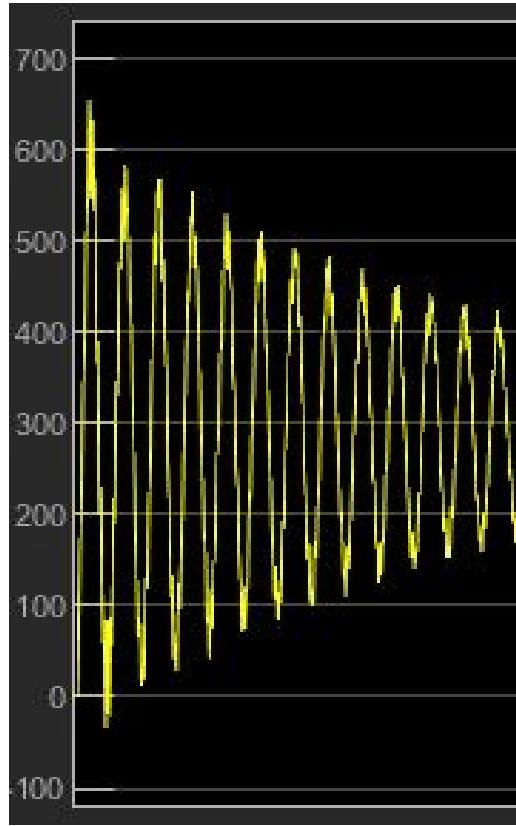


FIGURE 14 – Vitesse du rotor lorsque les oscillations soutenues apparaissent

À partir de ces deux valeurs on détermine les paramètres du correcteur PI, d'après le tableau associé à la méthode Ziegler-Nichols :

$$\begin{cases} K_p = 0.2 * K_u = 0.022A/(rad/s) \\ K_i = K_p/T_i = 5.98A/(rad/s^2) \\ T_i = 0.5 * T_u = 3.675ms \end{cases} \quad (4)$$

À partir des valeurs calculées ci-dessus et après simulations nous les avons ajustées pour améliorer les performances de notre correcteur. Nous obtenons donc les valeurs suivantes :

$$\begin{cases} K_p = 0.022A/(rad/s) \\ K_i = 2A/(rad/s^2) \end{cases} \quad (5)$$

3.1.3 Résultats de la simulation sous Simulink

La réponse temporelle du système complet présente peu de dépassement, et les courants dans chacune des phases ne sont pas trop élevés au démarrage. On constate que le courant diminue rapidement, contrairement à la commande U/F, et se stabilise vers la valeur minimale permettant de maintenir la consigne de vitesse.

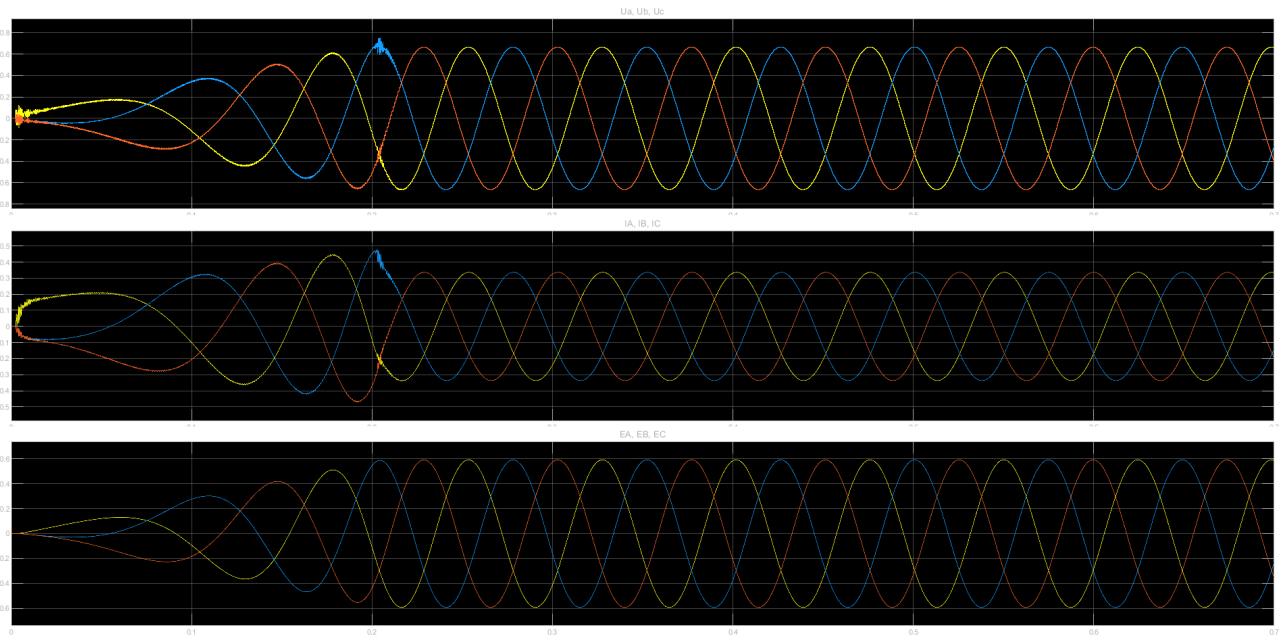


FIGURE 15 – Réponse temporelle sous Simulink en boucle fermée

3.2 Test sur carte micro contrôleur

3.2.1 Implémentation du code et simulations sur le moteur virtuel

Le code embarqué sur le microcontrôleur suit la structure développée dans le Simulink précédent. Les fonctions Exec_Fast_Loop et Exec_Slow_Loop sont réécrites pour intégrer la régulation.

En ce qui concerne la simulation sous Freemaster, les valeurs de nos coefficients de nos PI étaient trop importantes et entraînaient une divergence du système. De ce fait, nous avons réduit ces valeurs pour retrouver la stabilité du système. Ci-dessous les valeurs implémentées :

PI COURANT

- $K_p=0.05$
- $K_i=5$

PI VITESSE

- $K_p=0.08$
- $K_i=0.01$

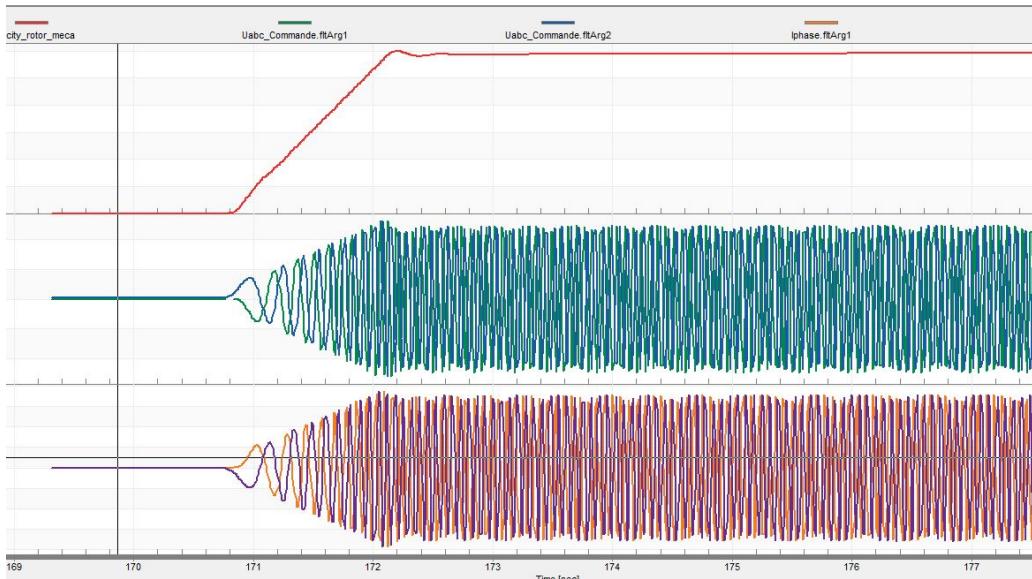


FIGURE 16 – Réponse temporelle du moteur virtuel en boucle fermée

3.2.2 Tests sur le moteur réel

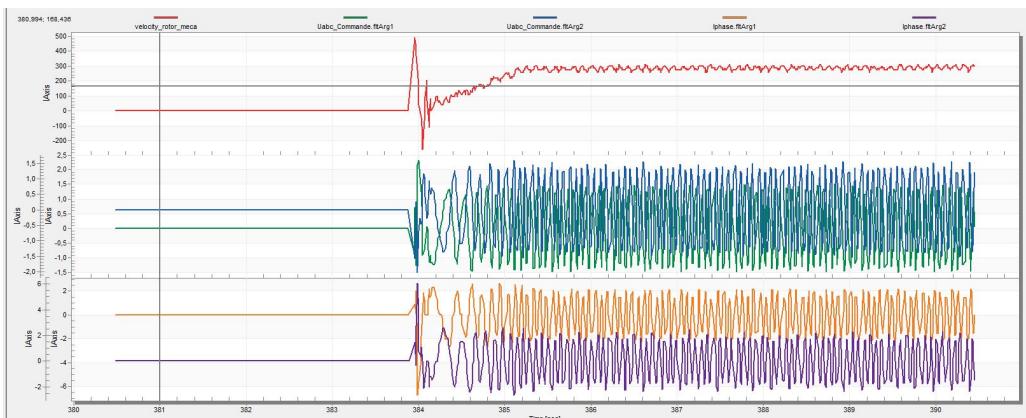


FIGURE 17 – Réponse sur moteur réel avec une consigne à 300 tr/min

La vitesse était globalement constante (environ 300 \pm 20 tr/min) même lorsqu'on augmentait le couple de charge du moteur, entraînant par conséquent une augmentation du courant.

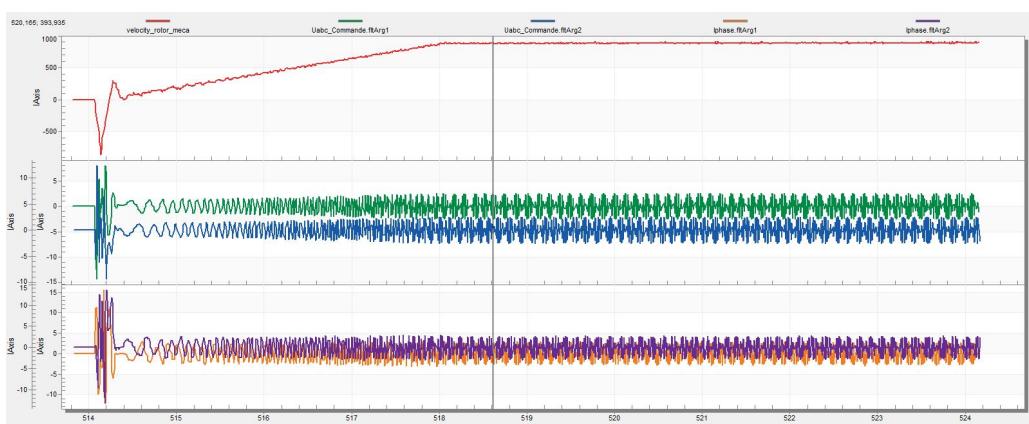


FIGURE 18 – Réponse sur moteur réel avec une consigne à 900 tr/min

- L'erreur de position pour une commande de 900 tr/min est de +/- 15%.
- Pour le temps d'exécution de la Slow Loop que nous avons mesuré est de : **80µs**
- Pour le temps d'exécution de la Fast Loop que nous avons mesuré est de : **40µs**

Dans les deux cas, le courant sans charge était bien plus faible que celui requis par la commande U sur F.

4 Analyse Safety

On se propose d'analyser l'architecture de commande du moteur du point de vue safety de fonctionnement. Dans ce cadre, nous traiteront le problème suivant : **Collage permanent d'une phase du moteur à '0', à '1' ou en état haute impédance**

4.1 Description de la défaillance, conséquences et safe states

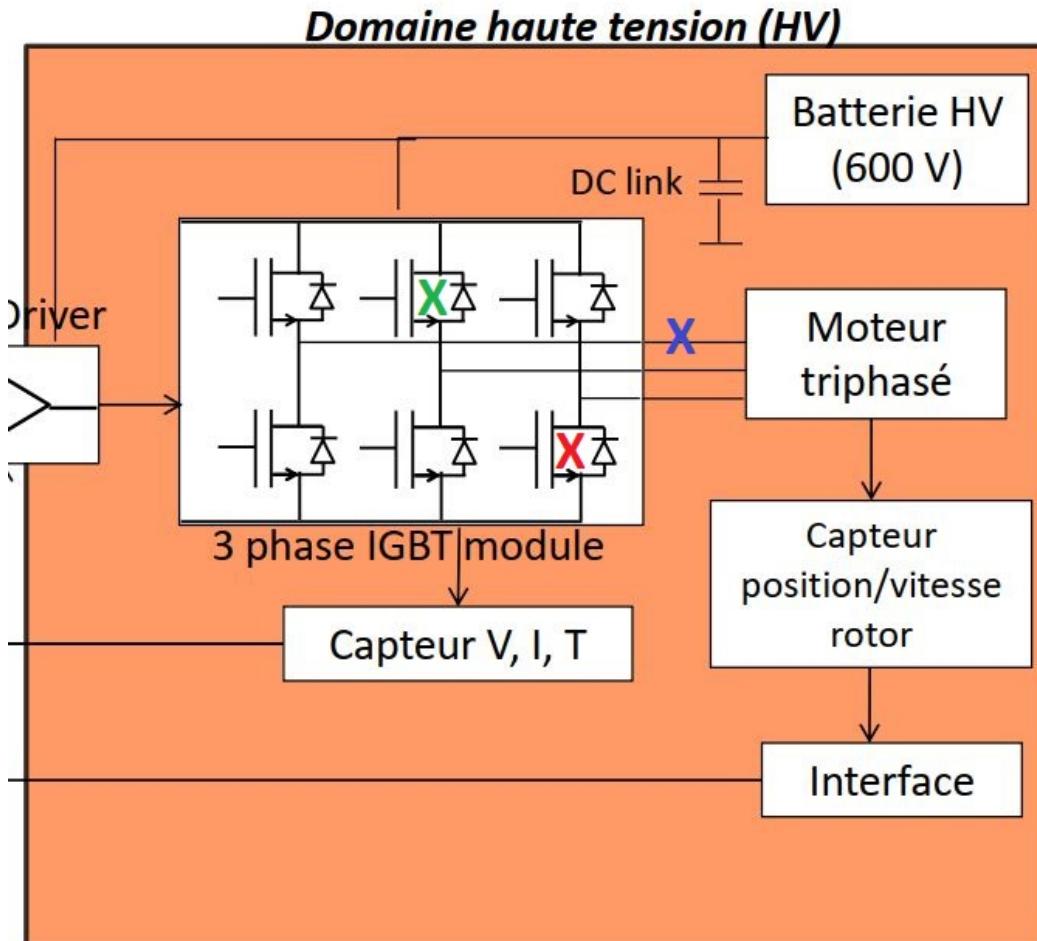


FIGURE 19 – Schéma du domaine à haute tension

Exemples de défaillances à l'origine du problème choisi ci-dessus.

X : Si ce High-side MOS est en court-circuit alors une des phases du moteur est collée à '1'.

X : Si ce Low-side MOS est en court-circuit alors une des phases du moteur est collée à '0'.

X : Si un circuit ouvert se crée (rupture d'un enroulement moteur suite à la fatigue/ aux vibrations), l'une des phases est en état de haute impédance.

Les conséquences :

- La commande triphasée complète ne peut plus être assurée.
- Possible court-circuit de la batterie (Shoot-through condition) : Si le low-side MOS est activé en cas de collage à '1' OU le high-side MOS en cas de collage à '0'.

Selon les modes de défaillance, le safe state n'est pas le même :

- Collage à '0' : on active les Low-side MOS, pour que le moteur soit en free-wheeling. Risque de blocage du moteur ou de shoot-through si les High-side MOS sont activés.
- Collage à '1' : on active les High-side MOS, pour que le moteur soit en free-wheeling. Risque de blocage du moteur ou de shoot-through si les Low-side MOS sont activés.
- Haute impédance : on peut tenter de continuer la commande sur les deux phases restantes, en compensant le manque de couple dû à la phase déconnectée. Cela peut être utile pour permettre au véhicule de ralentir progressivement si il était à grande vitesse ou de pouvoir continuer le déplacement à faible vitesse.
Cependant cette commande à deux phases est complexe à mettre en place.

4.2 Mécanismes de détection des défaillances

Dans un premier temps, nous nous sommes concentrés sur la détection des collages à '1' ou à '0' de l'un des MOS. Pour cela, le Gate Driver embarque un détecteur de saturation. Ce détecteur est présent pour chaque phase, c'est un comparateur qui permet de relever une erreur provenant soit d'un court circuit à la masse ou au V_{Supply} , en mesurant le V_{DS} du MOS mis en conduction.

Lorsque le MOS Low-Side est court-circuité (à la masse), alors le V_{sat} du MOS High-side (cf Figure ci-dessous), est supérieur au seuil de 1,4V. De ce fait, le montage comparateur, après une période de deadtime et de blanking, lève le flag "Desaturation error". Ces délais présents pendant le transitoire sont importants à prendre en compte dans l'algorithme de détection, le deadtime delay pouvant atteindre la dizaine du μs en fonction des réglages.

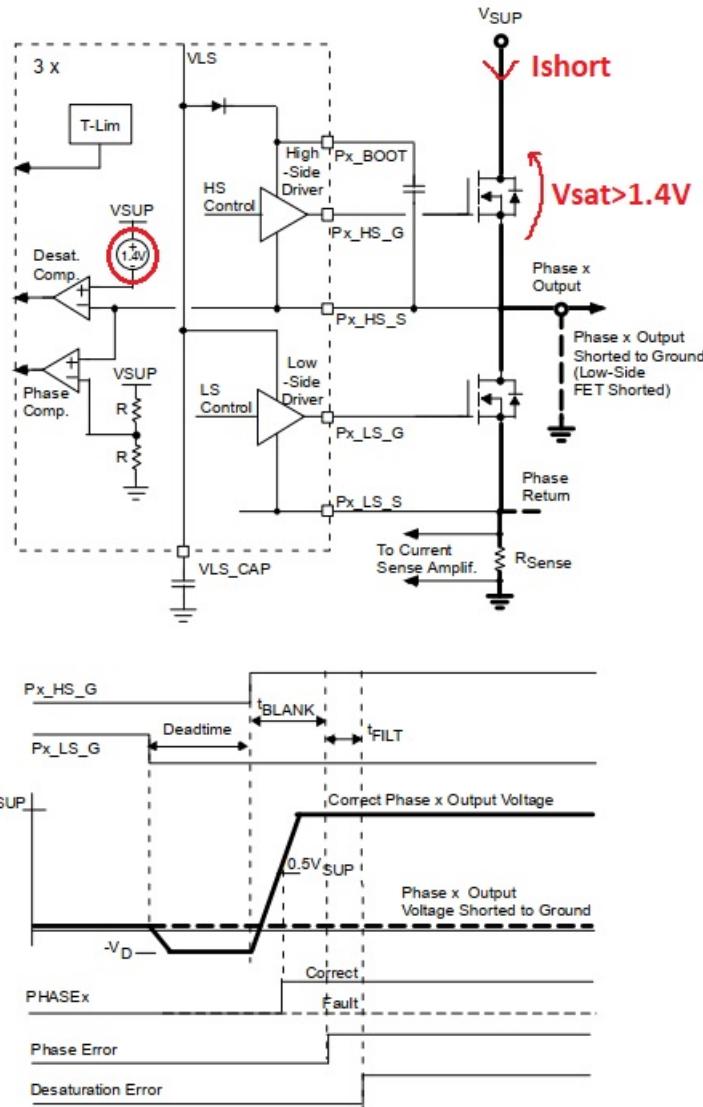


FIGURE 20 – Extrait de la datasheet du Gate Driver GD3000 présentant le principe de fonctionnement du détecteur de désaturation lors d'un court-circuit du MOS Low-Side.

Le gate driver comprend également une fonctionnalité de détection d'erreur de phase. Cette détection opère en comparant la consigne de pilotage (par exemple LOW-side MOS ON) à la tension du noeud Phase (dans l'exemple cette tension est attendue à V_{ss}). La mesure s'effectue par rapport à $\frac{1}{2}V_{sup}$. De même que pour la mesure de desat, un temps de blanking, en plus du dead time est appliqué pour obtenir des mesures correctes. Le temps de détection est cependant très critique, car en cas de court-circuit d'un MOS, les courants en jeu augmentent très rapidement, et le MOS désaturé peut être dégradé, car il doit absorber une forte puissance pendant cet intervalle.

Pour détecter une phase en circuit ouvert, le courant dans cette phase est une bonne indication de sa connexion. Ainsi, si le courant mesuré est inférieur à d'une certaine plage de valeurs attendues, on peut détecter cette défaillance. Comme pour la mesure de desat, il est important de valider cette défaillance, de s'assurer quelle ne provient pas d'une erreur de mesure, afin de ne pas changer de mode de commande de manière inappropriée.

4.3 Algorithme de détection des défaillances

La détection des défaillances et le passage dans un safe state doit se faire en moins de 100ms d'après l'énoncé. Cependant, en réalité les temps de réponse peuvent être spécifiés inférieurs à la μ s, pour limiter l'énergie absorbée par les MOS. Par exemple, pour une application automobile en 600V, en cas de court-circuit, le courant peut atteindre 5kA en moins de 1 μ s, d'où une énergie dans le MOS de quelques Joules.

Pour cette défaillance, la détection pourrait être faite via la mesure du courant. Si aucun courant ne circule, l'une des phases est donc en état de haute impédance. Dans ce cas on peu passer dans un état de fonctionnement dégradé où le moteur est commandé par seulement deux phases.

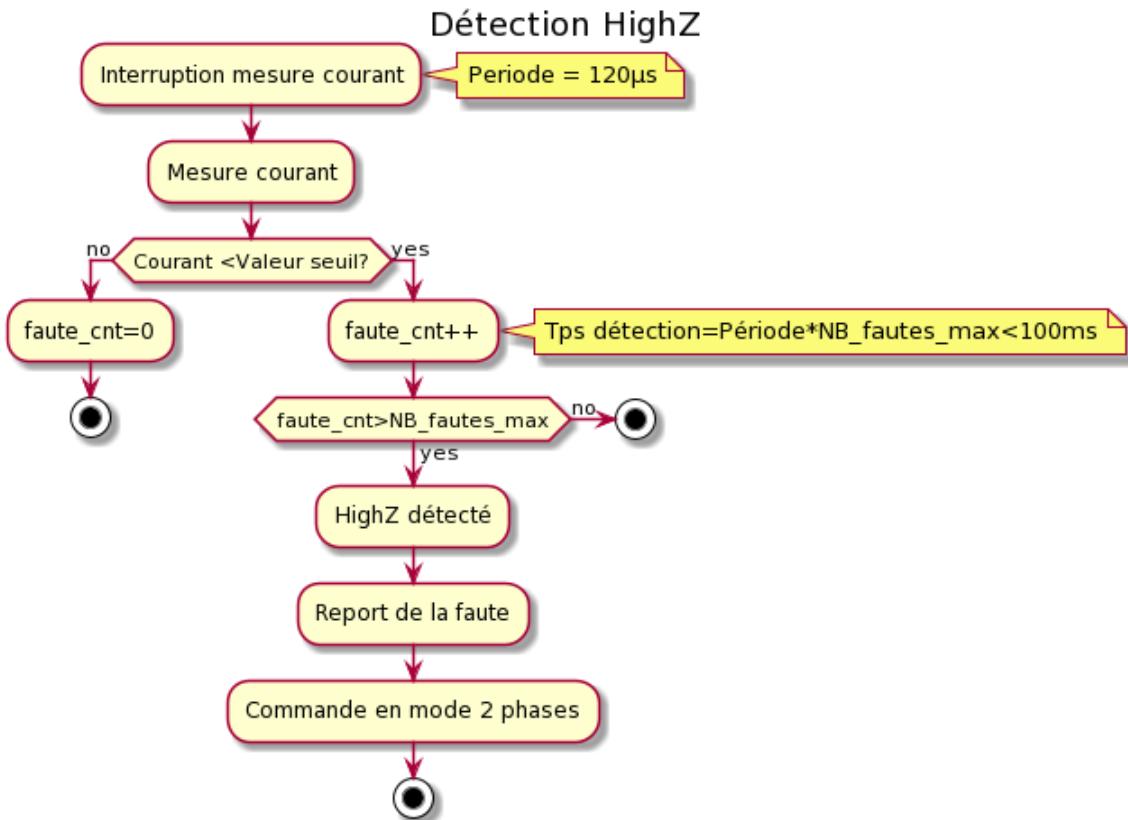


FIGURE 21 – Algorigramme de la détection du défaut d'une phase en "High Z"

Pour cette détection, le détecteur de saturation renvoie directement un flag en cas d'erreur. En fonction de l'erreur détectée collage à 1 ou à 0, la priorité est d'abord de mettre les autres phases dans le même état. Ici, la commande du moteur devient impossible et un des safe states envisageable est de passer en roue libre.

Détection Collage 0/1

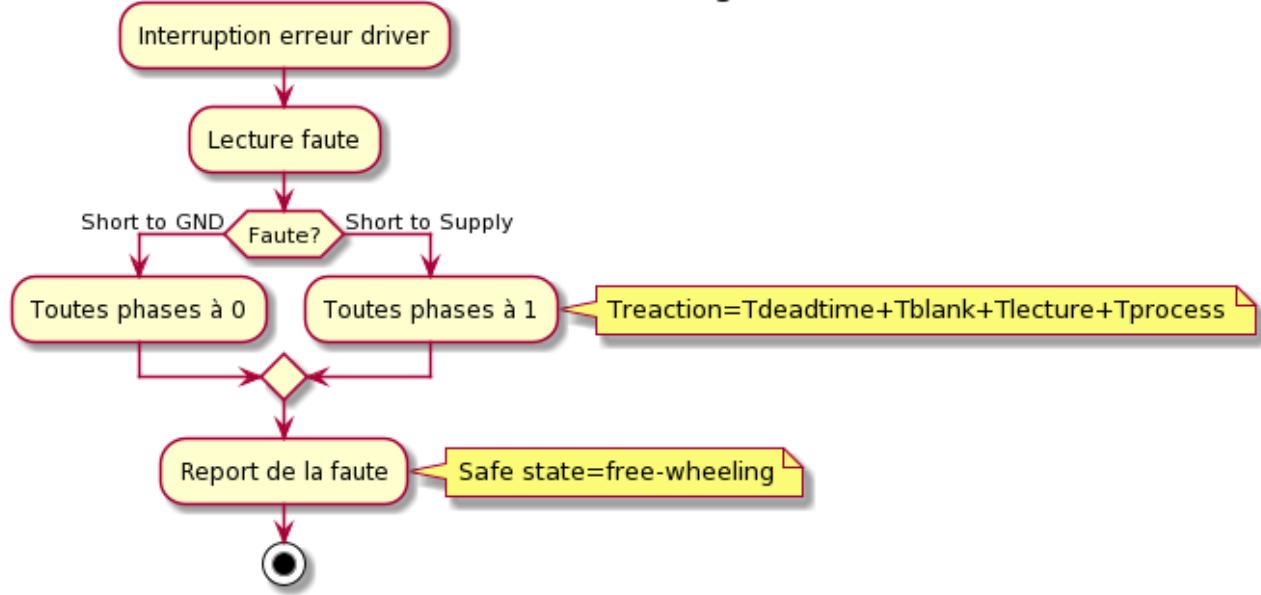


FIGURE 22 – Algorigramme de la détection du défaut d'une phase collée à '1' ou à '0'

5 Conclusion

Ce BE nous a permis d'expérimenter avec deux commandes moteurs, et d'évaluer leurs complexités d'implémentation et leurs performances respectives. Nous avons pu analyser les architectures de commande, et tenter de poser un regard safety sur ces architectures. Des solutions de mise en safe state ont été proposées.