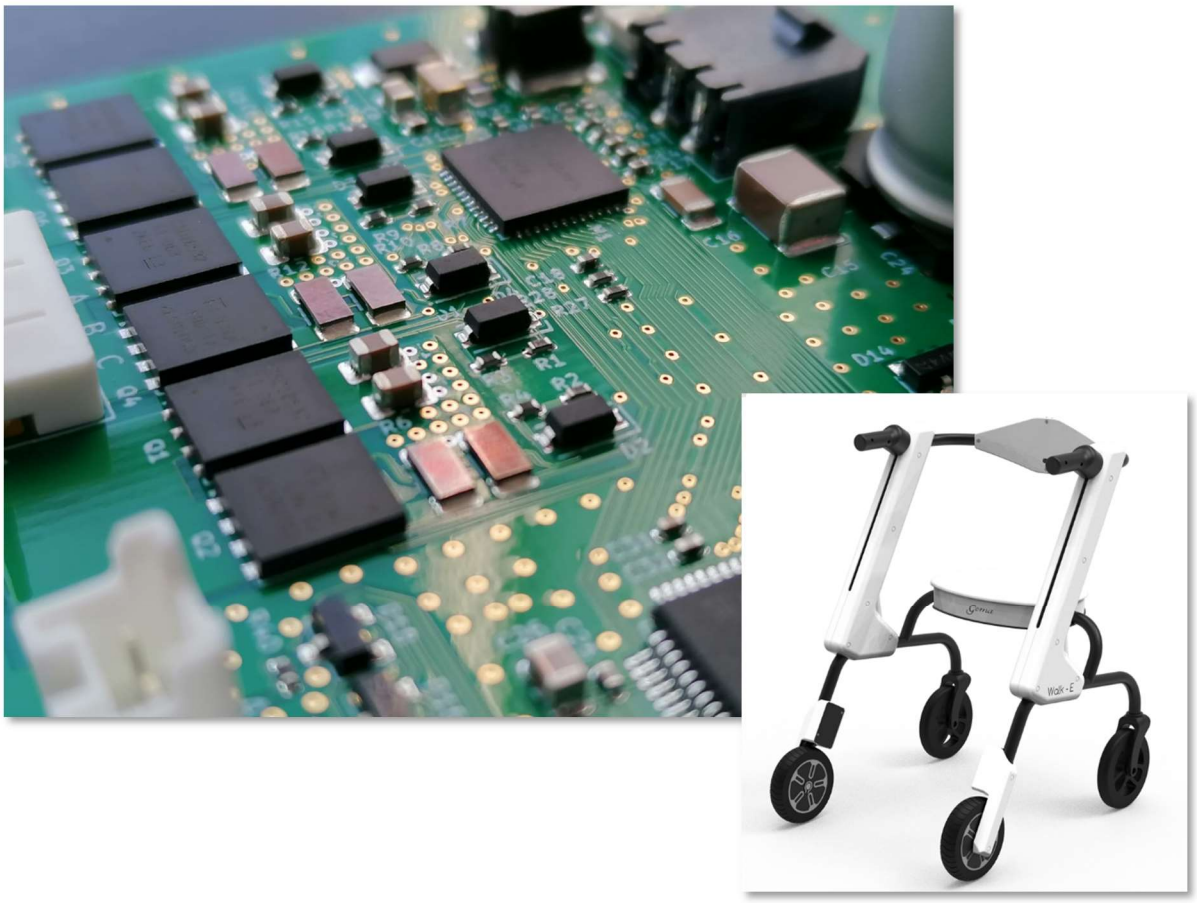
ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

Gema
Walk-E. Expand your life

# Développement de contrôleurs de moteur BLDC pour le robot Walk-E



## Rapport de projet industriel


*Tuteur pédagogique :* Rachid ZEBOUDJ

*Maître d'apprentissage :* Sébastien OSENAT

**MARTIN RYS**

**3DN - 2022/2023**

# 2 Conception

## 2.1 Synoptique

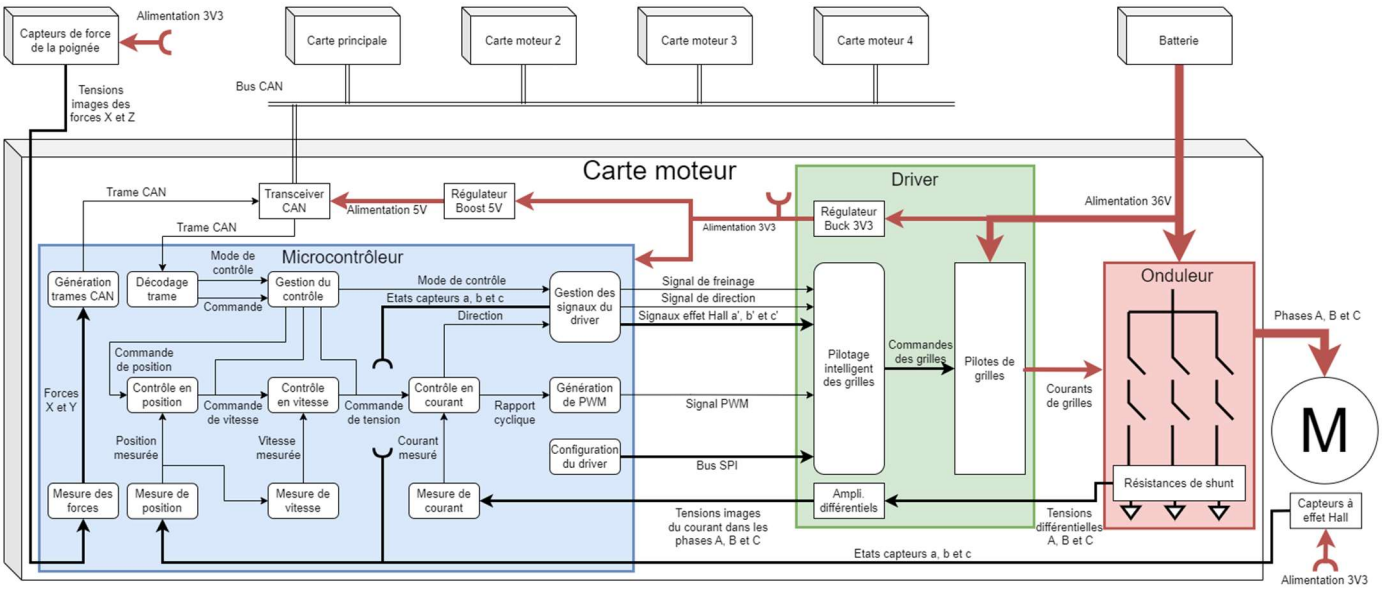



*Figure 1 Schéma synoptique de la carte moteur*

La carte est composée de trois éléments principaux. L'onduleur, composé de six MOSFET, permet d'alimenter les phases du moteur. Le driver de MOSFET permet de gérer les tensions des grilles des MOSFET, il apporte la puissance nécessaire. Le composant choisit apporte également quelques fonctionnalités intelligentes supplémentaires comme l'insertion de temps morts, la commutation de phase automatique et détection d'erreur.

Le microcontrôleur reçois les informations des capteurs à effet Hall du moteur, les informations de courant dans les phases et des informations de commande via le bus CAN. C'est lui qui gère les boucles de contrôle permettant de piloter le moteur en position, en vitesse et en tension/courant.

La carte peut également être connectée à une des poignées du robot selon son rôle. Les capteurs d'effort présent dans les poignées envoient une tension proportionnelle aux forces sur deux axes. La carte doit lire et filtrer les valeurs de force avant de les transmettre à la carte principale par via le bus CAN.

## 2.2 Moteur BLDC

### Différences par rapport au moteur DC

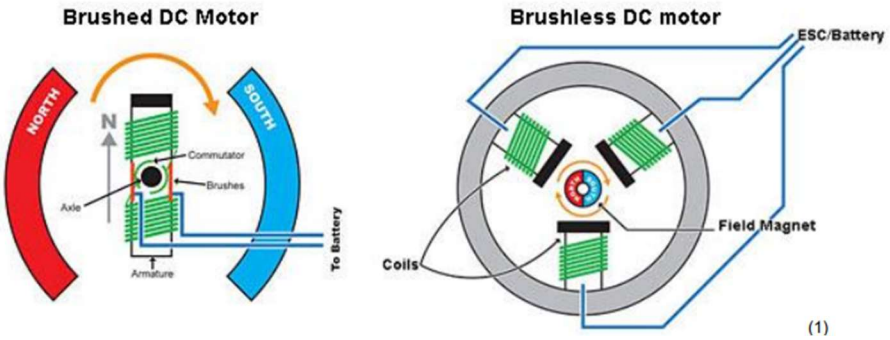



*Figure 2 Schéma d'un moteur à balais (à gauche) et d'un moteur sans-balais (à droite). Texas Instruments, Demystifying BLDC motor commutation.*

Les moteurs utilisés dans le projet sont des moteurs BLDC (brushless DC), Ils sont de la famille des moteurs synchrones à aimants permanents. Pour expliquer leur principe de fonctionnement simplement, on peut le voir comme un moteur à courant continu classique mais pour lequel la commutation de la bobine est réalisée de manière électronique et non mécaniquement par des balais, d'où le nom de brushless. Cette commutation électronique rend le système plus complexe mais plus robuste dans le temps car l'usure liée au frottement des balais n'existe plus. Cela permet également un contrôle plus précis que nous verrons par la suite.

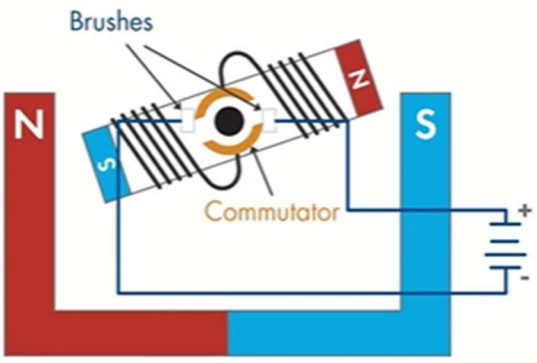



*Figure 3 Commutation mécanique dans un moteur à balais. MathWorks, Introduction to Brushless DC Motor Control.*

Pour un moteur DC, la bobine tourne et les aimants sont fixes. A chaque demi-tour de la bobine, les pads du commutateur mécanique entrent en contact avec l'autre charbons, ce qui inverse le sens de circulation du courant dans la bobine et donc la position des pôles nord et sud. Le pôle Nord de la bobine étant attiré par le pôle Sud de l'aimant (en vice-versa) sur la rotation continue.

## Alimentation des phases

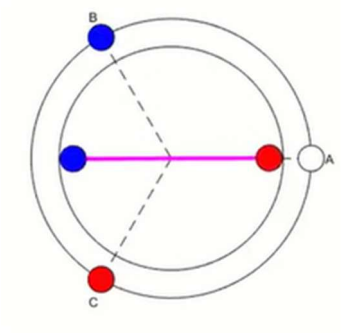



*Figure 4 Représentation des éléments magnétiques dans un moteur sans balais. MathWorks, Introduction to Brushless DC Motor Control.*

Dans le cas d'un moteur brushless, c'est l'aimant qui tourne. Les pôles magnétiques le mettant en rotation sont des bobines placées autour de l'aimants. Sur le schéma ci-dessus, les points A, B et C sont des bobines dont on pilote le courant. La ligne rose représente l'aimant tournant et ses pôles Nord (en rouge) et Sud (en bleu). Au lieu d'inverser les pôles magnétiques de l'élément tournant, on inverse la polarité des pôles fixes. Cela revient au même mais puisque les bobines sont fixe on peut les connecter avec un simple câble sans être gêné par leurs mouvements.

De la même manière que pour le moteur DC, en fonction de la position du rotor (élément tournant) par rapport au stator (partie fixe), les pôles magnétiques (donc le sens du courant) des bobines doivent être modifiés pour attirer le pôle opposé de l'aimant permanent et le mettre en mouvement. Ainsi, ici le pôle Sud (point bleu) du rotor est entre les pôles B et C, si on veut faire tourner le rotor en sens anti-horaire on va générer un pôle Nord sur la bobine C et un pôle Sud sur la bobine B. Cela va attirer le pôle Sud de l'aimant vers la bobine C. La bobine A n'est pas utilisée à cet instant, on la laisse dans un état déconnecté (haute impédance).

Quand le rotor s'est déplacé, avant que le pôle Sud n'atteigne la bobine C, on déplace le pôle Nord du stator sur la bobine A et on déconnecte la bobine C. Ainsi la rotation peut continuer et le rotor ne reste pas bloqué dans son alignement magnétique (Nord en face d'un Sud). Avec trois bobines pouvant être Nord, Sud ou déconnectées, on obtient six positions différentes dans lesquelles le rotor peut s'aligner. Sans changement d'état des bobines le rotor se bloque dans ses postions, il faut donc passer d'une position à la suivante au bon moment pour que la vitesse du moteur soit constante.

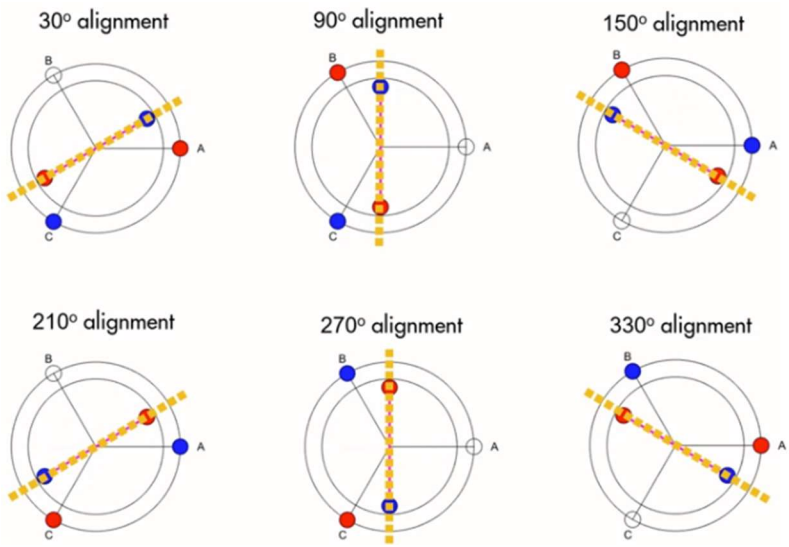



*Figure 5 Position du rotor auxquelles modifier l'état des pôles du stator. MathWorks, Introduction to Brushless DC Motor Control.*

On trouve 6 positions décalées de 60° auxquelles changer la polarité des bobines. On peut dire qu'il y à trois positions, avec deux polarités pour chaque. La difficulté est maintement de devoir tenir compte de la position magnétique du rotor. On parle ici de position magnétique car il s'agit de la position d'un pôle du rotor par rapport aux groupe de trois pôles du stator. En effet, les schémas présentés ici sont pour une paire de pole aux stator mais généralement on touve de nombreuses paires de pôle.

On cummutera toujours tous les 60° magnétiques, qui correspondent à 60/N ° « physique », N étant le nombre de paires de pôles. Tout les pôles sont synchronisés du point de vue magnetique, il suffit donc de connaitre la position magnétique d'un pôle pour les connaitre toutes. On a donc toujours trois positions à détécter (triangles jaunes) et la polarité à ces positions.

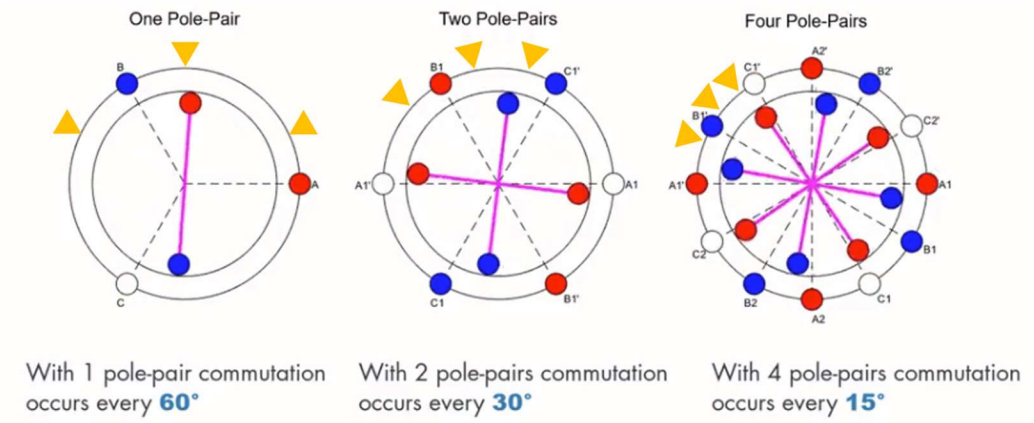



*Figure 6 Schéma d'un moteur BLDC pour 1, 2 et 4 paires de pôles. Modifiée à partir d'une image de MathWorks, Introduction to Brushless DC Motor Control.*

Pour détecter la position du rotor, la plupart des moteurs BLDC intègrent des capteurs à effet Hall espacés de 120° magnétiques. En résumé, ces capteurs affichent un 1 logique en sortie si un pôle Nord (ou Sud selon le sens du capteur) est dans son demi-cercle.

On appelle secteur les espaces entre les 6 positions décrites précédemment. Les capteurs à effet Hall sont centrés par rapport à 3 de ces secteurs. En lisant l'état des trois capteurs on peut déterminer dans quel secteur se trouve les pôles magnétiques du rotor, et de fait, savoir comment alimenter les bobines selon le sens de rotation voulu.

Du fait de leur positionnement, seulement 6 combinaisons de capteurs sont possibles car les trois capteurs ne peuvent pas être à 0 ou à 1 en même temps. On peut donc établir une table de vérité associant l'état des capteurs à effet Hall et le sens de rotation désiré, à l'alimentation de chaque bobine. Ce mode de fonctionnement est appelé 6-step car un tour se fait en 6 étapes.

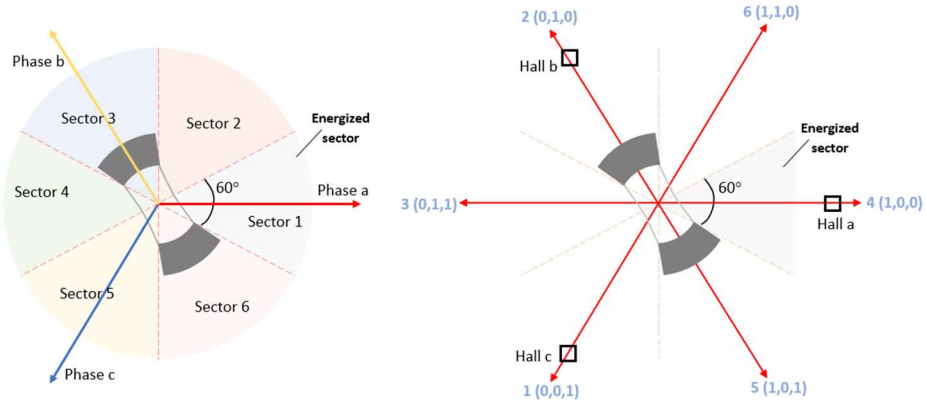



*Figure 7 Positionnement des capteurs à effet Hall et répartition des secteurs dans un moteur BLDC. MathWorks, Six-Step Commutation of BLDC Motor Using Sensor Feedback.*

L'ordre et la synchronisation de l'alimentation des bobines étant résolus on peut désormais piloter ce moteur comme un moteur DC classique en faisant varier les tensions aux bornes des bobines selon la vitesse désirée. Les capteurs à effet Hall permettent également de mesurer la vitesse du moteur en comparant le temps passé dans chaque secteur.

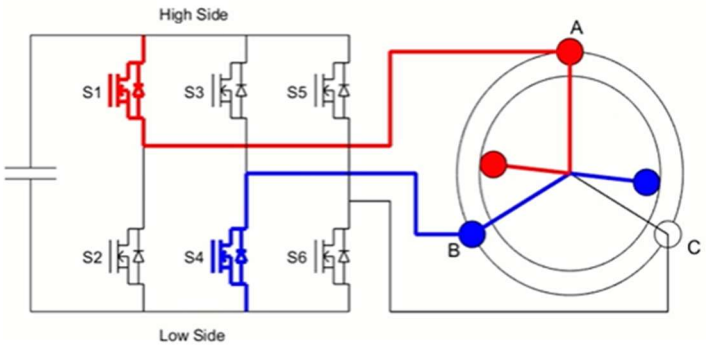



*Figure 8 Onduleur triphasé alimentant deux bobines. MathWorks, Introduction to Brushless DC Motor Control.*

Dans notre cas, les bobines sont reliées entre-elles en étoiles, le courant entre par une bobine et ressort par une autre. Chaque bobine est alimentée par une branche de pont. En appliquant un signal PWM, on peut ajuster précisément la tension appliquée à la bobine. Le courant entre par une bobine et ressort par une autre, ainsi le courant circule dans un sens dans un bobine et dans l'autre sens dans la deuxième. Les bobines se retrouvent donc magnétisées dans deux sens différents ce qui crée un pôle Nord et un pôle Sud.

## 2.3 Onduleur triphasé

### Alimentations des phases

De la même manière qu'un pont en H est utilisé pour contrôler le courant dans un moteur à courant continue, on utilise le même principe pour alimenter chacune des bobines du moteur BLDC. La différence étant qu'il y a maintenant 3 bobines à alimenter dans un ordre particulier. Les bobines sont connectées entre-elles en étoile : chaque bobine possède une borne accessible depuis l'extérieur (la phase) et l'autre est relié aux autres bobines. Ainsi, le courant est obligé de circuler à travers au moins 2 bobines en même temps.

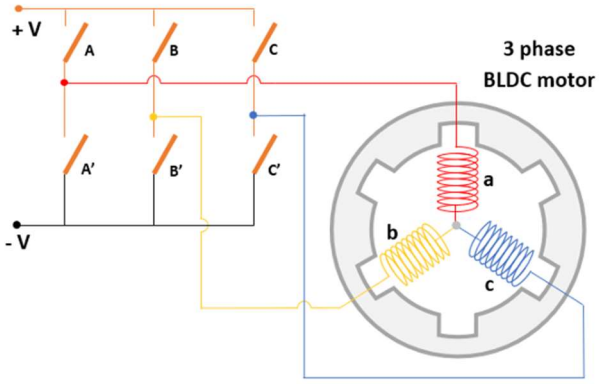



*Figure 9 Schéma d'un moteur BLDC et son onduleur triphasé. MathWorks, Generate switching sequence for six-step commutation of brushless DC (BLDC) motor.*

En fonctionnement 6-step, une phase est reliée à la masse et l'autre à l'alimentation pendant que la troisième est laissée flottante. Ainsi le courant ne circule qu'à travers 2 bobines. On pourra ajuster la valeur du courant en hachant la tension d'alimentation afin de la réduire.

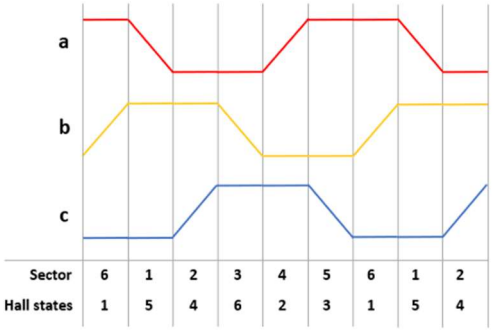



*Figure 10 Niveaux de tension appliqué aux phases en fonction de la position du rotor. MathWorks, Generate switching sequence for six-step commutation of brushless DC (BLDC) motor.*

Si on suppose que notre système est capable de lire les capteurs à effet Hall et d'appliquer le signal PWM au bon MOSFET, Il ne reste plus qu'à faire varier la vitesse du moteur en ajustant le rapport cyclique du signal PWM pour faire passer plus ou moins de courant.

## Contrôle des grilles

Étant donné la conception de l'onduleur, il est primordial que deux MOSFET d'une même branche ne soit jamais passant en même temps car cela créerait un court-circuit entre l'alimentation et la masse ce qui serait très dangereux. Pour éviter cela et empêcher les erreurs humaines d'endommager ou de détruire le système, il existe des contrôleurs de demi-point qui pilote les grilles des MOSFET.

Les grilles agissent comme des capacités, lorsqu'on leur applique un courant elle se charge petit à petit ce qui fait monter leur tension (tension grille-source). Plus la capacité de la grille est chargée, plus la résistance vue entre le drain et la source du MOSFET est faible jusqu'à devenir nulle quand la grille est chargée. Quand la tension grille source est nulle, la résistance drain-source deviens infinie et aucun courant ne circule.

Le MOSFET peut être vue comme ayant 3 états : bloqué passant ou résistif. Dans l'état bloqué, une tension est visible entre drain et source mais aucun courant ne circule donc aucune puissance n'est dissipé par la MOSFET, il ne chauffe pas. Il en est de même quand il est passant car le courant passe mais la résistance est nulle (extrêmement petite) donc la tension drain source est nulle. Dans l'état résistif en revanche du courant circule mais une tension est également visible ce qui fait qu'une puissance est dissipée et le MOSFET commence à chauffer. On génère donc des pertes d'énergie. L'objectif est d'utiliser le MOSFET dans les états passant et bloquant en raccourcissant au maximum le temps passé à l'état résistant pendant les commutations.

Les contrôleurs de demi-pont permettent de contrôler le courant entrant et sortant de la grille, cela permet de choisir le temps d'ouverture et de fermeture des MOSFET. Un élément primordial de ce contrôle est que la grille du MOSFET ne doit pas commencer à être chargée tant que celle de l'autre MOSFET de la banche n'est pas entièrement déchargée. Cela créerait un court-circuit et les MOSFET pourrait commencer à chauffer, voir griller. Pour éviter cela, en plus de pouvoir régler le temps de commutation des MOSFET (à travers le courant de grille) on peut ajouter un temps-mort entre l'ouverture d'un MOSFET et la fermeture du suivant.

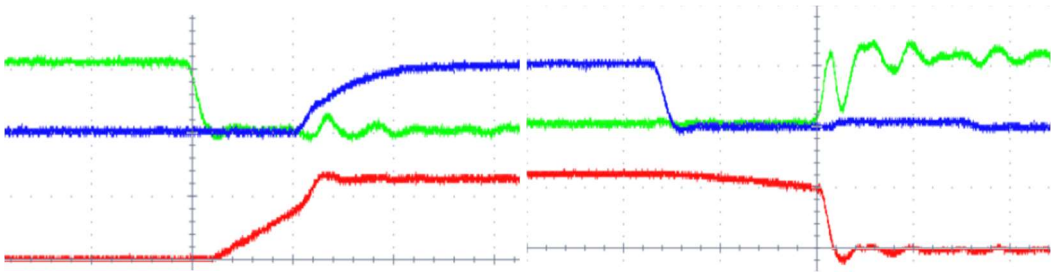


*Figure 11 Tension de grille du MOSFET haut (en bleu) et bas (en vert) au moment d'une commutation avec un temps morts. En rouge la tension de la phase. Texas Instruments, DRV835x 100-V Three-Phase Smart Gate Driver datasheet (Rev. A).*

A chaque période, la tension moyenne aux bornes de deux phases du moteur sera égale à la tension d'alimentation multipliée par le rapport cyclique du signal PWM.

## 2.4 Asservissement en courant

### Boucle de contrôle

Le moteur BLDC, comme les moteurs DC, tirent plus de courant pour générer du couple et tourne à une vitesse proportionnelle à leurs tensions.

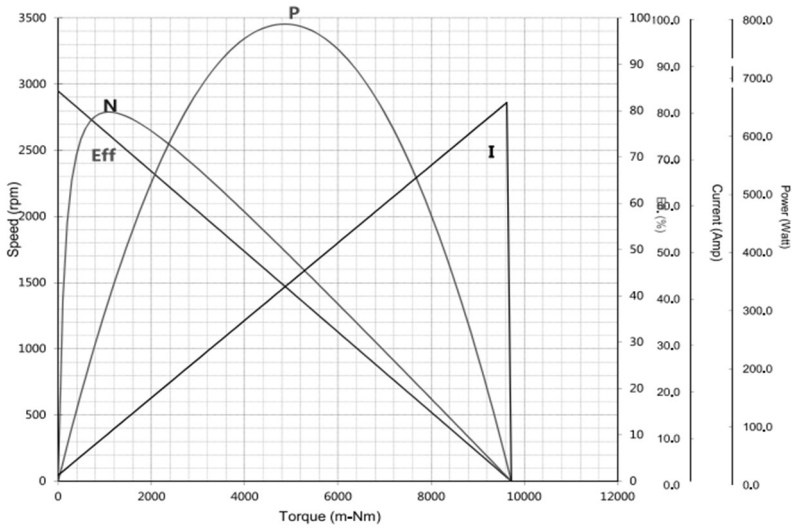



*Figure 12 Courbe de performance du moteur-hub utilisé pour les roues du Walk-E. Fiche technique du moteur Infoeel 8456, 2,5 pouces-RBE-533236-005X*

Autrement dit, un système de commande sous-dimensionné pourra certainement faire tourner un moteur très vite sans difficulté, mais ne pourrait pas fournir le courant suffisant si le moteur doit fournir du couple, même à basse vitesse. Pire encore, si aucune sécurité n'est mise en place le courant sera quand même appelé dans le moteur et les MOSFET, qui ne seront pas prévus pour une telle intensité, seront détruit.

Pour éviter cela, il nous faudra mesurer constamment le courant dans chaque phase pour s'assurer qu'il ne dépasse pas les valeurs maximales tolérées par les composants. Il s'agit ici de la première boucle de commande. On ne peut pas piloter directement le courant dans les phases, mais uniquement la tension à leurs bornes à travers le rapport cyclique du signal PWM. Le courant n'est pas proportionnel qu'à la tension. A tension fixe, si un faible couple est requis pour faire tourner le moteur, le courant demandé sera faible, et inversement si le moteur doit fournir du couple.

Ainsi on va chercher à asservir le courant. Pour cela, il faut mesurer le courant dans chaque phase et le comparer avec le courant demandé, la différence entre les deux est l'erreur sur laquelle nous appliquons un PID. Le résultat donne le nouveaux courant à appliquer. On calcul ensuite l'impédance actuelle du moteur en divisant la tension appliquée précédemment par le courant actuelle. En multipliant l'impédance par le courant calculé, on trouve la nouvelle tension à appliquer au moteur, et donc le rapport cyclique du moteur. Cette boucle doit idéalement être exécutée à la même fréquence que le signal PWM pour pouvoir ajuster le rapport cyclique à chaque période. Cela n'est pas toujours réalisable, dans ce cas on mesure le temps de monté du courant dans le moteur à rotor bloqué. C'est ce temps qu'il faudra respecter entre deux exécutions du contrôle du courant.

La mesure du courant est réalisée en utilisant des résistances de shunt qui sont des résistances précises et de très faible valeur (de l'ordre du milliohm). Lorsqu'elles sont traversées par un courant, une tension proportionnelle (U = R*I) apparait à ses bornes. La tension est très faible et doit être amplifiée pour être lu par l'ADC du microcontrôleur.

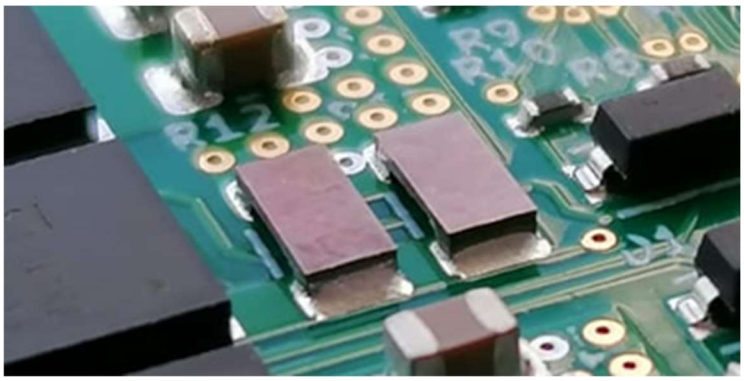



*Figure 13 Photo de deux résistances de shunt branchées en parallèle sur la carte moteur.*

Lors de la commutation des MOSFET un appel de courant est visible et peut fausser la mesure. Une première solution consiste à synchroniser les mesures avec les commutations des MOSFET, c'est faisable lorsque le microcontrôleur gère directement les commutations, mais ici on ne sait pas exactement quand le driver de MOSFET effectuera la commutation. Une autre solution consiste à mesurer en continu en appliquant un filtre médian qui est beaucoup plus efficace qu'une moyenne pour éliminer les valeurs très éloignées. C'est cette méthode qui a été choisie.

A l'aide d'une DMA les valeurs de l'ADC sont stockées en mémoire sans utiliser de temps processeur. Au moment de l'appelle de la tâche de mesure de courant, les médianes sont calculées et les valeurs des courants sont mise à jour en mémoire pour être utilisé par la boucle de contrôle du courant.

La résistance est choisie en fonction de sa valeur en tenant compte du courant maximal à mesurer, du gain de l'amplificateur différentiel et de la plage de mesure de l'ADC. Il faut éviter de prendre une valeur trop grande pour limiter la dissipation de puissance et donc la taille du composant. L'amplificateur différentiel est intégré au driver et son gain est configurable par SPI.

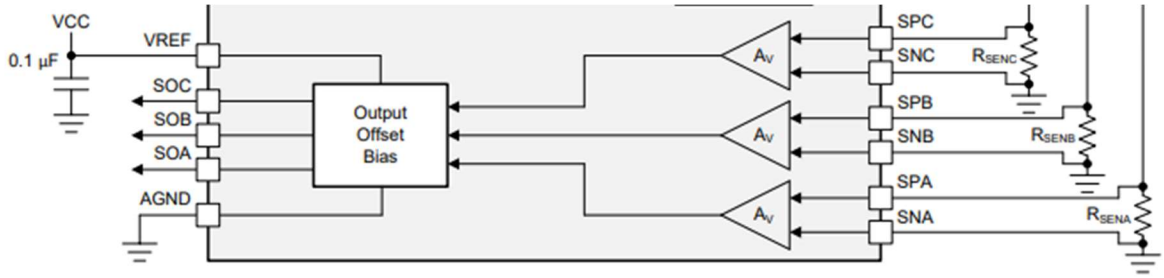



*Figure 14 Schéma bloque montrant les amplificateurs differentiels intégrés au driver. Texas Instruments, DRV835x 100-V Three-Phase Smart Gate Driver datasheet (Rev. A).*

## 2.5 Contrôleur de grille

### Fonctionnement général

Le driver DRV8353RS de Texas Instrument utilise une architecture de commande de grille intelligente qui permet de réduire le nombre de composants externes généralement nécessaires pour le contrôle du temps de montée des MOSFET et pour les circuits de protection.

L'architecture optimise également le temps mort pour éviter les conditions de recouvrement (deux MOSFET ouvert en même temps), permet une réduction des interférences électromagnétiques (EMI) grâce au contrôle du temps de montée des MOSFET et protège contre les courts-circuits de grille grâce à une surveillance de la tension. Un circuit de tirage fort de la grille aide à prévenir les allumages parasites de la grille lors des variations de tension rapides par exemple (dV/dt).

Le driver intègre également trois amplificateurs différentiels permettant d'amplifier la tension des résistances de shunt mesurant le courant des phases. Le gain est réglable par SPI de 5V/V à 40V/V. Un régulateur Buck LM5008A est également intégré au driver, nous l'avons réglé à 3.3V car la majorité de la consommation a lieu à cette tension.

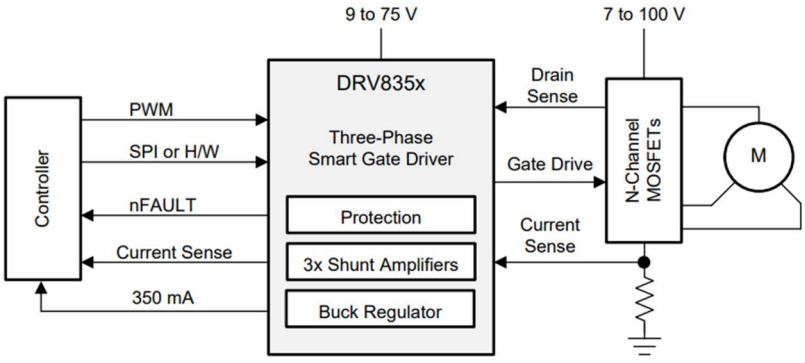



*Figure 15 Schéma simplifié du DRV8353RS. Texas Instruments, DRV835x 100-V Three-Phase Smart Gate Driver datasheet (Rev. A).*

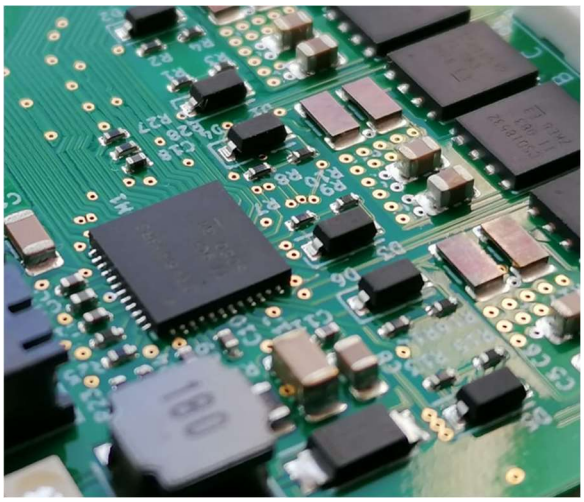


*Figure 16 Photo du driver DRV8353RS, de l'abaisseur de tension, du contrôle des grilles et des résistances de shunt.*

## Mode de contrôle

Le driver permet plusieurs modes de contrôle PWM (6x, 3x, 1x et indépendant). Il est possible de gérer chaque MOSFET avec sa propre PWM (6x et indépendant), ou chaque demi-pont (3x) ou encore de fournir un seul signal PWM (1x) et l'état des capteurs à effet Hall et laisser la table de vérité interne du driver s'occuper de la commutation. C'est ce dernier mode qui nous intéresse ici pour réaliser le fonctionnement en 6-step. Le mode 3x nous intéressera par la suite pour éventuellement implémenter un contrôle vectoriel qui permet une gestion plus précise du courant dans les phases.

Dans notre cas le moteur est contrôlé par un unique signal PWM envoyé au driver, accompagné de plusieurs signaux de commande comme l'état de capteurs à effet Hall, le signal de direction et celui de freinage. Les état des capteurs à effet Hall ne viennent pas directement du moteur mais sont recopiés par le microcontrôleur. Cela permet d'être flexible si on souhaite utiliser la carte dans un autre mode plus tard, mais pas uniquement. Des états impossibles à atteindre sont définie dans la table de vérité du driver et permettent des fonctions intéressantes comme la roue libre ou un mode d'alignement permettant de verrouiller le moteur face à la phase A.

Le signal de direction permet de dire au driver dans quel sens faire tourner le moteur et donc quel état de commutation lui appliquer. Le signal de freinage ferme tous les MOSFET bas et ouvre tous les MOSFET hauts, cela a pour effet d'appliquer une résistance à la rotation du moteur. En tournant, le moteur génère un courant dans ses phases, donc une tension apparait à leurs bornes mais comme elles sont toutes reliées entre elles, la tension ne peut pas varier librement, cela limite donc le courant et crée une contre réaction qui rend la rotation du moteur difficile.

**Table 3. Synchronous 1x PWM Mode**

| LOGIC AND HALL INPUTS | | | | | | | GATE-DRIVE OUTPUTS | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STATE | INHC = 0 | | | INHC = 1 | | | PHASE A | | PHASE B | | PHASE C | | DESCRIPTION |
| | INLA | INHB | INLB | INLA | INHB | INLB | GHA | GLA | GHB | GLB | GHC | GLC | |
| Stop | 0 | 0 | 0 | 0 | 0 | 0 | L | L | L | L | L | L | Stop |
| Align | 1 | 1 | 1 | 1 | 1 | 1 | PWM | !PWM | L | H | L | H | Align |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | L | L | PWM | !PWM | L | H | B → C |
| 2 | 1 | 0 | 0 | 0 | 1 | 1 | PWM | !PWM | L | L | L | H | A → C |
| 3 | 1 | 0 | 1 | 0 | 1 | 0 | PWM | !PWM | L | H | L | L | A → B |
| 4 | 0 | 0 | 1 | 1 | 1 | 0 | L | L | L | H | PWM | !PWM | C → B |
| 5 | 0 | 1 | 1 | 1 | 0 | 0 | L | H | L | L | PWM | !PWM | C → A |
| 6 | 0 | 1 | 0 | 1 | 0 | 1 | L | H | PWM | !PWM | L | L | B → A |

*Figure 17 Table de vérité du driver. Texas Instruments, DRV835x 100-V Three-Phase Smart Gate Driver datasheet (Rev. A).*

Le mode de fonctionnement doit être défini à chaque démarrage en utilisant la communication SPI. Cette configuration initiale inclue également tous les autres paramètres comme la durée des temps morts, la vitesse de commutation le courant de charge et décharge des grilles, etc.

## Driver bas-niveau

Pour simplifier la programmation et rendre le code plus lisible, la création d'un driver est très utile. L'objectif de ce driver n'est pas d'implémenter toutes les fonctionnalités du composant mais de créer des fonctions regroupant les processus les plus courants dans notre cas d'utilisation.

Fonctions publiques :

*HAL_StatusTypedef DRV_SetMode (Mode (= SLEEP, STOP, ALIGN, BRAKE, OPERATING))*

Permet de définir le mode de fonctionnement du DRV8353. Au changement de mode la valeur du rapport cyclique est remise à 0. En cas de sortie du mode veille (SLEEP) la fonction drv_wake est appelée pour réveiller le composant et envoyer la configuration via SPI.

*HAL_StatusTypedef DRV_SetPWM ( -100 < dc < 100)*

Permet de définir le rapport cyclique de la PWM envoyée au driver ainsi que l'état du bit du direction. L'entrée doit être comprise entre [-100 et 100] et correspond au rapport cyclique signé. La valeur d'entré est testée et saturée.

*HAL_StatusTypedef DRV_UpdateHalls ()*

Permet de mettre à jour la sortie les sorties Hall en fonction du mode actuel du driver et des entrées Hall. Les sorties sont égales aux entrées uniquement en mode OPERATING. En mode ALIGN elles sont forcées à l'état haut et dans tous les autres modes elles sont forcées à l'état bas.

Fonctions privées :

*HAL_StatusTypedef drv_wake ()*

Permet de sortir le DRV8353 du mode veille. Au réveil tous les registres sont effacés, la configuration est donc renvoyée via SPI. La valeur de la PWM est remise à 0 et le mode stop (roue libre) est appliqué.

*HAL_StatusTypedef drv_sleep ()*

Permet de mettre le DRV8353 en veille en abaissant le pin ENABLE à l'état bas dans cet état tous les MOSFET sont éteints, le moteur est donc en roue libre. Tous les registres interne sont effacés, il faudra renvoyer la configuration via SPI après le réveille (déjà inclus dans la fonction drv_Wake).

| Name | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Type | Address |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **DRV8350S and DRV8350RS** | | | | | | | | | | | | | |
| Fault Status 1 | FAULT | VDS_OCP | GDF | UVLO | OTSD | VDS_HA | VDS_LA | VDS_HB | VDS_LB | VDS_HC | VDS_LC | R | 0h |
| VGS Status 2 | SA_OC | SB_OC | SC_OC | OTW | GDUV | VGS_HA | VGS_LA | VGS_HB | VGS_LB | VGS_HC | VGS_LC | R | 1h |
| Driver Control | OCP_ACT | DIS_GDUV | DIS_GDF | OTW_REP | PWM_MODE | | 1PWM_COM | 1PWM_DIR | COAST | BRAKE | CLR_FLT | RW | 2h |
| Gate Drive HS | LOCK | | | IDRIVEP_HS | | | | IDRIVEN_HS | | | | RW | 3h |
| Gate Drive LS | CBC | TDRIVE | | IDRIVEP_LS | | | | IDRIVEN_LS | | | | RW | 4h |
| OCP Control | TRETRY | DEAD_TIME | | OCP_MODE | | OCP_DEG | | VDS_LVL | | | | RW | 5h |
| Reserved | Reserved | | | | | | | | | | | RW | 6h |
| Reserved | Reserved | | | | | | | | | | | RW | 7h |
| **DRV8353S and DRV8353RS** | | | | | | | | | | | | | |
| Fault Status 1 | FAULT | VDS_OCP | GDF | UVLO | OTSD | VDS_HA | VDS_LA | VDS_HB | VDS_LB | VDS_HC | VDS_LC | R | 0h |
| VGS Status 2 | SA_OC | SB_OC | SC_OC | OTW | GDUV | VGS_HA | VGS_LA | VGS_HB | VGS_LB | VGS_HC | VGS_LC | R | 1h |
| Driver Control | OCP_ACT | DIS_GDUV | DIS_GDF | OTW_REP | PWM_MODE | | 1PWM_COM | 1PWM_DIR | COAST | BRAKE | CLR_FLT | RW | 2h |
| Gate Drive HS | LOCK | | | IDRIVEP_HS | | | | IDRIVEN_HS | | | | RW | 3h |
| Gate Drive LS | CBC | TDRIVE | | IDRIVEP_LS | | | | IDRIVEN_LS | | | | RW | 4h |
| OCP Control | TRETRY | DEAD_TIME | | OCP_MODE | | OCP_DEG | | VDS_LVL | | | | RW | 5h |
| CSA Control | CSA_FET | VREF_DIV | LS_REF | CSA_GAIN | | DIS_SEN | CSA_CAL_A | CSA_CAL_B | CSA_CAL_C | SEN_LVL | | RW | 6h |
| Reserved | Reserved | | | | | | | | | | CAL_MODE | RW | 7h |

*Figure 18 Registre SPI pour la configuration du DRV8353RS. Texas Instruments, DRV835x 100-V Three-Phase Smart Gate Driver datasheet (Rev. A).*

## 2.6 Gestion de la lecture des capteurs

### Mesure par ADC

Les cartes Moteurs peuvent être connectées aux capteurs de forces des poignées du robot. Ces capteurs doivent être alimentés en 3.3V et revoient une tension analogique image de la force qui est appliquée selon leurs axes. Il y a un capteur pour l'axe Z, force verticale, et pour l'axe X, vers l'avant/arrière du robot. La tension du signal est comprise entre 0 et 3.3V ce qui correspond bien à la plage de mesure de l'ADC du microcontrôleur. Un moyenneur numérique permet de filtrer le signal.

*Figure 19 Poignée du robot Walk-E intégrant les capteurs de force*

L'autre grandeurs mesuré par ADC est le courant des phases. Il faut appliquer un filtre médian à ces valeurs pour supprimer les éventuelles parasite et les pointes lors des commutations des MOSFET. Dans les deux cas, la conversion de l'ADC s'effectue en continue, une entrée après l'autre et les résultats sont stockés en mémoire grâce à une DMA.

La DMA est configurée pour enregistrer les valeurs dans un buffer circulaire de la taille de l'ordre des filtres +1. A chaque conversion le résultat arrive en mémoire dans le buffer à la place de la plus ancienne valeurs, l'indice s'incrémente pour la conversion suivante. Si l'indice est à la fin de l'espace mémoire il retourne au début, c'est pour ça qu'il est dit circulaire.

Lors de l'exécution de la tâche de mesure de force la moyenne de chaque axe est mise à jour en soustrayant la plus ancienne mesure (N-ordre du filtre -1) à la nouvelle mesure (N) divisé par l'ordre du filtre+1 (nombre de valeur utilisé dans la moyenne). Le résultat de cette opération est ajouté à la valeur de la moyenne précédente pour obtenir la nouvelle moyenne. Cela permet de mettre à jour la moyenne avec seulement deux opération mathématique.

Pour le calcul de le médiane, les mesures sont stockées dans un tableau par la DMA et trié par ordre croissant lors de la mise à jour de la valeur du courant. La médiane est la valeur centrale de ce tableau. Cela prend beaucoup de temps si on a beaucoup de valeur.

## Mesure par interruption extérieure

Pour certaines mesures on ne cherche pas à connaitre une valeur mais l'instant d'un changement d'état. Pour cela on utilise les interruptions extérieures. Il s'agit d'une interruption déclenchée sur un front montant descendant ou les deux d'un signal numérique d'entrée, cela permet de déclencher un processus rapidement lors d'un changement de signal.

Ce type de mesure est notamment utilisée pour les capteurs à effets halles. A chaque changement d'état d'un des trois capteurs, les sorties correspondantes sont mises à jour pour que le driver DRV8353 puisse faire avancer sa table de vérité sans latence. Un temps de propagation trop long pourrait nuire à la fluidité du mouvement du moteur.

Cette interruption est également utilisée pour les capteurs de fin de course des chariots poignée afin qu'il n'entre pas en buter à la fin de leur mouvement. On utilise également ce système pour déclencher des processus de sécurité lorsque le driver nous avertit d'un problème via son pin d'erreur.

## Mesure par timer

La mesure par timer consiste à incrémenter un compteur à chaque front d'un signal d'entré. Ici, ce type de mesure est utilisé pour mesurer la position du moteur grâce aux encodeur incrémentaux présent dans les moteurs roues. Un encodeur envoie 2 signaux créneaux déphasés entre eux. Chaque front correspond à un déplacement d'un pas, la phase entre les fronts indique la direction du déplacement.

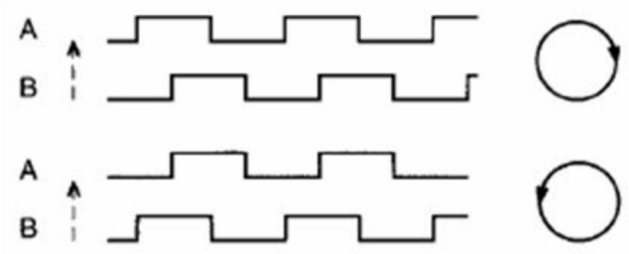



*Figure 20 Signaux de sortie d'un encodeur. Mytopschool.net, Les capteurs numériques.*

Dans notre microcontrôleur ST, un des timer est capable de gérer ce type de capteurs de façon matériel. Le timer reçoit les deux signaux et, à chaque front, détecte la phase entre les signaux et incrémente ou décrémente le compteur en fonction du sens de rotation. Cela permet de gagner énormément de temps processeur car sinon il faudrait déclencher plus de 3000 interruptions par tour de roue. Même si l'interruption était très courte cela prendrait la totalité du temps processeur.

Pour connaitre la position il suffit maintenant de lire la valeur du compteur quand nous en avons besoin. On peut également connaitre le sens de rotation en comparant la valeur de ce registre à celle lue précédemment.

## 2.7 Bus CAN

Le bus CAN est un bus de communication très utilisé dans l'automobile pour sa robustesse. Sur les cartes précédente la communication entre les cartes Moteurs et la carte principale était réalisée à l'aide d'une liaison série multiplexée. Cela posait plusieurs problèmes, notamment la vitesse de transmission et l'impossibilité pour les cartes Moteurs d'envoyer des données continuellement.

Le bus CAN fonctionne à l'aide deux lignes permettant de coder des états hauts et bas, il s'agit d'une boucle de courant. Le courant circulant dans une ligne circule aussi en sens inverse dans l'autre ligne. Ces deux lignes sont torsadées et leurs courant s'annulant permet d'avoir une impédance de ligne très faible car leurs inductions se compensent. Des résistances de terminaison doivent être ajouté aux extrémités du bus pour assurer la bonne continuité du courant.

Le microcontrôleur choisis intègre un contrôleur CAN qui permet d'envoyer des trames au bon format sur un Tx et d'en recevoir sur un Rx. Ce contrôleur ne peut pas être connecté directement au bus CAN mais à un transceiver CAN qui convertira le trame série reçue en niveau de tension CAN et inversement sur la liaison Rx.

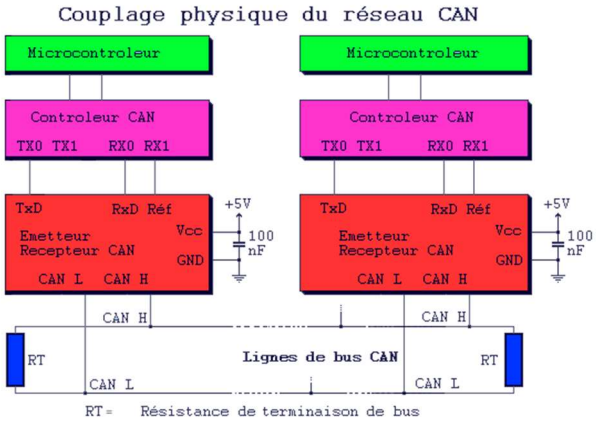



*Figure 21 Schéma de deux nœuds sur un bus CAN. lelectronique.com, DOSSIER D'ÉLECTRONIQUE, LE BUS CAN.*

Nous avons eu besoin de différencier chaque carte du robot entre-elles afin de leurs attribuer un rôle et donc une adresse CAN. Pour cela un interrupteur a été ajouté, permettant de changer l'état d'un pin du microcontrôleur qui peut ainsi connaitre le rôle de la carte dans le robot et s'attribuer la bonne adresse CAN. Un deuxième interrupteur permet également de connecter ou non la résistance de terminaison.

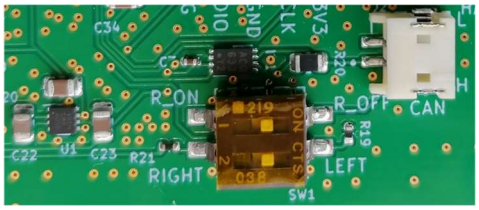



*Figure 22 Photo du connecteur pour le bus CAN et des interrupteurs.*

# 3 Fabrication

## 3.1 Choix des composants

Une des raisons pour lesquelles nous devons développer ces cartes est que les composants nécessaires à la fabrication des cartes précédents ne sont plus disponibles (plus de stock ou obsolète). Le manque de stock de composant liée à la COVID-19 est une énorme contrainte. Cela nous oblige à stocker des composants avant de valider le design de la carte pour ne pas avoir à redévelopper une partie du circuit avec un autre composant.

L'élément le plus critique était le driver DRV8323 initialement que nous avons remplacé par un DRV8353 par manque de stock. La différence étant la tension d'alimentation maximale plus élevé pour le 53. La seule référence que nous avons pu conserver a été celle des MOSFET. Le microcontrôleur de la famille STM32F0 a été choisis en fonction des stocks disponibles.

Une fois la liste des composants principaux établis, la réalisation des schémas sur Kicad a pu être entamé. Tout en commandant petit à petit les composants ajoutés à la BOM. Le schéma électrique a été réalisé sur Kicad, il s'agit d'un logiciel gratuit et open-source permettant de réaliser un schéma électrique ainsi que de router une carte.

*Figure 23 Logo du logiciel Kicad.*

La première étape consiste à importer ou à créer les composants manquant dans le logiciel. Puis à dessiner le schéma électrique. On utilise des feuilles hiérarchiques afin de distinguer les parties de la carte et gagner en lisibilité. Le schéma électrique est réparti en 5 feuilles :

- MOSFET pour l'onduleur et les résistances de shunt permettant la mesure du courant
- Driver pour le DRV8353 et son régulateur Buck 3.3V intégré
- Microcontroller pour le microcontrôleur et les connecteurs de signaux, de programmation et de liaison série
- CAN pour le transceiver CAN et les interrupteurs
- Power pour le connecteur d'alimentation, le LDO 1V8 pour le microcontrôleur et le régulateur boost 5V pour le transceiver CAN et les capteurs à effet Hall du moteur

Une fois le schéma complet et revu, on peut passer au routage de la carte.

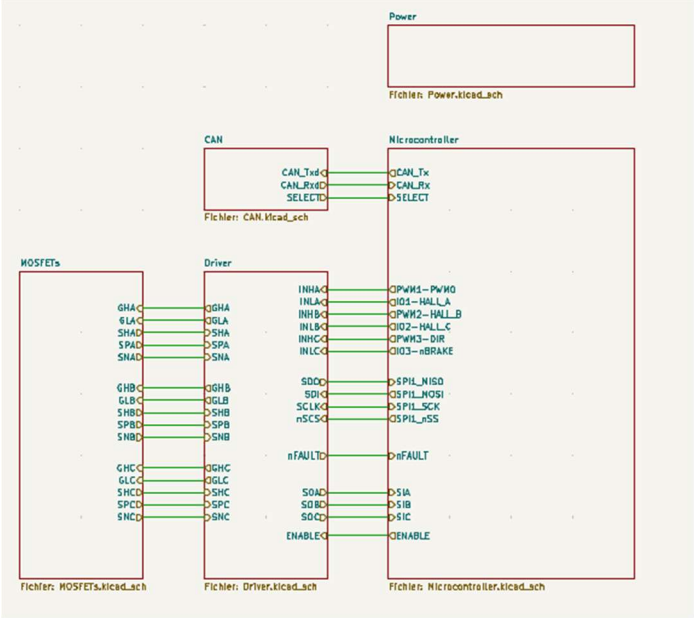



*Figure 24 Feuille principale contenant les 5 sous-feuilles.*

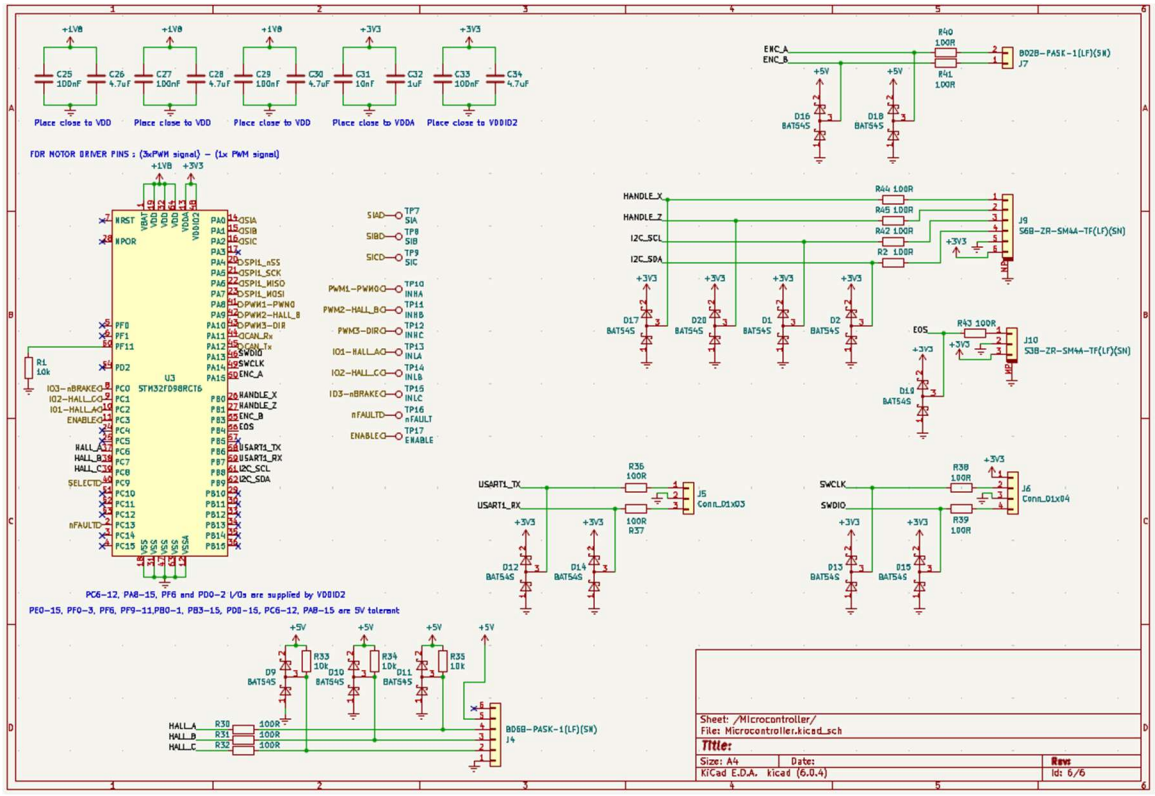



*Figure 25 Sous-feuille Microcontroller.*

## 3.2 Routage de la carte

La première étape est d'associer les empreintes à chaque composant. C'est à cette étape que les composants comme les résistances et les condensateurs sont choisis.

| | | | |
|---|---|---|---|
| C16 | Capacitor_SMD:C_1206_3216Metric | CAP 1206 100nF 100V X5R 10% | 1 |
| C22 | Capacitor_SMD:C_0805_2012Metric | CAP 0805 1uF 10V X5R 10% | 1 |
| > C20, C23 | Capacitor_SMD:C_0805_2012Metric | CAP 0805 10uF 10V X5R 10% | 2 |
| C24 | Capacitor_SMD:CP_Elec_10x10.5 | CAP ALU CMS 220µF 20% 80V | 1 |
| > C26, C28, C30, C34 | Capacitor_SMD:C_0805_2012Metric | CAP 0805 4.7uF 10V X5R 10% | 4 |
| > D1, D2, D9-D20 | BAT54S:D_SOT-23_BAT54S | DIODE SCHOTTKY 30V 200MA TO236AB | 14 |
| D7 | MBR120AFC_R1_00001:SODFL5226X110N | DIODE SCHOTTKY 120V 1A CMS | 1 |
| D8 | LED_SMD:LED_PLCC-2 | LED RED CLEAR 2PLCC SMD | 1 |
| IC1 | Package_DFN_QFN:TDFN-8-1EP_3x2mm_P0.5mm_EP1.3x1.4mm | CI INTERFACE CAN TDFN-8 | 1 |
| J1 | Connector_JST:JST_VH_B3P-VH-B_1x03_P3.96mm_Vertical | CON EMBASE CI VH 3.96MM 3PINS | 1 |
| J2 | JST_SXB_ZR-SM4A-TF:JST_S2B-ZR-SM4A-TF(LF)(SN) | CON HEADER ZH CMS 1.5MM 2PINS | 1 |
| J3 | Connector_Molex:Molex_Micro-Fit_3.0_43650-0200_1x02_P3.00mm_Horizont | CON HEADER ANGL/DR 3MM 2PINS | 1 |
| J4 | B06B-PASK-1__LF__SN_:B06BPASK1LFSN | CON HEADER MALE PTH 2MM 6PINS | 1 |

*Figure 26 Table associant les éléments du schéma à leurs empreintes.*

Le routage peut maintenant commencer en délimitant la taille de la carte sur la couche de découpage. On choisit ensuite le placement des composants de façon à réduire au maximum les croisements entre les chevelus. Et on passe ensuite au dessin des pistes et des plans. La carte est une double couche, la face du dessous doit donc être un plan de masse le moins coupé possible. En suivant les recommandations du DRV8353 on peut avoir une bonne idée de routage.

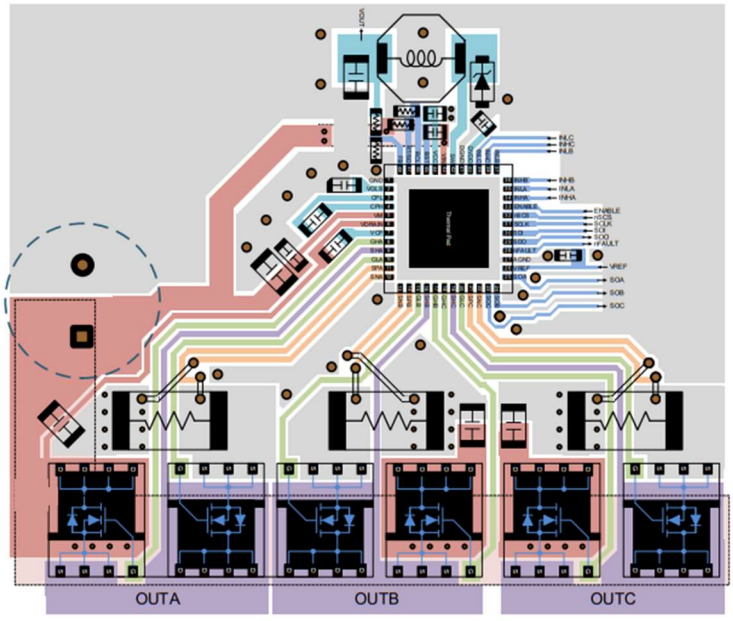



*Figure 27 Exemple de disposition du DRV8353 et de ses composants associés. Texas Instruments, DRV835x 100-V Three-Phase Smart Gate Driver datasheet (Rev. A).*

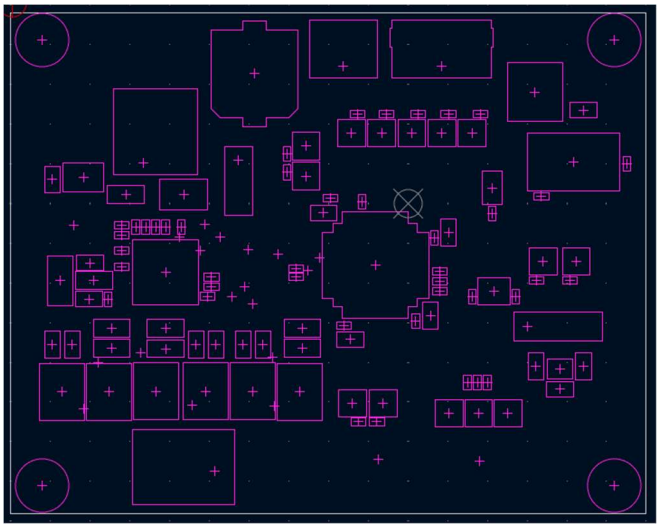


*Figure 28 Placement des composants.*

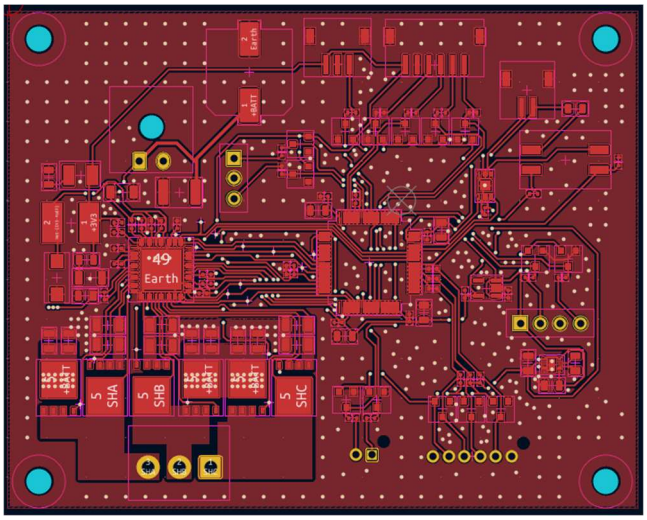



*Figure 29 Routage de la face supérieure.*

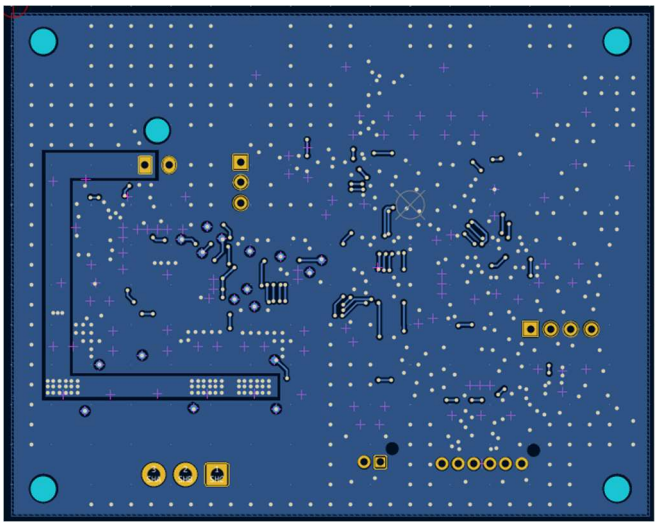


*Figure 30 Plan de masse inferieur. Sur la gauche le plan d'alimentation pour les MOSFET.*

Grâce à la visualisation 3D on peut avoir un bon aperçu du résultat final.

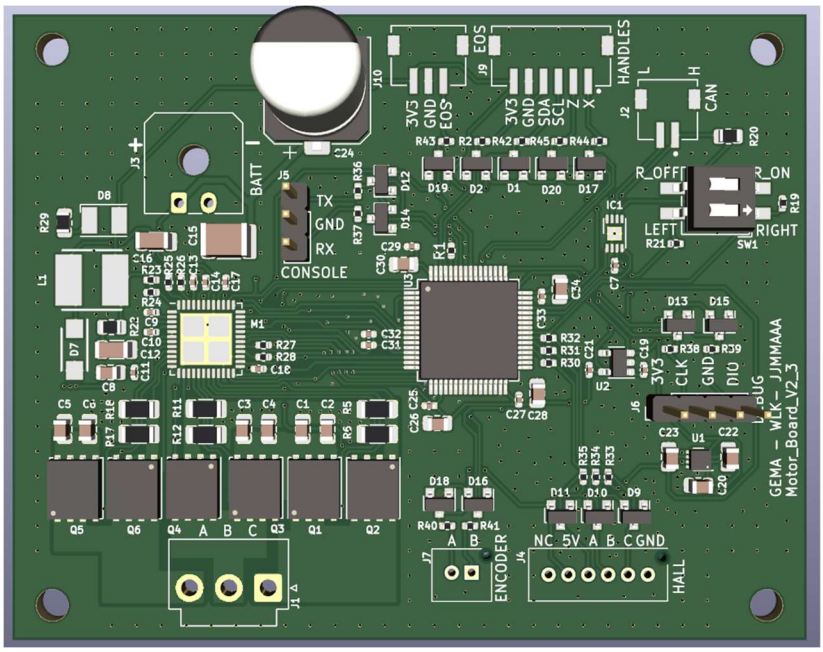



*Figure 31 Visualisation 3D de Kicad*

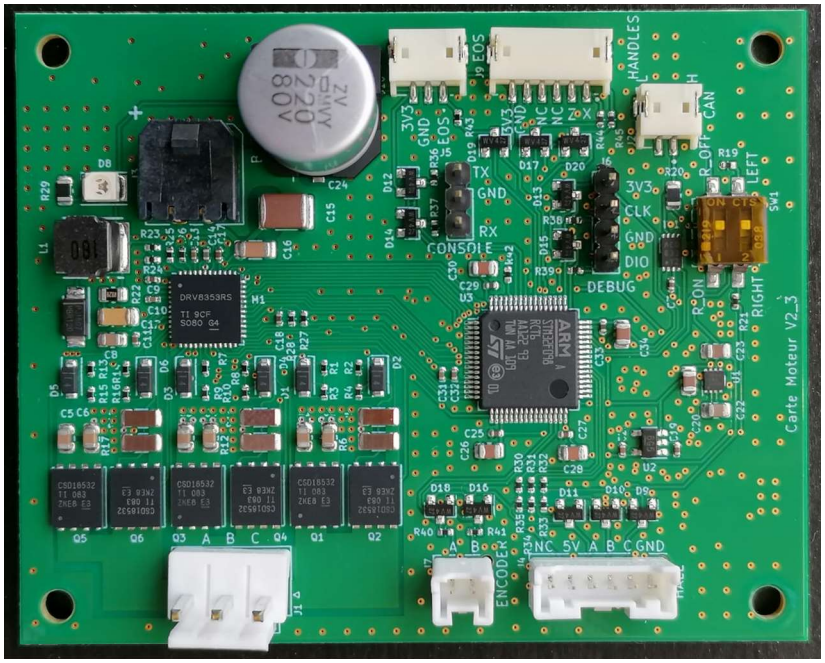



*Figure 32 Photo de la carte produite.*

# 4 Tests et validations

## 4.1 Validation hardware

A la réception des cartes assemblées, il est important de vérifier que le sens des composants sur une première carte pour vérifier qu'il n'y a pas eu une erreur à la création du PCB. Ensuite la polarité des connecteurs peut être vérifier notamment pour celui de l'alimentation. Une fois cela vérifié nous appliquons une tension d'alimentation minimal pour éviter limiter les dégâts en cas de problème. Avec un oscilloscope on tests les différents niveaux de tensions d'alimentations et leurs variations.

Les alimentations étant opérationnelles nous pouvons passer au test du microcontrôleur. Malheureusement il était impossible de le flasher à cause d'une erreur de câblage, le pin de reset du microcontrôleur était connecté au niveau de tension 3.3V pour le maintenir à l'état haut. Il eut fallu le relié au 1.8V ou de le laisser flottant car il a une pull-up interne.

Cela a conduit à devoir couper certaines pistes dont une se trouvant sous le microcontrôleur. Une opération minutieuse et risquée a donc été nécessaire pour rendre la carte fonctionnelle. Une fois l'opération réussis, le microcontrôleur a pu être flashé correctement et l'opération répété sur les autres cartes.

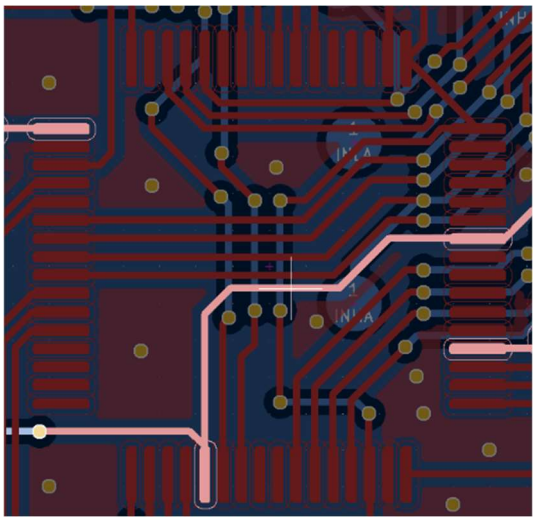


*Figure 33 Piste à couper sous le microcontrôleur.*

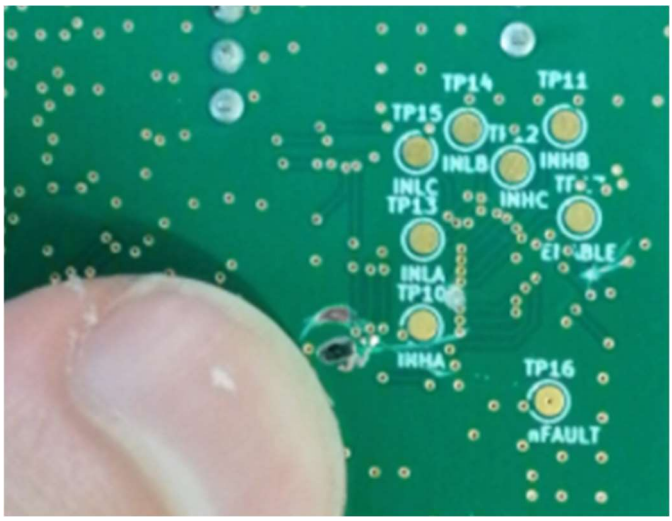



*Figure 34 Perçage réalisé pour couper la piste de la face avant.*

Un autre problème qui a nécessité une correction hardware était le transceiver CAN qui avait était soudé à l'envers. Il a fallu le dessouder à l'air chaud sans l'endommager puis le ressouder à la main.

## 4.2 Test du microcontrôleur

Tester le microcontrôleur est important car cela permet de vérifier qu'il fonctionne normalement et mettra en lumière toute erreur de branchement, de tension d'alimentation, etc. La première chose à vérifier est la fréquence de l'horloge et le fonctionnement des GPIO.

On test ensuite les bus de communications CAN et SPI. Le bus CAN peut être testé à l'aide d'un adaptateur CAN vers USB USART. On peut ainsi recevoir et envoyer des trames CAN depuis le port série d'un PC. Lors de cette vérification nous avons constaté que le transceiver était monté à l'envers, nous l'avons donc ressoudé.

La communication SPI a pu être validé en essayant de communiquer avec le driver de MOSFET, en modifiant un de ses registres puis en le relisant. Nous avons donc pu valider le bus SPI coté microcontrôleur mais également du côté du driver, en plus de valider une partie du fonctionnement du driver.

Nous avons ensuite pu configurer et tester les interruptions. D'abord les interruptions extérieures en vérifiant l'entré dans les fonctions appropriées lors de changements d'état sur certains pins. Ensuite les interruptions par timer en faisant changer l'état d'un pin à une fréquence donné et en affichant l'état du pin sur un oscilloscope. Dans le même temps nous avons pu tester la génération de signaux PWM.

Grace à l'interruption timer nous avons pu mettre en place le séquenceur et vérifier sa bonne exécution. Le comptage et décomptage en fonction de l'état des encodeurs incrémentaux a également été testé en lisant la valeur d'un timer tout en faisant tourner une roue. Une fois tout le système bas niveau mis en place, nous avons pu passer à la programmation du logiciel embarqué.

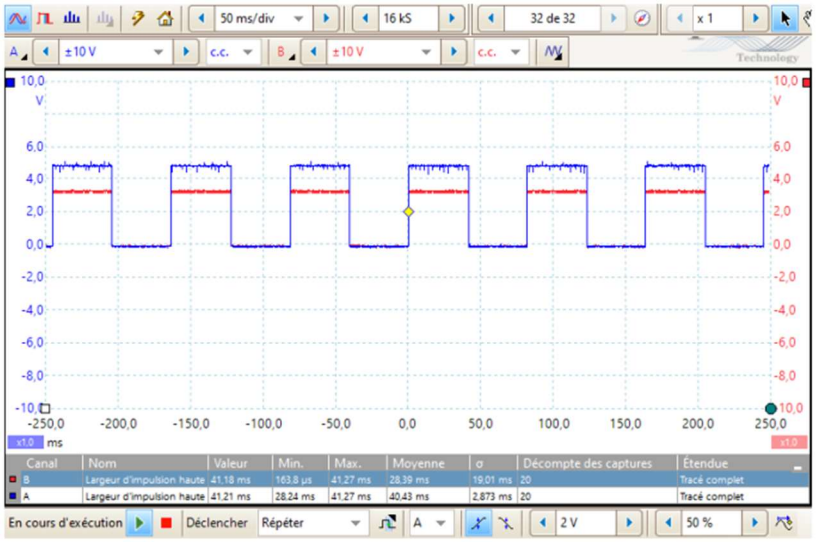



*Figure 35 Signal d'un capteur à effet Hall (en bleu) et sa recopie par le microcontrôleur (en rouge).*

## 4.3 Tests du driver

Une fois la configuration par SPI du driver validée, il est possible de mettre en marche un moteur en envoyant un signal PWM au driver, en commençant avec un rapport cyclique faible pour limiter les risques en cas de problème.

En simulant l'état des capteurs à effets Hall avec le microcontrôleur, on vérifie l'état des MOSFET et la correspondance avec la table de vérité données dans la documentation. Grâce à un oscilloscope on peut mesurer les temps de monté des grilles ainsi que les temps morts et vérifier l'absence de recouvrement entre les MOSFET d'une même branche. Nous avons ainsi pu ajuster la configuration du driver en fonction des valeurs mesurées. Une fois tout cela validé nous avons pu connecter un moteur.

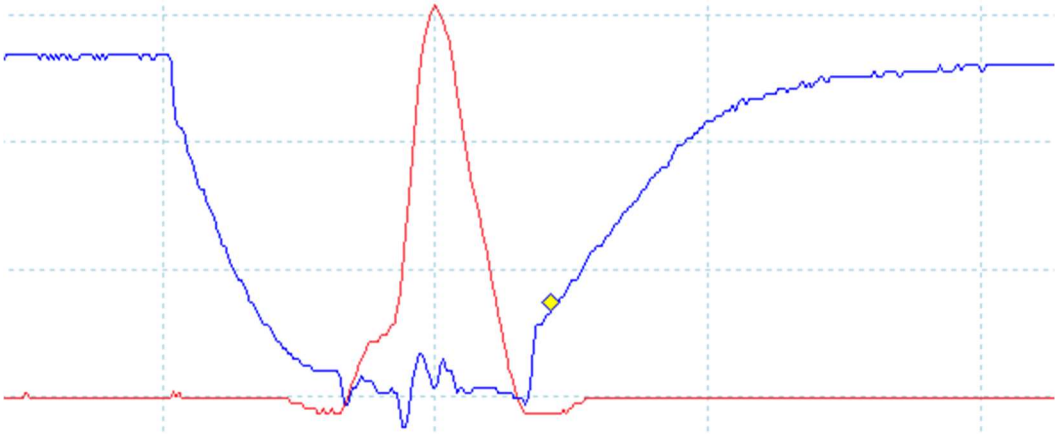


*Figure 36 Tension des grilles des MOSFET haut (en rouge) et bas (en bleu) pour un rapport cyclique de 4%.*

## 4.4 Tests unitaires

### Fonctionnement en boucle ouverte

Un élément important pour la fluidité de la rotation du moteur est la correspondance entre les capteurs à effet Hall et les phase du moteur. Cette correspondance n'étant pas indiqué dans la documentation des moteurs utilisé, nous avons dû le faire expérimentalement. Pour cela il s'agit de connecter le moteur et d'appliquer une PWM faible. Si le moteur vibre, ne tourne pas régulièrement où se boque les phases ne sont pas dans le bon ordre.

Une fois la bonne combinaison trouvée il est facile d'intervertir les signaux dans le programme du microcontrôleur, c'est une des avantages d'avoir fait arriver ces signaux sur le microcontrôleur plutôt que directement sur le driver. Le moteur tourne à une vitesse proportionnelle au rapport cyclique du signal PWM et change de sens en fonction du signal de direction.

## Mesure de position et de vitesse

On cherche à valider la position et la vitesse mesurée. Pour cela on observe la valeur indiquée par le compteur après avoir réalisé une dizaine de tour et on compare la valeur théorique à la valeur mesurée.

Chaque tour correspond à 3200 pas, soit une précision de 0.1125°. La valeur du compteur étant sur 32bits non signée, la valeur maximale est de 4 294 967 295. Il est donc possible de faire 1 342 177 tour complet avant de reboucler à 0. Etant donné que le diamètre de la roue est d'environ 20 cm cela correspond à 843 km ce qui est amplement suffisant pour une balade avec Walk-E.

La mesure de vitesse consiste à lire la valeur d'un timer à chaque changement de capteur à effet Hall. En connaissant la distance d'un pas de capteur à effet Hall et le mis pour parcourir ce pas on obtient une vitesse.

## Commande par CAN

A l'aide d'une console codée en python nous pouvons envoyer des trames en série à un convertisseur CAN qui peut communiquer avec la carte. L'objectif est de vérifier que la trame reçue est correctement décodée par le microcontrôleur et que la donnée placée en mémoire soit encodée correctement.

Pour cela on envoi des trames de commande et de changement de mode de contrôle. Et on observe l'état de la mémoire grâce au débugueur. La communication fonctionne correctement pour toutes les trames définies

CAN frame format for communication in Walk-E system

| Type de message | Header (+ n° nœud) | DLC | PAYLOAD ( BYTE 0 to 7 ( Hex format ) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| | | | Index | | Subindex | | Data | | | |
| TPDO1 | 0x180 | 8 | Speed | | Torque | | Position | | | |
| TPDO2 | 0x280 | 8 | Fx | | Fz | | Ctrl mode | Temp° | Error flags | Status flags |
| SDO_Req_Version | 0x600 | 8 | 1001 | | 0 | | 0 | | | |
| SDO_Rep_Version | 0x610 | 8 | 1001 | | Major | | Minor | | Rev | |
| SDO_Set_Motor_On | 0x600 | 8 | 2300 | | 1100 | | 1 | | | |
| SDO_Set_Motor_Off | | | | | 1200 | | 0 | | | |
| SDO_Set_Setpoint | 0x600 | 8 | 2000 | | 0 | | Setpoint | | | |
| SDO_Set_Ctrl_Type | 0x600 | 8 | 2100 | | 0 | | Control type | | | |
| SDO_Start_Homing | 0x600 | 8 | 3100 | | 0 | | 1 | | | |
| SDO_Set_coeff_kp_position | 0x600 | 8 | 4100 | | 0 | | kp | | | |
| SDO_Set_coeff_kp_speed | 0x600 | 8 | 4200 | | 0 | | kp | | | |
| SDO_Set_coeff_kp_torque | 0x600 | 8 | 4300 | | 0 | | kp | | | |
| SDO_Set_coeff_ki_position | 0x600 | 8 | 4010 | | 0 | | ki | | | |
| SDO_Set_coeff_ki_speed | 0x600 | 8 | 4020 | | 0 | | ki | | | |
| SDO_Set_coeff_ki_torque | 0x600 | 8 | 4030 | | 0 | | ki | | | |
| SDO_Set_coeff_kd_position | 0x600 | 8 | 4001 | | 0 | | kd | | | |
| SDO_Set_coeff_kd_speed | 0x600 | 8 | 4002 | | 0 | | kd | | | |
| SDO_Set_coeff_kd_torque | 0x600 | 8 | 4003 | | 0 | | kd | | | |

*Figure 37 Format des trames CAN.*

## 4.5 Tests fonctionnels

### Séquenceur et taches

Cette étape consiste à valider le fonctionnement du séquenceur et le cadencement des tâches. Si les tâches durent trop longtemps, les autres tâches prendrons du retard et si ce retard s'accumule cela peut entraîner des conséquences inattendues lors de l'utilisation.

La méthode de vérification est d'utiliser les sorties non utilisées et de les faire basculer au début et à la fin de chaque tâche pour voir le temps pris par chacune d'entre elles et le temps libre restant. Après quelques corrections, le séquenceur fonctionne correctement et les tâches ne dépassent plus leurs temps attribués

### Asservissement vitesse et position

L'asservissement en position consiste à prendre la commande reçue par CAN et soustraire la position mesurée cela nous donne l'erreur. Cette erreur passe dans un régulateur PID et en ressort une commande de vitesse. La commande de vitesse est soustraite à la vitesse mesurée et passe également dans un PID qui donne la tension à appliquer. Cette tension est ensuite utilisée par l'asservissement en courant pour régler le rapport cyclique.

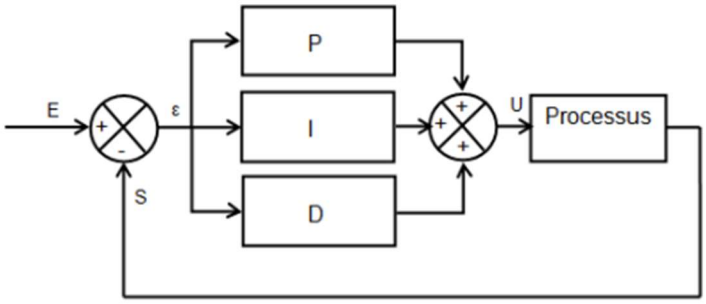



*Figure 38 Structure d'un régulateur PID. ENSTA Bretagne, Estimation et Contrôle-Commande du robot*

L'étape de test a également consisté à régler les différents coefficients des PID implémentés en commençant par la boucle de plus bas niveau puis en remontant : boucle de courant, boucle de vitesse et enfin boucle de position. Cette étape a été chronophage car il y a beaucoup de coefficient à régler.