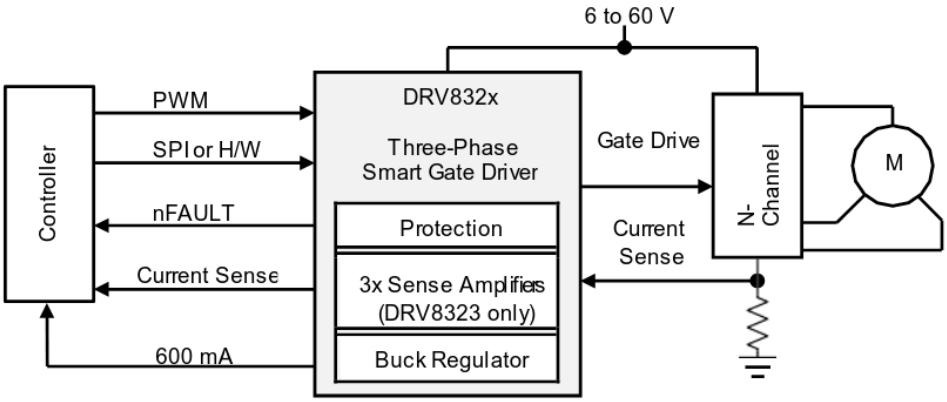
Notes de conception contrôleur de moteur BLDC

Ce document est une compilation des différentes notes prises par les membres du groupe lors de la conception. L'objectif n'est pas d'être un rapport expliquant le projet mais de mettre en lumière les réflexions et attentions prises afin d'aider les prochains à comprendre les choix techniques réalisés.

Table des matières

Driver de MOSFET

DRV832x Driver de pont triphasé pour MOSFET



Description

- La puce dispose de 3 drivers de grille en demi-pont, chacun capable de piloter des MOSFET de puissance à canal N côté haut et côté bas.

- Le DRV832x génère les tensions de commande de grille adéquates à l'aide d'une pompe de charge intégrée MOSFET haut et d'un régulateur linéaire pour les MOSFET bas.

- L'architecture prend en charge des courants de commande de grille de pointe allant jusqu'à 1A source et 2A.

- Le DRV832x peut fonctionner à partir d'une seule alimentation sur une plage d'alimentation de 6 à 60 V pour les MOSFET et de 4 à 60 V pour le régulateur buck optionnel.

- Configurables des drivers via l'interface SPI ou par interface hardware.
(SPI : DRV832xx**S** HDW : DRV832xx**H**)

- Dispose de 3 amplificateurs pour mesures de courant sur chaque demi-pont côté bas. (disponible sur le DRV832**3)**

- Le DRV832x**R** intègre un régulateur DC Buck (600mA)

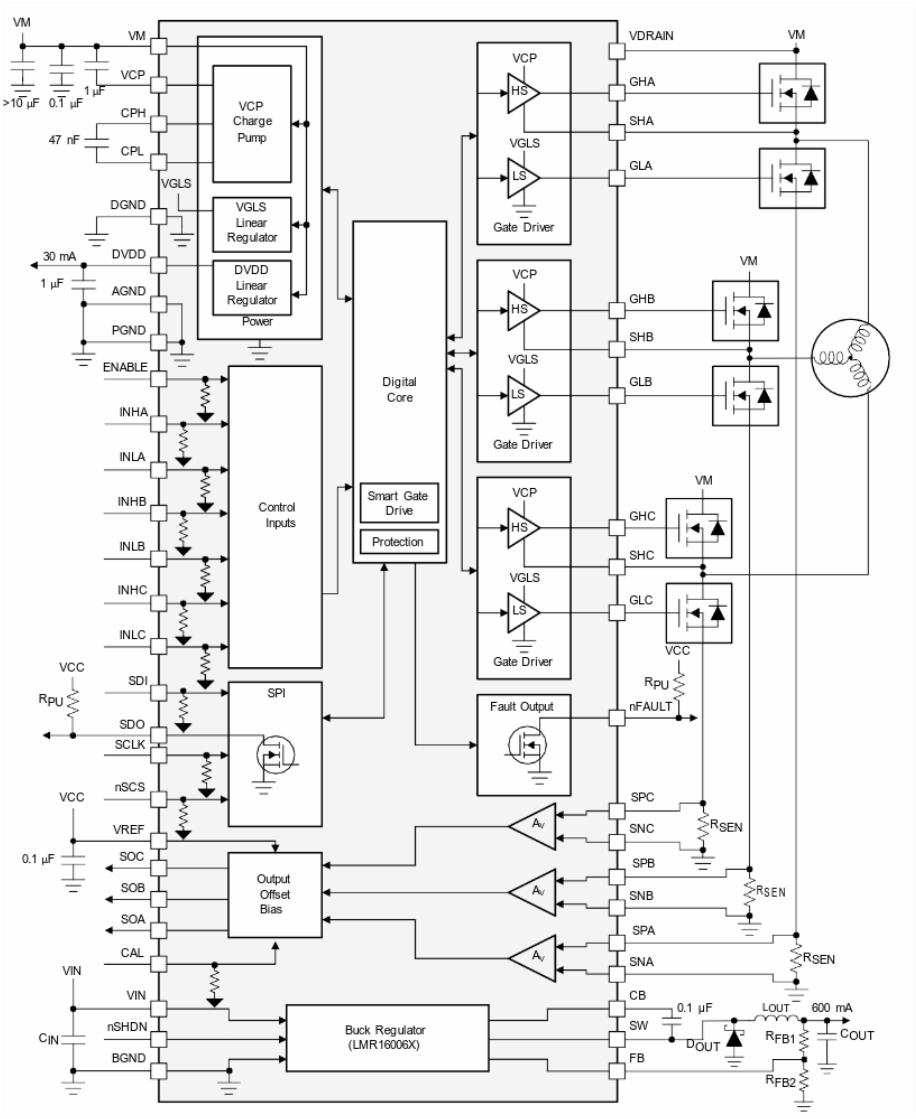
- Mode “veille” disponible

Choix du composant

Nous avons sélectionné le DRV8323RS pour les raisons suivantes :

- Ce composant met à disposition un régulateur buck qui nous est utile pour alimenter d’autre composant
- Interface de gestion SPI

Architecture

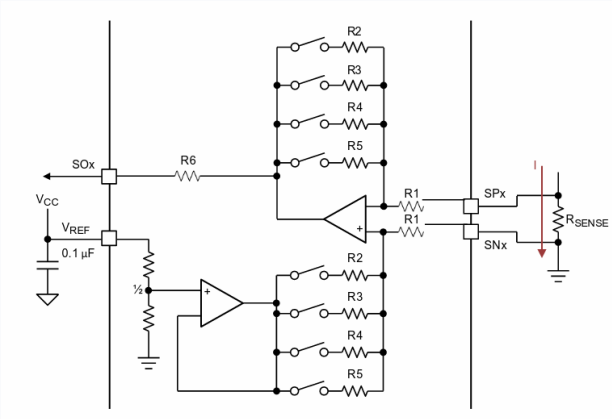


Mesure de courant

Les pins de sorties analogique SOx sont égales à la tension entre les pins SPx et SNx multiplié par le gain de l'amplificateur.

Le gain est ajustable selon 4 niveaux : 5V/V, 10V/V, 20V/V, et 40V/V

Formule : $$I = \frac{\frac{V_{VREF}}{2} - V_{SOx}}{G_{CSA} \times R_{SENSE}}$$



Amplitude :
3.3-0.25 -> 3.05
3.05 - 1.65 -> 1.4

1.4 = I x RSense x Gcsa. => Rsense = 1.4/I*Gcsa

Uni/Bi directionel	Gain	Imax	Rsense	P Rsense
1,4	40	20	0,00175	0,7
1,4	40	25	0,0014	0,875
1,4	40	30	0,001166667	1,05
1,4	20	20	0,0035	1,4
1,4	20	25	0,0028	1,75
1,4	20	30	0,002333333	2,1
2,8	40	20	0,0035	1,4
2,8	40	25	0,0028	1,75
2,8	40	30	0,002333333	2,1
2,8	20	20	0,007	2,8
2,8	20	25	0,0056	3,5
2,8	20	30	0,004666667	4,2

Choix BiDir : 1 mOhms : Choix1 (30Amax 1W max)
Choix UniDir : 2 mOhms : Choix 1 ou choix 2 (35A max, 2.45W max)

2,8	40	35	0,002	2,45

Mode de contrôle

Registre PWM_MODE

00b : 6x PWM Mode :
- 2 PWM pour chaque demi-pont : INLx et INHx
- État haute impédance: INLx et INHx

01b : 3x PWM Mode
- 1 PWM pour le contrôle de chaque demi-pont : INHx
- INLx est utilisé pour mettre le demi-pont en haute impédance.

10b : 1x PWM Mode
- 1PWM pour les 3 demi-pont
- Mosfet recirculation (synchronous 1PWM_COM=0) ou diode de roue libre des Mosfets uniquement (asynchronous 1PWM_COM=1).
- INLA = HALLA, INHB = HALLB, INLB = HALLC
- INHC = sens de rotation du BLDC

11b : Independent PWM Mode
- 2 PWM en entrées : INLx et INHx
- Pas d'état haute impédance

Mode retenu :

Synchronous 1x PWM

LOGIC AND HALL INPUTS							GATE DRIVE OUTPUTS(1)						
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

Configuration SPI

Control Registers

10	9	8	7	6 5	4	3	2	1	0
Reserved	DIS _CPUV	DIS _GDF	OTW _REP	PWM_MODE	1PWM _COM	1PWM _DIR	COAST	BRAKE	CLR _FLT
R/W-0b	R/W-0b	R/W-0b	R/W-0b	R/W-00b	R/W-0b	R/W-0b	R/W-0b	R/W-0b	R/W-0b

- PWM Mode -> 10b 1x PWM mode)
- 1PWM_COM -> 0b (uses synchronous rectification)

Gate drive HS register

10 9 8	7 6 5 4	3 2 1 0
LOCK	IDRIVEP_HS	IDRIVEN_HS
R/W-011b	R/W-1111b	R/W-1111b

- IDRIVEP_HS -> 1111b
- IDRIVEN_HS -> 1111b

Design du convertisseur Buck

Composant intégré

Le composant intégré au driver est un LMR6006X qui commute à 700kHz.

BUCK REGULATOR SUPPLY (VIN)						
I_{nSHDN}	Shutdown supply current	V_{nSHDN} = 0 V			1	3	µA
I_Q	Operating quiescent current	V_{VIN} = 12 V, no load; not switching			28		µA
V_{VIN_UVLO}	VIN undervoltage lockout threshold	VIN Rising				4	V
		VIN Falling		3			
BUCK REGULATOR SHUTDOWN (nSHDN)						
V_{nSHDN_TH}	Rising nSHDN threshold			1.05	1.25	1.38	V
I_{nSHDN}	Input current	V_{nSHDN} = 2.3 V			–4.2		µA
		V_{nSHDN} = 0.9 V			–1		
I_{nSHDN_HYS}	Hysteresis current				–3		µA
BUCK REGULATOR HIGH-SIDE MOSFET						
R_{DS_ON}	MOSFET on resistance	V_{VIN} = 12 V, V_{CB} to V_{SW} = 5.8 V, T_A = 25°C			900		mΩ
BUCK REGULATOR VOLTAGE REFERENCE (FB)						
V_{FB}	Feedback voltage			0.747	0.765	0.782	V
BUCK REGULATOR CURRENT LIMIT						
I_{LIMIT}	Peak current limit	V_{VIN} = 12 V, T_A = 25°C			1200		mA
						1700	
BUCK REGULATOR SWITCHING (SW)						
f_{SW}	Switching frequency			595	700	805	kHz
D_{MAX}	Maximum duty cycle				96%		
BUCK REGULATOR THERMAL SHUTDOWN						
T_{SHDN} (1)	Thermal shutdown threshold				170		°C
T_{HYS} (1)	Thermal shutdown hysteresis				10		°C

Choix de l'inductance

Point clé : Inductance
Courant pointe
Resistance DC

$$L_{o\,\min} = \frac{V_{in\,\max} - V_{out}}{I_o \times K_{IND}} \times \frac{V_{out}}{V_{in\,\max} \times f_{sw}} \quad (1)$$

$$I_{ripple} = \frac{V_{out} \times (V_{in\,\max} - V_{out})}{V_{in\,\max} \times L_o \times f_{sw}} \quad (2)$$

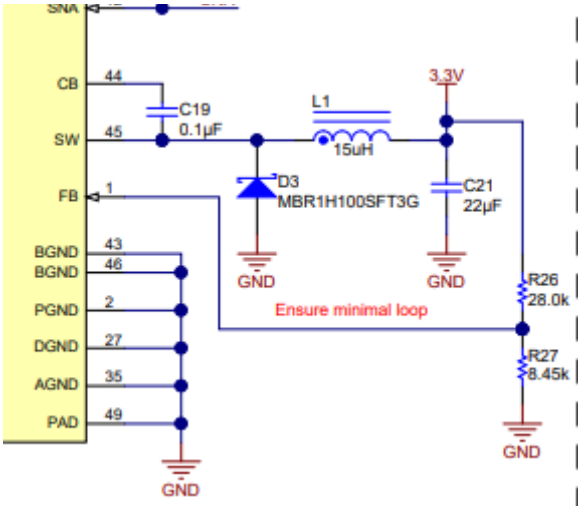
$$I_{L\text{-}RMS} = \sqrt{I_o^{\,2} + \frac{1}{12} I_{ripple}^{\,2}} \quad (3)$$

$$I_{L-peak} = I_o + \frac{I_{ripple}}{2} \quad (4)$$

Le courant d'ondulation (ripple current) augmente avec Vin. On effectue donc le calcul avec Vin_max.
On utilise l'équation 1 pour obtenir l'inductance minimale, avec KIND qui est le rapport du courant d'oscillation sur le courant de sortie maximal.
Une valeur raisonnable de L0 crée un rapport Iripple/Io entre 30 à 40 %.
Ici L0_min = 40uH
On choisit la valeur standard supérieur la plus proche : L0 = 47uH
On peut maintenant déterminer le courant efficace (RMS) et le courant de pointe avec l'équation 2.
On trouve I_RMS = 0.4 A et I_peak = 0.47 A.

Comparaison avec une carte d'évaluation

Sur le schéma d'une carte d'évaluation on trouve ces valeurs de composant :



La carte est donnée pour Vin allant de 8V à 54 V.

Comparaison avec la documentation du régulateur

Dans la documentation on trouve ces valeurs :

9.2 Typical Application

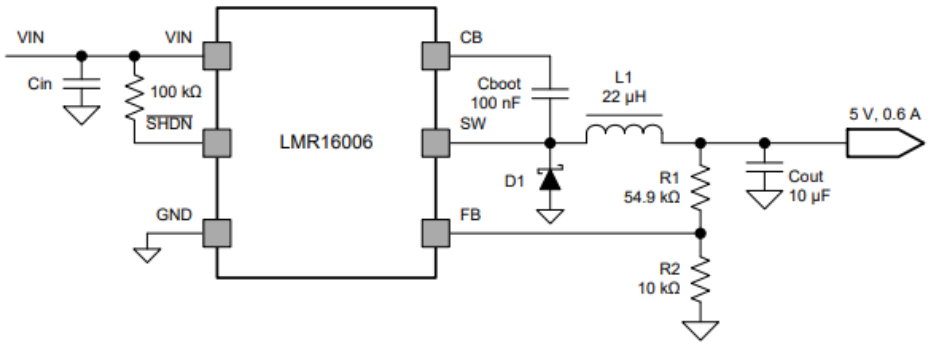


Figure 9. Application Circuit, 5 V Output

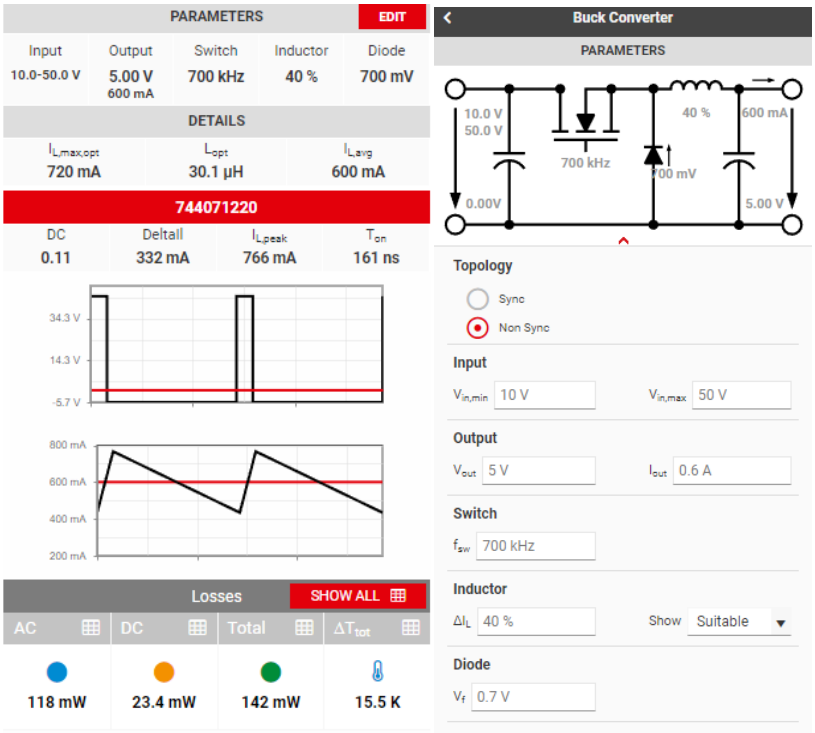
Le modèle est donné pour Vin allant de 9V à 16V.

Choix inductance

On choisit donc une inductance de 22uH environ et supportant un courant de 1.6A comme recommandé dans la documentation.
La capacité de sortie fera environ 10µF et devra supporter 5V (on prendra une marge).
La diode doit supporter une tension inverse de 45V et un courant de 1A.
On suivra également les recommandations de la doc du drv8323 :

BUCK REGULATOR			
C_{VIN}	VIN	BGND	X5R or X7R, 1 to 10 µF, VM-rated capacitor
C_{BOOT}	SW	CB	X5R or X7R, 0.1-µF, 16-V capacitor
D_{SW}	SW	BGND	Schottky diode
L_{SW}	SW	OUT(2)	Output inductor
C_{OUT}	OUT(2)	BGND	X5R or X7R, OUT rated capacitor
R_{FB1}	OUT(2)	FB	Resistor divider to set buck output voltage
R_{FB2}	FB	BGND	

Inductance choisis :
https://www.we-online.com/en/components/products/WE-PD2SR#744787220
744787220



Choix des autres composants

Cout = 10uF X5R or X7R, 16 V 0804
Cboot = 0.1uF X5R or X7R, 16 V
Cin = 10uF X5R or X7R, 100 V 2204

Diode Vrrm = 60V Vf = 0.7V If = 1A
https://www.mouser.fr/ProductDetail/Micro-Commercial-Components-MCC/SS16-LTP?qs=cLwp4wpikzDsiZm8qAd%2Fng%3D%3D

Bus CAN

Messages et filtres

On peut filtrer les messages reçus en fonction de leur identifiant grâce aux registres filtres et masques.

• Filtre :

– Seuls les bits non masqués sont testés

– Ils doivent correspondre au bits du filtre pour que le message soit accepté

• Masque :

– les bits de l'ID correspondants aux « 0 » du masque ne sont pas filtrés (ils sont acceptés)

Ensemble de registres pour stocker les trames :

- l'identifiant (3 registres 16 bits)

- la donnée (8 octets) = 4 registres 16bits

- les masques

- les filtres

2 « mailbox » de réception RX0 et RX1

3 « mailbox » d'émission TX0 TX1 et TX2

On peut transmettre 4 types de messages :

Les Data Frames sont utilisées pour transporter des données sur le bus.

Les Remote Frames sont utilisées par un nœud pour demander la transmission d'une Data Frame par d'autres nœuds avec le même ID. Deux éléments distinguent cette trame d'une trame de données normale : elle ne contient aucun bit de données et son bit RTR est récessif.

Les Error Frames sont transmises par un nœud ayant commis une erreur.

Les Overload Frames sont utilisés pour produire un délai entre deux Data ou Remote Frames successives.

Composants

MCP2515 : C'est un contrôleur CAN populaire de Microchip Technology qui peut être utilisé pour communiquer avec un moteur BLDC. Il est compatible avec les protocoles CAN 2.0A et 2.0B et dispose d'une interface SPI.

STM32F446RE : C'est un microcontrôleur de la famille STM32 de STMicroelectronics qui dispose d'un contrôleur CAN intégré. Il est compatible avec les protocoles CAN 2.0A et 2.0B et dispose d'une interface SPI.

SAM3X8E : C'est un microcontrôleur de la famille SAM3X d'Atmel qui dispose également d'un contrôleur CAN intégré. Il est compatible avec les protocoles CAN 2.0A et 2.0B et dispose d'une interface SPI.

TJA1050 : C'est un contrôleur CAN de NXP Semiconductors qui peut être utilisé pour communiquer avec un moteur BLDC. Il est compatible avec les protocoles CAN 2.0A et 2.0B et dispose d'une interface SPI.

http://wiki.labaixbidouille.com/index.php/Mise_en_place_d%27un_bus_CAN

https://controllerstech.com/can-protocol-in-stm32/

https://www.technologuepro.com/cours-systemes-embarques/cours-systemes-embarques-Bus-CAN.htm

MCP2551

Tout d'abord, il faut déterminer l'adresse que l'on souhaite attribuer à la carte. Chaque carte dans un réseau CAN doit avoir une adresse unique pour éviter les conflits. Nous voulons réaliser 4 cartes donc le bus d'adresse sur **2 bits sera suffisant.**

Pour communiquer avec une carte spécifique, on doit connaître son adresse unique. Cette adresse peut être configurée dans le code de votre microcontrôleur ou de votre processeur. Une fois que l'on connaît l'adresse de la carte avec laquelle on souhaite communiquer, on peut envoyer des messages CAN à cette adresse spécifique.

La bibliothèque de communication CAN pour le STM32G4 doit être incluse dans le code du projet. Si le logiciel STM32CubeIDE est utilisé, la bibliothèque CAN peut être ajoutée en sélectionnant "STM32CubeMX" dans la barre de menu, puis en cochant la case "CAN".

Pour définir l'adresse unique de la carte dans le code du projet, la fonction "HAL_CAN_ConfigFilter ()" peut être utilisée. Cette fonction configure les filtres d'acceptation de messages CAN. Un filtre peut être configuré pour chaque adresse unique à surveiller sur le réseau CAN.

Par exemple, si une adresse unique de 0x123 doit être configurée pour une carte, la fonction suivante peut être utilisée pour configurer le filtre :

```
CAN_FilterTypeDef filter ;

filter.FilterBank = 0;

filter.FilterMode = CAN_FILTERMODE_IDMASK;

filter.FilterScale = CAN_FILTERSCALE_32BIT;

filter.FilterIdHigh = 0x0000;

filter.FilterIdLow = 0x0123;

filter.FilterMaskIdHigh = 0x0000;

filter.FilterMaskIdLow = 0xFFFF;

filter.FilterFIFOAssignment = CAN_RX_FIFO0;

filter.FilterActivation = ENABLE;

HAL_CAN_ConfigFilter(&hcan, &filter);
```

Cette fonction configure un filtre pour accepter uniquement les messages avec l'identifiant 0x123. Les messages avec d'autres identifiants seront rejetés.

Enfin, les fonctions de la bibliothèque CAN pour le STM32G4, telles que "HAL_CAN_Transmit()" et "HAL_CAN_Receive()", peuvent être utilisées pour envoyer et recevoir des messages CAN.

En résumé, pour configurer l'adresse unique d'une carte pour un microprocesseur STM32G4, il faut définir le filtre d'acceptation de messages CAN pour surveiller l'adresse unique souhaitée en utilisant la fonction "HAL_CAN_ConfigFilter()". Les fonctions de la bibliothèque CAN pour le STM32G4 peuvent ensuite être utilisées pour envoyer et recevoir des messages CAN.

Routage

Les signaux de communication doivent être acheminés le plus directement possible entre le MCP2551 et le microcontrôleur pour minimiser les pertes de signal. Évitez les chemins de signal trop longs ou avec trop de virages, car cela peut affecter la qualité du signal.

Pour minimiser les interférences électromagnétiques, il est recommandé de placer le MCP2551 loin des sources de bruit électrique, comme les moteurs ou les alimentations à découpage. Si cela n'est pas possible, utilisez des filtres et des blindages pour protéger les signaux.

Pour éviter les courts-circuits, assurez-vous que les broches de l'IC ne sont pas connectées les unes aux autres ou à d'autres circuits. Lorsque vous dessinez les traces sur votre carte, gardez une distance suffisante entre les broches du MCP2551 et les autres composants pour éviter toute confusion ou erreurs de connexion.

Utilisez des pistes larges pour les signaux CAN et de masse pour minimiser les pertes de signal et les interférences électromagnétiques.

Assurez-vous que le MCP2551 est correctement alimenté en vérifiant la qualité de la tension d'alimentation. Si la tension d'alimentation est bruitée, utilisez des filtres pour minimiser le bruit.

Vérifiez que les résistances de terminaison de bus sont correctement dimensionnées. Les résistances de terminaison doivent avoir la valeur appropriée pour garantir une bonne communication sur le bus CAN. En général, une valeur de 120 ohms est utilisée pour chaque extrémité du bus.

Vérifiez que les broches de l'IC sont correctement polarisées. Le MCP2551 dispose d'une broche de polarité qui doit être correctement connectée pour garantir le bon fonctionnement de l'IC.

Type de Message :

Le message de données (Data Frame) : il s'agit du type de message le plus couramment utilisé sur le bus CAN. Il est utilisé pour transmettre des données entre différents nœuds du réseau CAN. La longueur du message de données varie de 0 à 8 octets.

Le message de commande à distance (Remote Frame) : il est utilisé pour demander des données spécifiques à un nœud du réseau. Contrairement au message de données, le message de commande à distance ne contient pas de données. Il est simplement utilisé pour demander une réponse de la part d'un nœud particulier. La longueur du message de commande à distance est toujours de 0 octet.

Le message d'erreur (Error Frame) : il est utilisé pour signaler une erreur sur le bus CAN. Il est transmis par un nœud pour signaler une erreur de communication ou un problème de transmission. La longueur du message d'erreur est toujours de 8 octets.

Commande de vitesse : cette commande permet de réguler la vitesse du moteur BLDC en envoyant une consigne de vitesse. La commande est généralement envoyée sous forme de message CAN avec l'identifiant approprié et la charge utile contenant la valeur de la consigne de vitesse.

Commande de position : cette commande permet de réguler la position du moteur BLDC en envoyant une consigne de position. La commande est généralement envoyée sous forme de message CAN avec l'identifiant approprié et la charge utile contenant la valeur de la consigne de position.

Commande de courant : cette commande permet de réguler le courant qui alimente les bobines du moteur BLDC. La commande est généralement envoyée sous forme de message CAN avec l'identifiant approprié et la charge utile contenant la valeur de la consigne de courant.

Commande d'arrêt d'urgence : cette commande permet d'arrêter immédiatement le moteur BLDC en envoyant un message CAN avec l'identifiant approprié et une charge utile indiquant que le moteur doit être arrêté d'urgence.

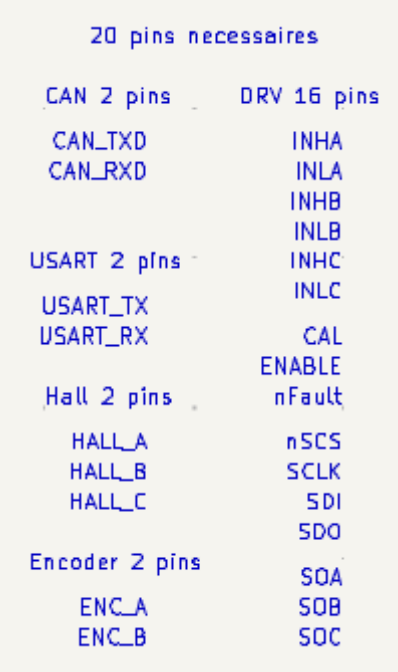
Bibliothèques CAN

https://community.st.com/s/question/0D50X00009XkeOsSAJ/stm32f4-can-bus-example-using-hal-library

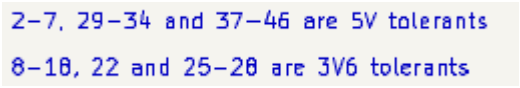
https://github.com/timsonater/stm32-CAN-bus-example-HAL-API

Configuration du microcontrôleur

Choix des pins



Certaines entrées sont compatibles avec un niveau de tension de 5V, d'autre uniquement avec 3V3.



Tous les composants à l'exception du DRV8323 (car Vref réglable) sont susceptible d'envoyer des signaux allant jusqu'à 5V.

Il faut donc interfacer ces signaux sur les pins compatibles.

Pour utiliser les pleines capacités du DRV8323 et avoir un contrôle indépendant des MOSFET, nous souhaitons avoir accès aux 6 sortie PWM du TIMER 1 (pins 27 à 32).

Il est important de placer le CAN car il y a peu de possibilité. Puis l'USART.

Les timers 1,2,3 et 8 sont capable de s'interfacer avec des encodeurs incrémentaux et des capteurs à effets Halls. Nous n'utiliserons pas la deuxième fonction dans un premier temps mais il est important de la laisser disponible pour de potentiels utilisations futures.

Le timer 1 et déjà utilisé et le timer 2 n'a plus de pin compatible 5V disponible.

Le timer 3 possède 2 pin compatible 5V et le timer 8 en possède 3.

Le timer 3 sera utilisé pour l'encodeur et le timer 8 pour les capteurs à effet Halls.

Il faut encore placer le bus SPI et son pin nCS que nous piloterons en software.

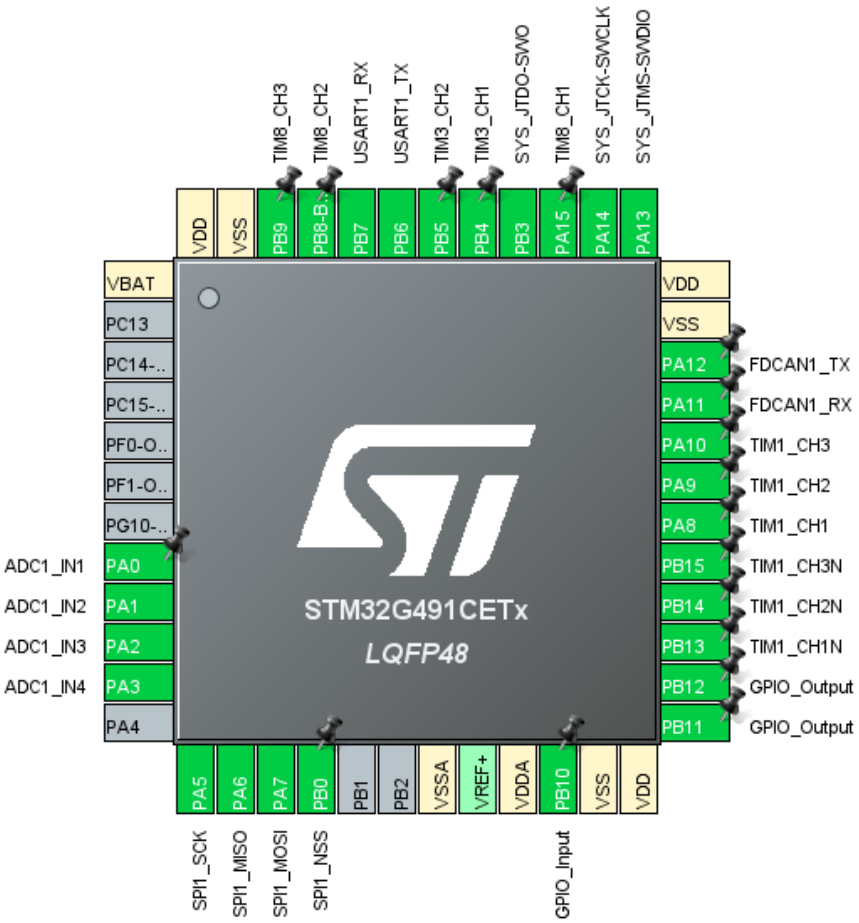
Il ne faut pas oublier Les pins de programmations et debug SWDIO SWCLK et idéalement SWO.

La dernière difficulté est de placer les quatre entrées ADC (3 pour la mesure du courant et 1 pour la mesure de tension d'alimentation). Le DRV8323 est réglé pour une plage de tension allant jusqu'à 3V3 et le pont diviseur sur Vin est aussi calculé pour 3V3. Il est donc pas nécessaire de prendre des entrées tolérant le 5V.

Il ne reste plus qu'à placer les 2 sortie CAL et ENABLE (pin 25 et 26) ainsi que l'entrée nFAULT.

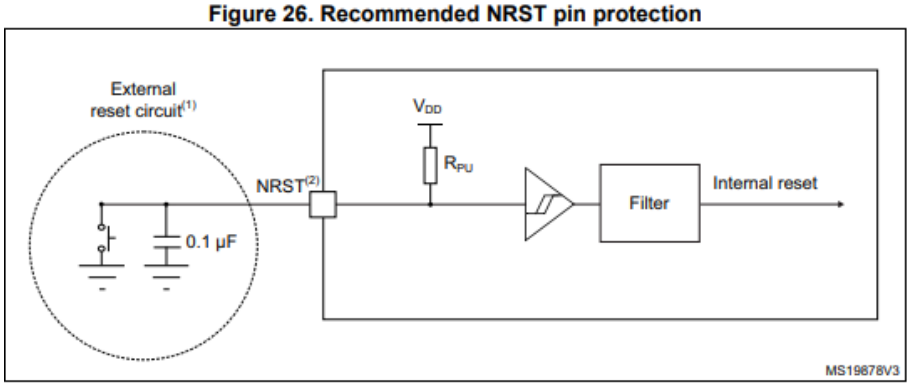
Les EXTI 4 5 8 9 et 15 sont utilisé par les capteurs halls et les encodeurs et on vaut pouvoir déclencher des interruptions si besoin. On souhaite également pouvoir en déclencher une sur nFault on lui choisira don un numéro de pin GPIO différent de ceux énuméré précédemment. On choisira ici PB10 (pin22).

Ces GPIO pourront être déplacé au moment du routage pour réduire les croisements.



Le pin 13 pourra servir de nRST si besoin.

Figure 26. Recommended NRST pin protection

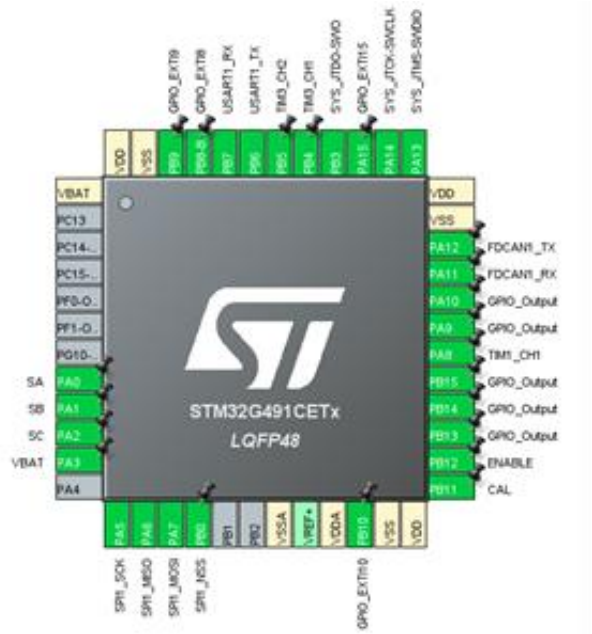


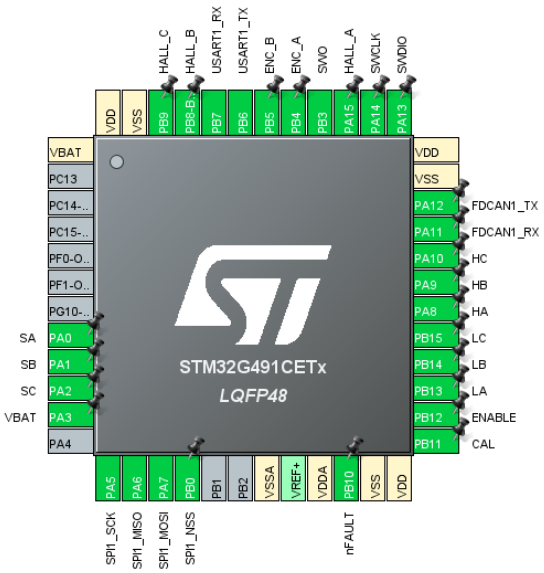
Configuration des pins

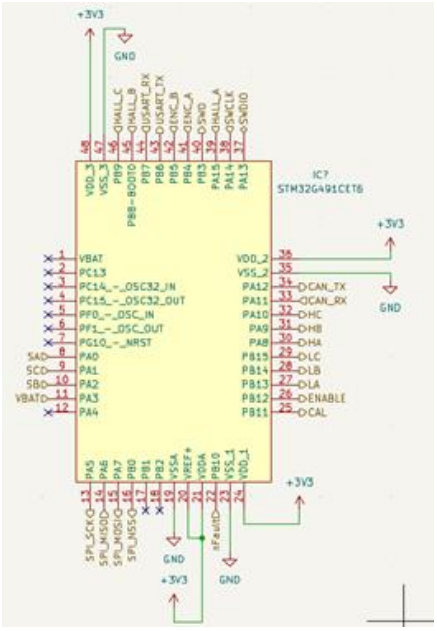
Nous avons anticipé les pins pour tirer pleinement profit du microcontrôleur dans le futur mais pour la configuration initiale nous allons activer uniquement ce qui nous sera utile.

Uniquement le Channel 1 du timer 1, le reste en sortie GPIO.

Les entrées des capteurs à effets Halls deviennent des interruption extérieur (EXTI).





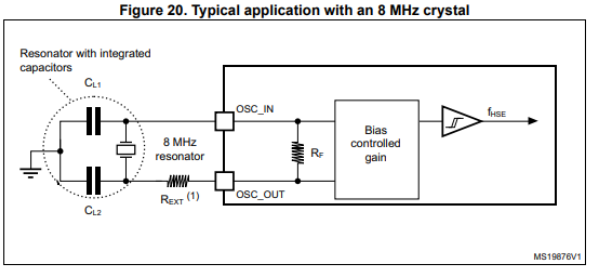


Pour obtenir une horloge plus précise on utilise un quartz externe.

Symbole : Device : Crystal_GND24_Small

Empreinte : Crystal : Crystal_SMD_EuroQuartz_MT-4Pin_3.2x2.5mm

Figure 20. Typical application with an 8 MHz crystal



1. R_{EXT} value depends on the crystal characteristics.

Fabrication

Nous avons choisi de faire sous-traiter la fabrication et l'assemblage de la carte si les composants sont stockés chez le fabricant et si les délais sont cohérés avec la durée du projet. Si non, l'assemblage sera réalisé à l'ENSEA.

Le fabricant choisis est JLC PCB dû au facilité de paiement pour l'ENSEA

Choix des composants

Microcontrôleur	STM32G491CET6		
Driver	DRV8323RSRGZT		
MOSFET	CSD18532Q5BT		
CAN	MCP2551-I/SN	datasheet	achat

Choix du microcontrôleur

Deux microcontrôleurs sont déjà stockés à l'ENSEA

STM32G431RBT6 :

Commercial Part ...	Part No	Reference	Mark...	Unit P...	Package	Flash	RAM	I/O	Frequency
STM32G431RBT6	STM32G431RB	STM32G431RBTx	Active	3.3398	LQFP 64 10x10x1.4 mm	128 kBytes	32 kBytes	52	170 MHz
STM32G431RBT6TR		STM32G431RBTx	Active	3.3398	LQFP 64 10x10x1.4 mm	128 kBytes	32 kBytes	52	170 MHz

STM32G491CET6 :

Commercial Part ...	Part No	Reference	Mark...	Unit P...	Package	Flash	RAM	I/O	Frequency
STM32G491CET6	STM32G491CE	STM32G491CETx	Active	4.3101	LQFP 48 7x7x1.4 mm	512 kBytes	112 kBytes	38	170 MHz

Nous avons choisi le STM32G491CET6 car il est plus petit est sont nombre de broche est suffisant.

Alimentation

DRV8323

Le driver intègre un Buck permettant de fournit 600mA avec une tension de sortie comprise entre 0.8V et 60V, ainsi qu'un régulateur 3V3 pouvant délivrer 30 mA.

Le driver possède 3 entrées pour l'alimentation : VM qui permet d'alimenter le driver, Vin qui sert d'entrée d'alimentation pour le Buck et Vref qui sert d'alimentation et de tension de référence pour les amplificateurs de mesure de courant.

VM et Vin seront connecté à la tension de la batterie (qui devra donc être inférieur à 60V). Vref sera connecté à la sortie du régulateur 3V3 intégré. La documentation indique que le courant max entrant dans Vref est de 5mA pour une tension de 5V. On a suffisamment de marge pour que le régulateur 3V3 intégré soit suffisant.

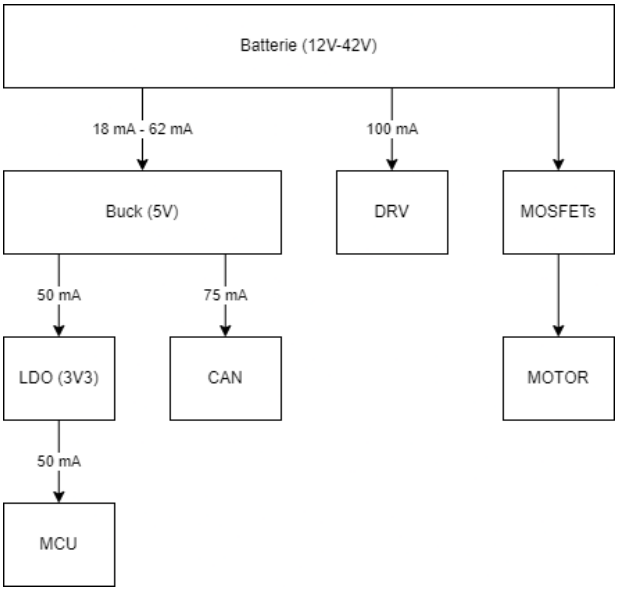
Il est important que Vref soit à 3V3 car le uC ne supporte le 5V que sur peu d'entrées. Les sorties du driver sont en open drain, c'est la pull-up qui définit la tension de sortie. On utilisera les pull-up interne du microcontrôleur.

STM32G491

Le uC doit être alimenté en 1V7 à 3V6, on utilisera une tension d'alimentation de 3V3. Sa consommation max est de 50mA d'après la datasheet. On choisira un LDO fournissant 3V3 à partir des 5V du Buck.

CAN

Le transceiver CAN nécessite une alimentation 5V et peut tirer jusqu'à 75mA. Le transceiver CAN sera donc alimenté directement par le 5V du Buck.



Architecture software

Pour valeur des capteurs à effet hall :

- 1 timer qui compte le temps entre deux mesures.
- Lire les 3 capteurs -> 3 GPIO.

Pour mesurer le courant dans chaque branche du pont :

- 1 timer qui se déclenche toutes les 'résolution' ms.
- 3 petite résistances et calcul de la tension aux bornes. 3 I/O - entrée ADC

Ou

- Utiliser le timer de la PWM, pour déclencher les mesures à chaque période de PWM.
- ADC et DMA pour les valeurs.
- filtrage, avec flottant c'est mieux.

Configuration du driver :

- 1 liaison SPI pour envoyer les paramètres au driver.

Génération de la PWM:

- 1 compteur avec output-compare

Finalement, on va utiliser un compteur pour tout faire :

- Il va générer la PWM avec un seuil,
- Il va compter-décompter,
- Il va lancer des conversions à chaque fois qu'il arrive aux max-min

En 'Live':

Générer la PWM, avec le Timer 1.

- Timer

Mesurer le courant tous les X millisecondes, et ajuster/limiter. (avec le capteur à effet hall)

- Timer avec interruptions, à choisir.

Recevoir les messages CAN qui arrive toutes les Z millisecondes.

- Voir si le CAN peut nous produire une interruption plutôt que d'avoir à vérifier en boucle avec un timer.
- Si on arrive à tout traiter suffisamment vite, alors pas besoin d'avoir des messages de freinage 'prioritaire' par rapport à d'autre.

Envoyer des messages sur l'état de la carte via le CAN toutes les Y millisecondes.

- Utiliser un timer avec Interruption.

Interruption si le driver met le PIN nfault à 0.

- Lire les erreurs en SPI

Agir en conséquence :

- Recopier l'erreur sur bus CAN ?
- Arrêter la carte ? Le moteur ? ou attendre un message CAN qui nous le demande explicitement

Au début :

Envoyer la config en SPI.

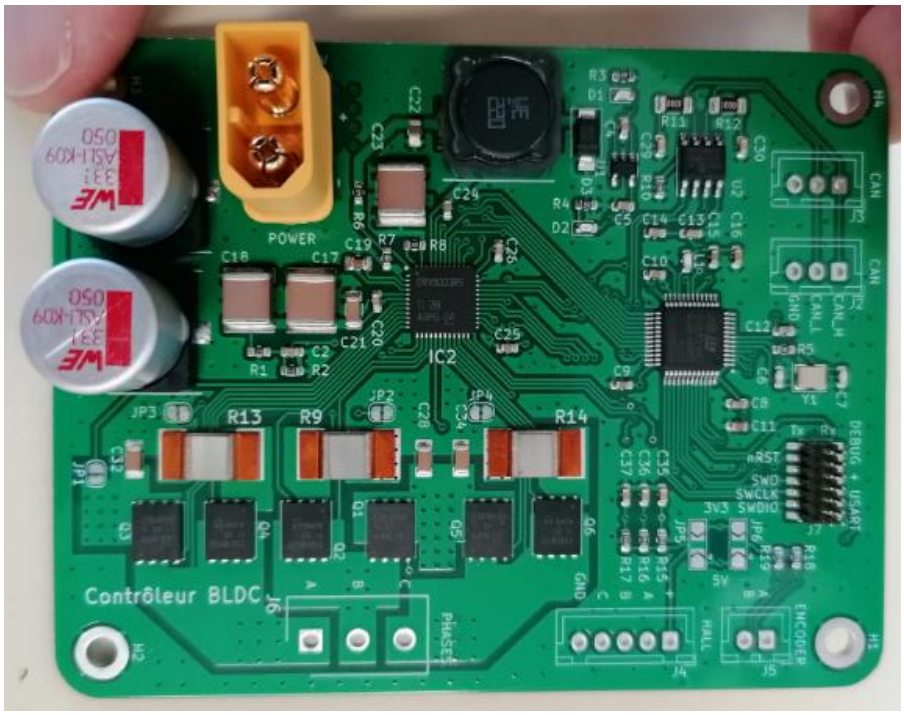
Lire les switches pour savoir notre adresse/configuration.

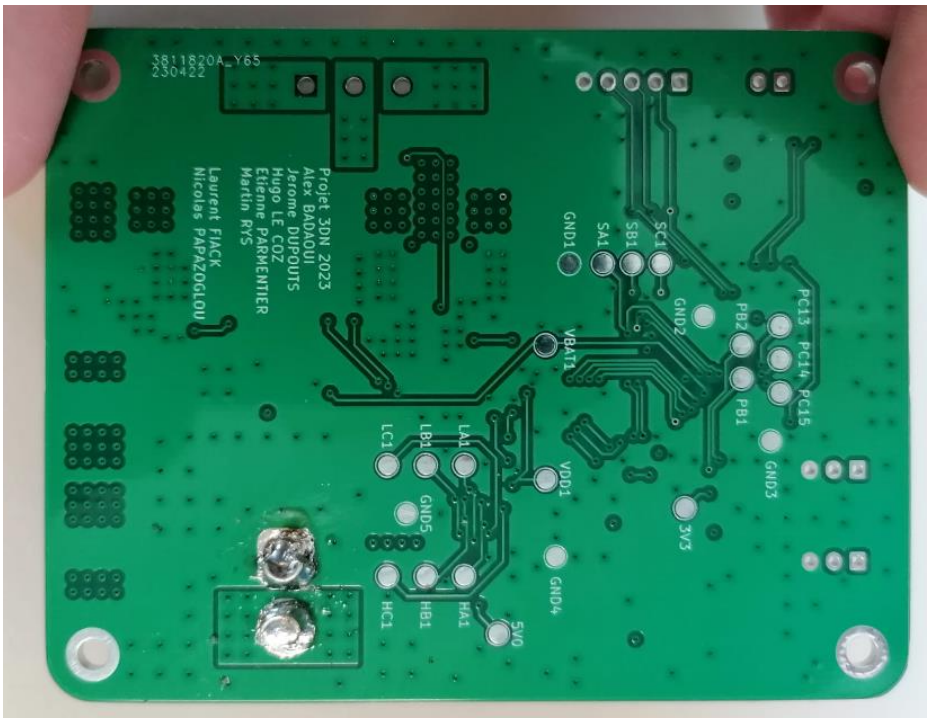
-> Fixer une horloge, un quartz peut être ?

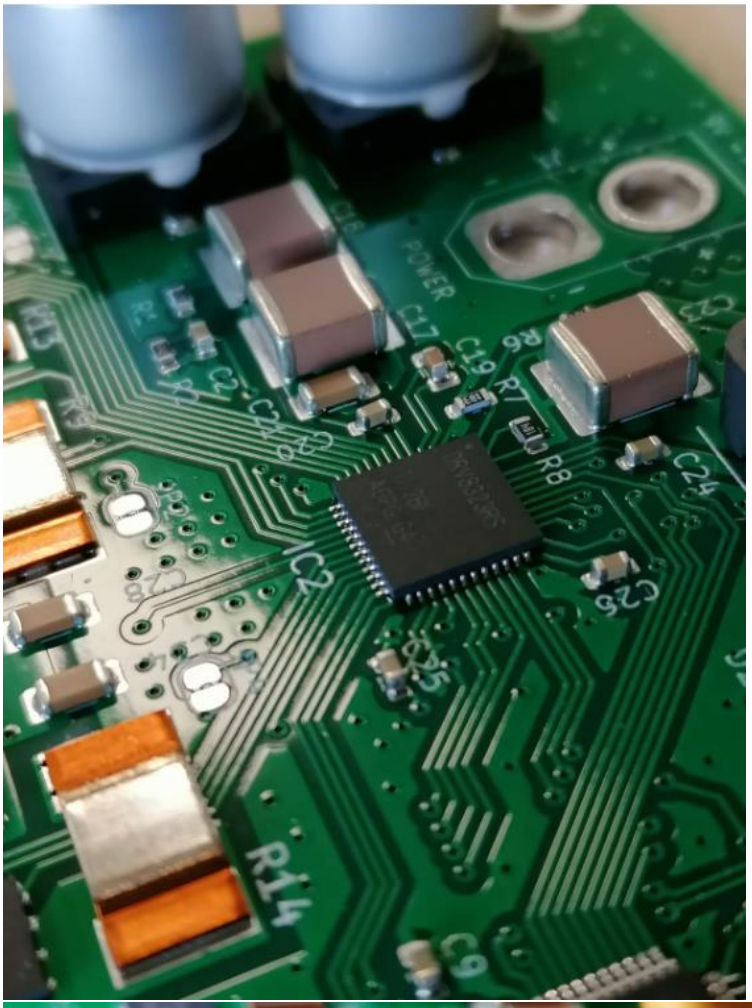
Fréquence max du mcu à 170 MHz. Donc à partir de là calculer toutes les fréquences des différents timers.

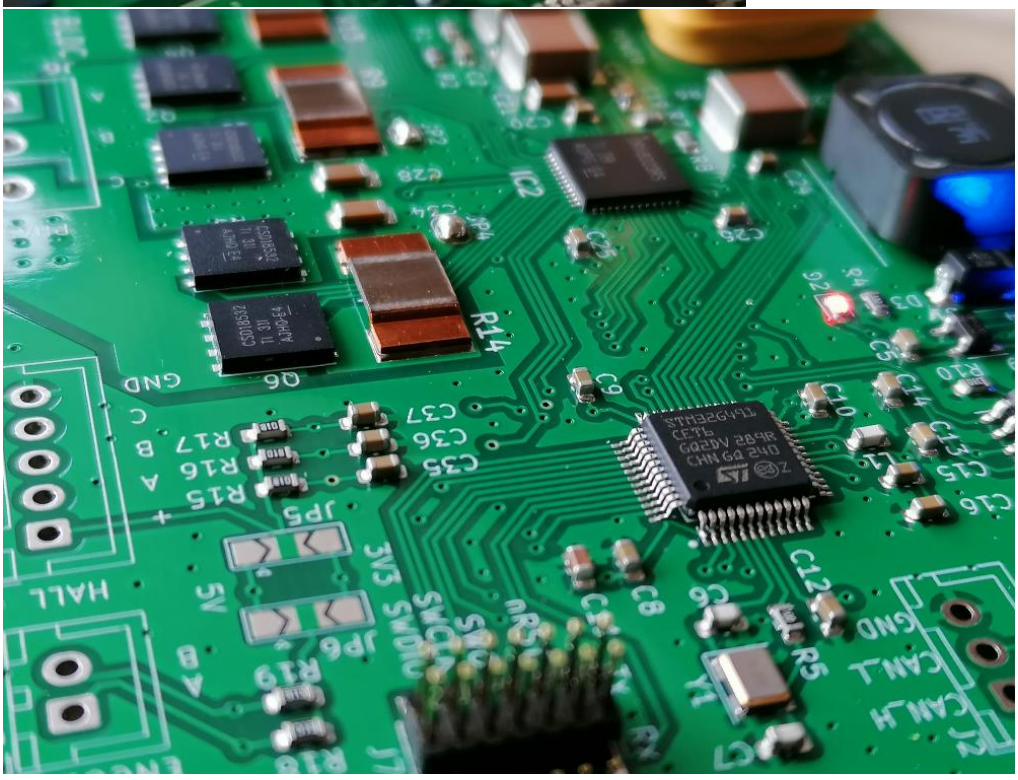
On sait que la fréquence de la PWM du moteur sera de ~40 kHz, ce qui évitera qu'elle s'entende.

Carte assemblée






POWER
C17
C19
R6
R7
R8
C23
C24
C25
IC2
R14
C9


IC2
R14
C9
GND
C37
C36
C35
R17
R16
R15
JP5
JP6
3V3
5V
SWDIO
SWCLK
HALL
C12
C8
R5
CAN_L
CAN_H
Y1