Encadrant : Mr. Nicolas PAPAZOGLOU
3^e**Année** : 2021/2022
Classe : **ESE**
Romain TACAIL
Wissam MAKHLOUF

Rapport TP Actionneur et Automatique Appliquée

ENSEA
ETABLISSEMENT PUBLIC
Cergy-Pontoise(95)
Ministère de l'Enseignement Supérieur et de la Recherche

Table des matières :

1. Objectif du TP

L'objectif de ce TP consiste à réaliser une interface de type « console » pour commander le hacheur d'un moteur. Dans un premier temps, la commande des 4 transistors du hacheur seront réalisés en commandes complémentaires décalées. Dans un deuxième temps, une acquisition des données des différents capteurs. Enfin, l'asservissement du moteur sera fait en temps réel.

2. TP n°1 : Commande MCC basique

2.1. Commandes accessibles & création d'un menu

L'objectif de cette partie et de pouvoir taper des commandes pour déclencher des actions par le biais d'un menu. Tout en gardant l'**Echo** activé à la réception des caractères, il s'agit de stocker les données obtenues dans un tableau de **char** avec un pointeur qui s'incrémentera à chaque caractère reçu.

Lorsque la touche **entrée** est détectée le code doit pouvoir remplir les fonctions ci-dessous :

- Traitez la chaine de caractères en la comparant aux commandes connues, pour le moment nous resterons à un nombre limitée de commandes :
  - help : qui affiche toutes les commandes disponibles,
  - pinout : qui affiche toutes les broches connectées et leur fonction
  - start : qui allume l'étage de puissance du moteur (pour l'instant nous ne ferons qu'afficher un message de "Power ON" dans la console)
  - stop : qui éteind l'étage de puissance du moteur (pour l'instant nous ne ferons qu'afficher un message de "Power OFF" dans la console)
  - toute autre commande renverra un message dans la console "Command not found".
- Videz la chaine de caractère et mettez l'index pointant vers le prochain caractère à remplir à 0.

Figure 1 : Cahier des charges du code.

La fonction **help**, permet d'afficher toutes les commandes disponibles, effectuée la forme suivante :

```
71 const uint8_t help[128] = {"\r\npinout:contient la liste des pin utilisées\r\nstart: démarre le moteur \r\nst
```

La commande **start** affiche le texte **Lancement du moteur** :

```
73 const uint8_t power_on[CMD_BUFFER_SIZE]={"\r\nLancement du Moteur\r\n"};
```

La commande **stop** affiche le texte **Arrêt du moteur** :

```
const uint8_t power_off[CMD_BUFFER_SIZE]={"\r\nArret du Moteur\r\n"};
```

Toutes ces commandes sont exécutées lorsque la touche entrée est enfoncée, ce qui correspond à **\b**. Ces fonctions sont reconnues en utilisant la fonction **strcmp**, de la manière suivante :

```
if(strcmp(c,"help")==0) {
    Uartprint((char *) help);
    return 1;
}
```

2.2. Génération de 4 PWM

Pour commander le moteur MCC, il est nécessaire de générer 4 signaux **PWM**. Ces 4 signaux seront générés via **TIM1**, le cahier des charges est le suivant :

- Fréquence de la **PWM** : 16 kHz
- Temps mort minimum : 2 µs
- Résolution minimum : 10 bits

Lors de la réalisation du test, le rapport cyclique des signaux seront fixés à 60%. La **figure 2** ci-dessous illustre ces 4 signaux vus sur l'oscilloscope.

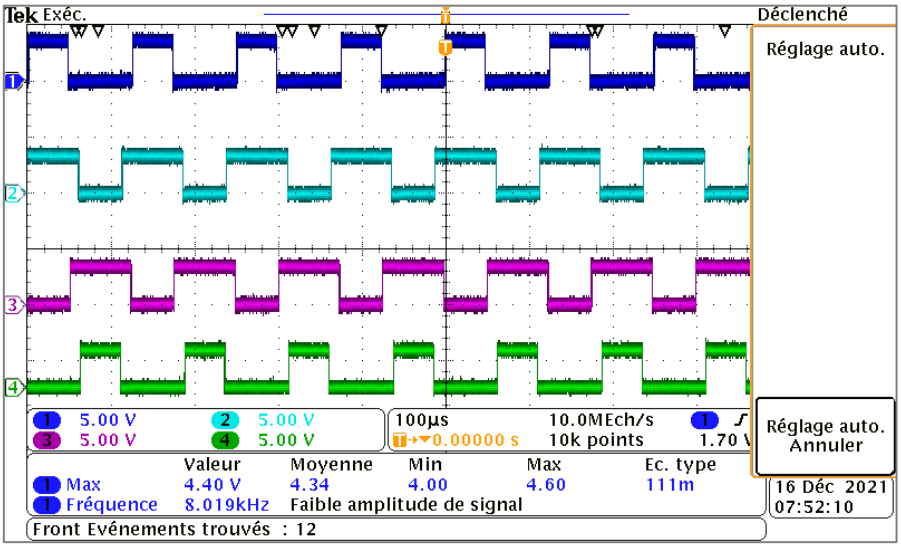

*Figure 2 : Signaux **PWM** issus de **TIM1**.*

Ces 4 signaux **PWM** permettent de contrôler l'ouverture ou la fermeture des transistor qui compose la machine à courant continu. Parmi ces 4 signaux, 2 d'entre eux correspondent à des **TOP** (signaux **1** et **2** de la **figure 2**). Ils ne sont pas à l'état haut en même temps et comprennent un « temps mort » pour que les transistors ne conduisent pas en même temps. Ces 2 signaux **TOP** comprennent également des signaux complémentaires (**BOTTOM**). Les signaux **BOTTOM** des signaux **1** et **2** sont respectivement les signaux **3** et **4**.

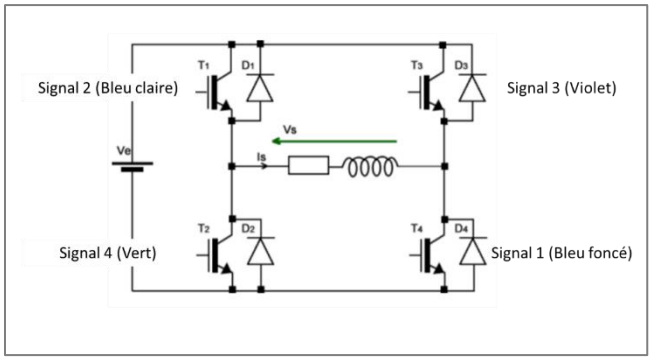

Figure 3 : Transistors piloté par les signaux de commande.

La **figure 3** ci-contre précise quel signaux commande chacun des transistor de la MCC. Les transistors $\mathbf{T_1}$ et $\mathbf{T_3}$ conduisent en même pendant que $\mathbf{T_2}$ et $\mathbf{T_4}$ seront ouverts.

2.3. Prise en main du hacheur et câblage

Le hacheur utilisé dans ce TP comprend de nombreuses broches d'entrée/sortie. La datasheet du hacheur contient la liste des broches à alimenter, à relier à la masse et de commande **PWM**.

Les broches à alimenter sont les suivantes :

- **Broches 12** : Yellow Phase Top Switch Firing Command.
- **Broche 13** : Red Phase Top Switch Firing Command.
- **Broche 30** : Yellow Phase Bottom Switch Firing Command.
- **Broche 31** : Red Phase Bottom Switch Firing Command.
- **Broches 33** : Fault Reset Command (réalise la séquence de démarrage).

Le câblage du hacheur est présenté en **figure 4** ci-dessous.

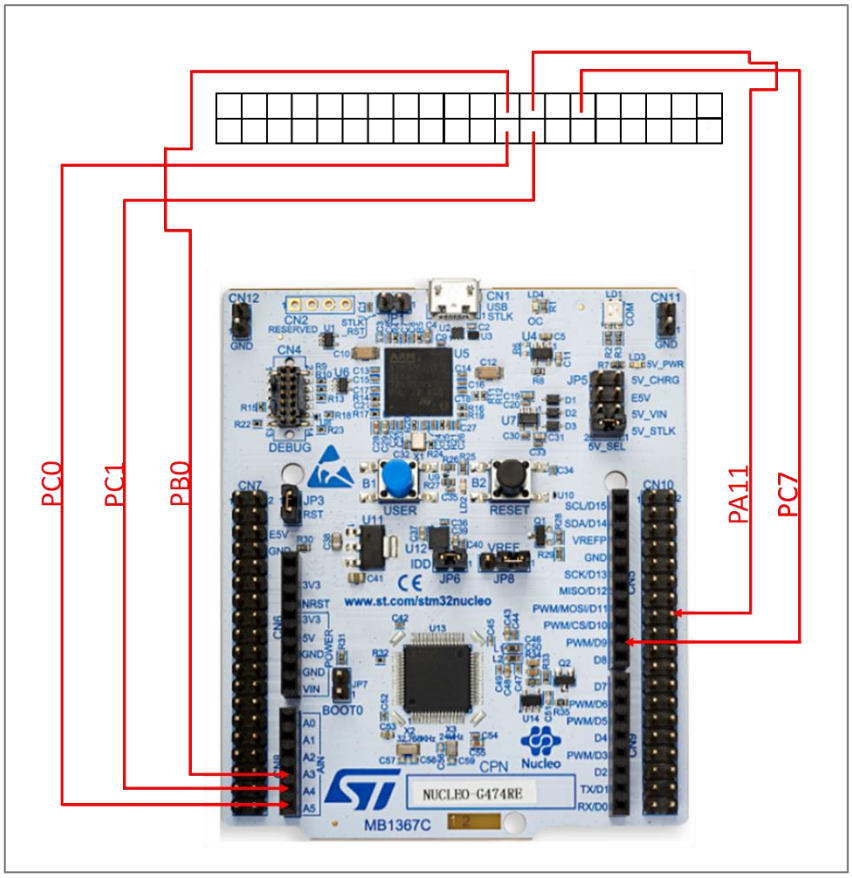

Figure 4 : Câblage du hacheur.

2.4. Commande de start

Le hacheur a besoin d'une séquence d'amorçage pour obtenir une tension de sortie. Dans cette sous partie, il s'agit de mettre en place un code visant à déclencher la séquence d'allumage du moteur. Le code sera établi de manière à activer la séquence d'allumage du moteur via l'entrée de la commande start et via le bouton bleu de la carte **Nucleo-G431RB**. Le **GPIO** relatif à ce bouton est le **GPIO PC13**. Les codes permettant de déclencher les différentes séquences d'allumages du moteur sont présenté ci-dessous

```
else if(strcmp(c,"start")==0) {
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET);
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);
    HAL_Delay(1);

    Uartprint((char *) power_on);
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
    HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
    HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
    TIM1->CCR1 = val;
    TIM1->CCR2 = TIM1->ARR - val;
}
```

*Figure 5 : Code permettent de déclencher la séquence d'allumage du moteur par le biais de la commande **start**.*

```
236 void EXTI15_10_IRQHandler(void)
237 {
238     /* USER CODE BEGIN EXTI15_10_IRQn 0 */
239     HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
240     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_SET);
241     HAL_Delay(1);
242     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);
243     HAL_Delay(1);
244
245     TIM1 -> CCR2 = TIM1->ARR - (TIM1 -> CCR1);
246     HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
247     HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
248     HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
249     HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_2);
250     //TIM1->CCR1 = val;
251     //TIM1->CCR2 = TIM1->ARR - val;
```

Figure 6 : Code permettent de déclencher la séquence d'allumage du moteur par le biais d'un appui sur le bouton bleu.

2.5. Commande de vitesse

3. TP n°2 : Mesure de vitesse et de courant

Dans cette partie, le but est de mesurer la vitesse du moteur. Pour cela, il est nécessaire d'exploiter les codeurs incrémentaux présents sur chacun des moteurs. L'encodeur doit tout d'abord être connecté au microcontrôleur. La **figure 7** ci-dessous illustre la modification au câblage du microcontrôleur

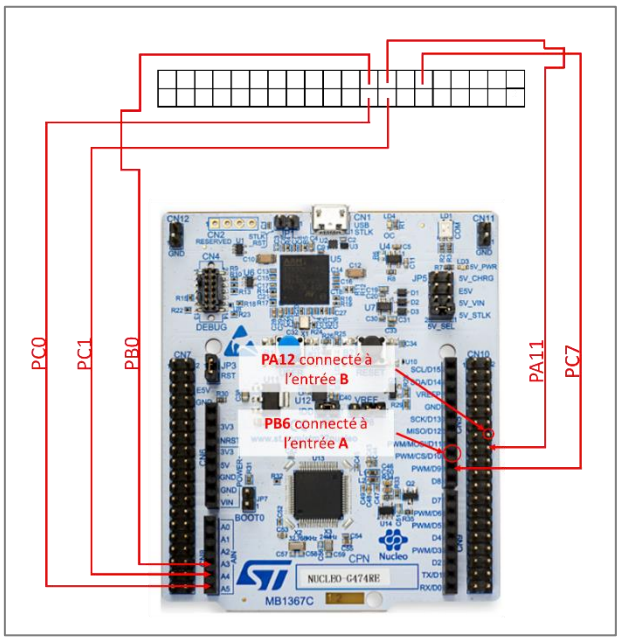

Figure 7 : Câblage du microcontrôleur avec encodeur.

Pour calculer la vitesse, il faut tout d'abord relever les valeurs de l'encodeur. À partir de ces valeurs d'encodeur, il est alors possible de déterminer la vitesse. Le principe consiste à comparer plusieurs valeurs les unes après les autres. Un **Timer** est utilisé en mode encodeur (**Encoder mode**) et la valeurs est récupérée dans le code. Le paramétrage de ce **Timer** est présenté en **figure 8** ci-dessous.

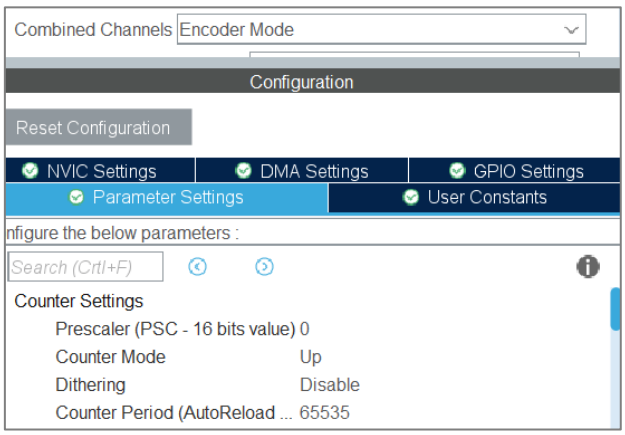

*Figure 8 : Paramétrage du **Timer**.*

Le paramètre **ARR** est initialisé à sa valeur maximale, à savoir **65535** car il est codé sur 16 bits. Ainsi, ce **timer** relèvera régulièrement les valeurs de l'encodeur. Pour effectuer le calcul de la vitesse, un deuxième **timer** est nécessaire. En effet, celui-ci fonctionne par interruption et sert à comparer deux valeurs d'encodeur. Une fois la comparaison faite, il est nécessaire de diviser cette valeur par le nombre de crans par tour, soit **1024**.

```
vitesse = (valeur_nouvelle-valeur_ancienne)/1024;
valeur_ancienne = valeur_nouvelle;
printf("Vitesse = %d tr/s\r\n", vitesse);
```

La **figure 9** ci-dessous présente l'affichage de la vitesse du moteur.

```
6080
6189
Vitesse = 4 tr/s
```

Figure 9 : Relevé de la vitesse du moteur.