



Université Paris-Saclay
UFR des Sciences
L3 E3A

Travail d'étude en électronique *Asservissement numérique*

Réalisé par :

BENAMARA Mohammed Younes
BENALI Lounes
CHAOUCHAOUI Mohamed Nazim

Table des matières

1	Introduction	3
2	Logique derrière le schéma	3
2.1	Bloc transmission et amplification du signal	4
2.2	Bloc moteur	5
2.3	Bloc détection	6
3	Dimensionnement	7
3.1	Résistance R_1	7
3.2	Résistance $R_2 = R_3$	8
3.3	Capacité C_1	9
3.4	Capacité de lissage C_2	9
3.5	Résistance R_4	9
4	Schématisation et routage	10
5	Impression et soudure	12
5.1	Mise en liaison	13
6	Asservissement et Programmation Mbed	14
6.1	Théorie	14
6.2	Code	14
7	Pour Conclure	18

1 Introduction

Le projet consiste en la réalisation d'un système d'asservissement en fréquence et en phase d'un moteur à courant continu à l'aide d'un microcontrôleur de la famille ARM Cortex, plus précisément lea platine NUCLEO-F411RE de STM. Pour ce faire, nous utiliserons la plateforme de développement en ligne Keil Studio pour écrire le programme en C++ qui servira de correcteur pour l'asservissement.

Ce projet a pour objectif d'étudier plusieurs techniques de base dans le domaine de l'électronique et de la programmation. Tout d'abord, nous verrons comment convertir des signaux numériques en signaux analogiques à l'aide de la modulation de largeur d'impulsion (PWM). Ensuite, nous étudierons l'électronique de commande d'un circuit de puissance, en particulier l'utilisation d'un hacheur à transistor et d'opto-coupleurs. Enfin, nous aborderons la théorie des asservissemens de type proportionnelle-intégrale et nous mettrons en pratique cette théorie dans le cadre de ce projet.

Le projet comporte donc deux parties principales : une partie "électronique" où nous réaliserons le circuit de commande du moteur et une partie "programmation" où nous écrirons le programme qui servira de correcteur pour l'asservissement. Ce projet permettra donc de mettre en pratique des connaissances en électronique et en programmation, ainsi que de découvrir les principes de base des systèmes d'asservissement.

2 Logique derrière le schéma

On pourrait se dire qu'un seul transistor suffirait à commander le moteur. Cependant un transistor ne peut pas fournir assez de courant pour alimenter un moteur, l'autre raison est qu'il génère également une chute de tension importante lorsqu'il est utilisé pour commuter un courant élevé, ce qui peut entraîner une perte de puissance et une dissipation thermique excessive. Pour éviter cela, un circuit de commande de moteur plus complexe est généralement utilisé, qui utilise des transistors de puissance et des circuits de protection pour assurer un fonctionnement fiable et efficace du moteur.

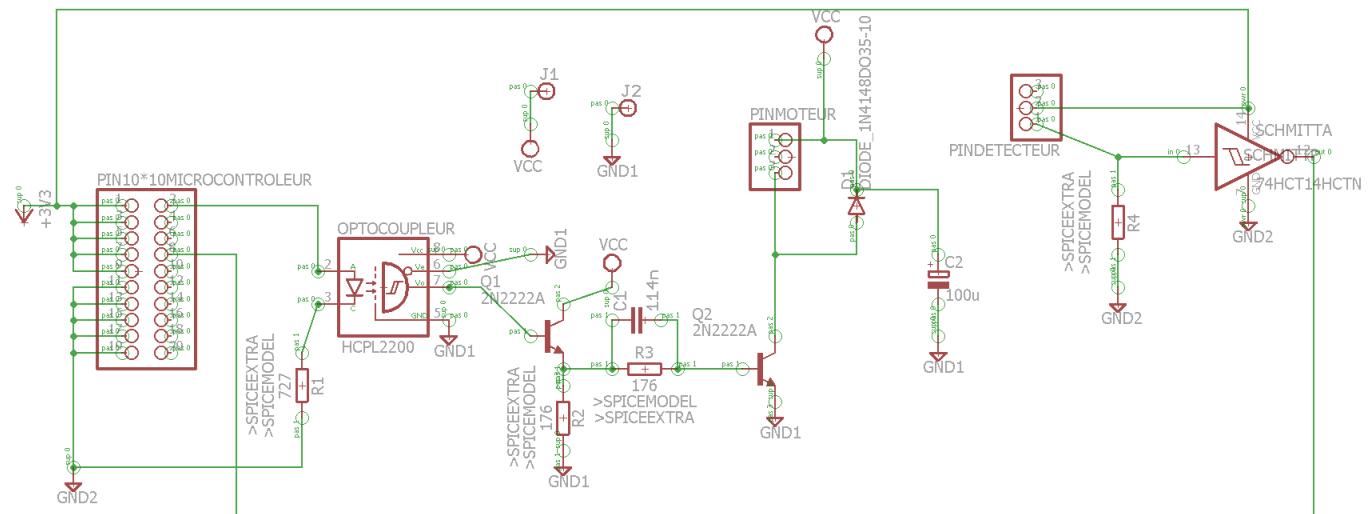
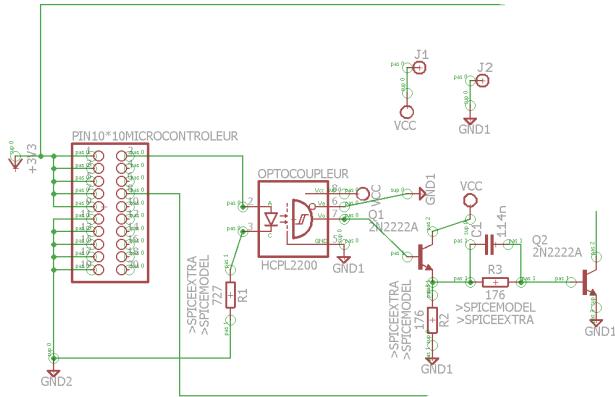


FIGURE 1 – Schéma totale du projet

2.1 Bloc transmission et amplification du signal



2.2 Bloc moteur

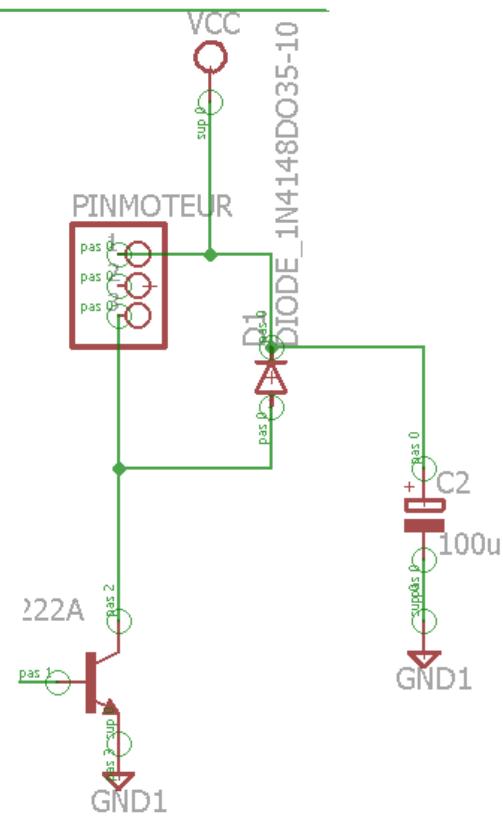


FIGURE 3 – Schéma du bloc moteur

Le moteur sera mis en parallèle à une diode de roue libre, permettant au circuit d'être fermé lors de la commutation, et ce afin d'éviter le déchargement des bobines moteur (arc électrique). Enfin on mettra en parallèle de l'ensemble moteur-diode-transistor une capacité de lissage C_2 , qui sert à garder une tension d'alimentation constante lors des commutations.

2.3 Bloc détection

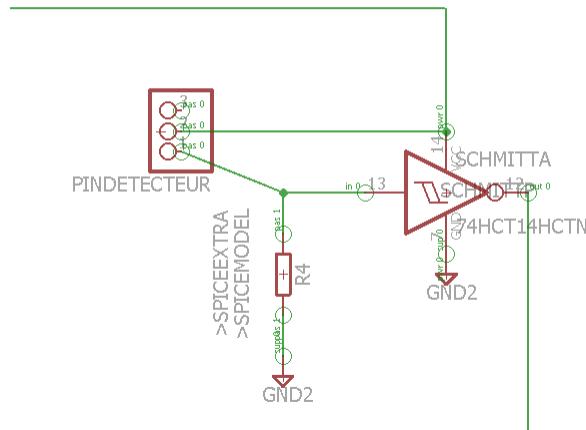


FIGURE 4 – Schéma du bloc détection

Sur l'arbre du moteur est encastré un disque doté d'une fente, à travers laquelle une photodiode émet des Rayons IR, captés par une autre photodiode , une résistance R_d est placée en série pour la protéger. Le signal de sortie de la seconde photodiode est ensuite transmis à un trigger de Schmidt, qui convertit le signal d'entrée analogique en un signal de sortie numérique stable et fiable. Il élimine les rebonds et les bruits sur le signal d'entrée, la resistance R_4 ici servant à canaliser le courant afin d'avoir une lecture en tension convenable en entrée du trigger de schmidt.

3 Dimensionnement

3.1 Résistance R_1

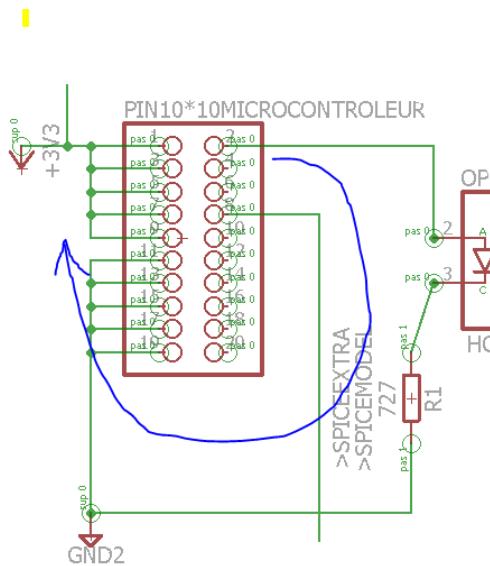


FIGURE 5 – Maille utilisée pour le dimensionnement

La maille utilisée est composé de trois éléments : le signal PWM en 3.3V, la résistance R_1 ainsi que la diode de l'optocoupleur. Pour cette dernière afin d'obtenir les informations pertinentes la concernant, nous utiliserons le datasheet de l'optocoupleur :

Recommended Operating Conditions

Parameter	Symbol	Min.	Max.	Units
Power Supply Voltage	V_{CC}	4.5	20	V
Enable Voltage High	V_{EH}	2.0	20	V
Enable Voltage Low	V_{EL}	0	0.8	V
Forward Input Current	$I_{F(ON)}$	1.6*	5	mA
Forward Input Current	$I_{F(OFF)}$	–	0.1	mA
Operating Temperature	T_A	0	85 ^[1]	°C
Fan Out	N		4	TTL Loads

Input Current Hysteresis	I_{IHYS}		0.12		mA	$V_{CC} = 5 \text{ V}$
Input Forward Voltage	V_F		1.5	1.7	V	$T_A = 25^\circ\text{C}$
Input Reverse Breakdown Voltage	BV_R	5			V	$I_R = 10 \mu\text{A}$

*The initial switching threshold is 1.6 mA or less. It is recommended that 2.2 mA be used to permit at least a 20% CTR degradation guardband.

FIGURE 6 – Datasheet HCPL2200

On trouve alors une tension seuil (Forward input Voltage) V_0 maximal de 1.7V et le courant correspondant maximal I_0 (Forward input current) est de 5 mA. On prends les valeurs limites pour que le dimensionnement de la résistance correspondent à n'importe quelle optocoupleur HCPL2200.

La loi des mailles donne :

$$3.3 = V_0 + R_1 I_0$$

donc :

$$R_1 = \frac{3.3 - V_0}{I_0} = 320\Omega$$

3.2 Résistance $R_2 = R_3$

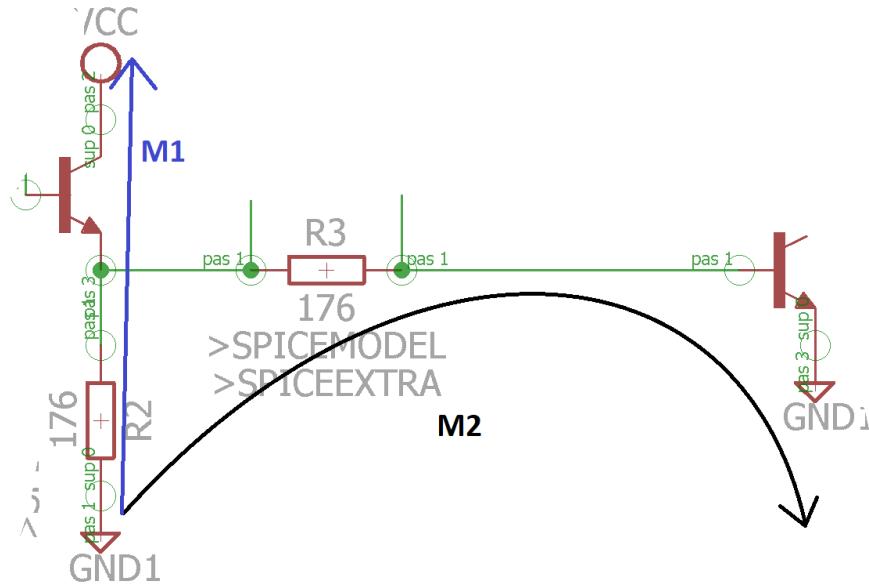


FIGURE 7 – Mailles utilisées pour le dimensionnement

Après passage par l'optocoupleur, le signal arrive au transistor, ayant deux inconnues dans la maille M1 (R_2 et I_2), il est nécessaire d'inclure une autre maille (M2) afin de déterminer la résistance R_2 (respectivement R_3). La première maille est composée d'un transistor, d'une tension d'alimentation 10V et R_2 . La deuxième maille quant à elle, est composée de R_2 , R_3 et d'un transistor. On aura donc besoin pour la maille 1 de V_{CE} et pour la maille 2 de V_{BE} et I_B , valeurs qu'on trouvera sur le datasheet du composant :

$V_{CE(sat)*}$	Collector-Emitter Saturation Voltage	$I_C = 150 \text{ mA}$ $I_C = 500 \text{ mA}$	$I_B = 15 \text{ mA}$ $I_B = 50 \text{ mA}$		0.3 1	V V
$V_{BE(sat)*}$	Base-Emitter Saturation Voltage	$I_C = 150 \text{ mA}$ $I_C = 500 \text{ mA}$	$I_B = 15 \text{ mA}$ $I_B = 50 \text{ mA}$	0.6	1.2 2	V V

FIGURE 8 – Datasheet 2N2222A

On posera $R_2 = R_3 = R$ La maille M1 donne :

$$V_{CC} - V_{CE} = V_{R_2}$$

$$10 - 0.3 = 9.7 = V_{R_2}$$

La maille M2 donne :

$$V_{BE} + V_{R_3} = V_{R_2}$$

$$0.9 + R * I_B = 9.7$$

Donc :

$$R = \frac{9.7 - 0.9}{5 * 10^{-2}} = 176 \Omega$$

3.3 Capacité C_1

Le temps caractéristique idéale pour le bloc R_3C_1 serait de $\tau = 0.02s$. Donc :

$$R_3 * C_1 = 0.02$$

$$C_1 = \frac{R_3}{0.02} = 114 \text{ nF}$$

3.4 Capacité de lissage C_2

Le moteur à notre disposition fournit, sous tension de 24V un couple de 5769 Tr/min ainsi que $0.16 * 10^{-3} Nm$, on peut alors en déduire la valeur de la capacité :

$$\frac{C_2 V^2}{2} = T\omega$$

Donc :

$$C_2 = \frac{0.16 * 604 * 10^{-3}}{288} = 335 \mu F$$

3.5 Résistance R_4

Le capteur peut être représenté comme montré sur la figure 10 par une photodiode, une résistance ainsi qu'un transistor contrôlé par les émissions de la photodiode.

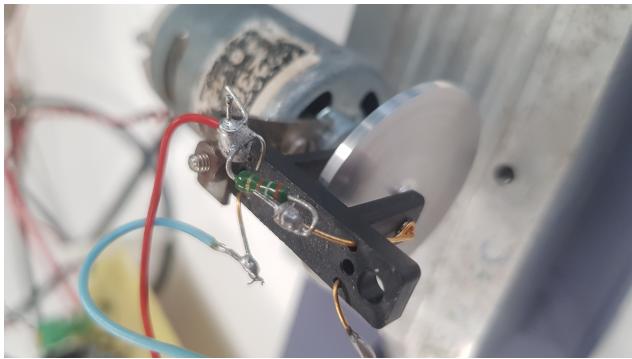
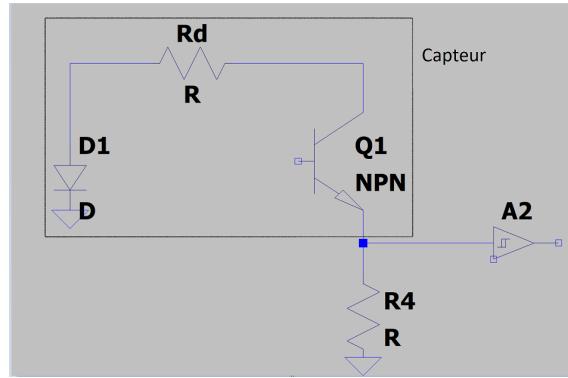
FIGURE 9 – Résistance $R_d = 180\Omega$ 

FIGURE 10 – Schéma partie détection

les donnés nécessaires à la determination de la résistance R_4 sont la tension seuil V_0 de la diode, la valeur de R_d , la tension V_{CE} et le courant I_E du transistor ainsi que le courant d'entrée sur le trigger de schmitt.

Collector-Emitter Saturation Voltage	$V_{CE(SAT)}$	V	$I_r=30 \text{ mA}$
HOA1877-001	0.4		$I_c=10 \mu\text{A}$
HOA1877-002	0.4		$I_c=60 \mu\text{A}$
HOA1877-003	1.1		$I_c=190 \mu\text{A}$

FIGURE 11 – $V_{CE} = 1.1V$, $I_E = 190\mu\text{A}$

ELECTRICAL CHARACTERISTICS (25°C unless otherwise noted)					
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS
IR Emitter					
Forward Voltage	V_f			1.6	V
Reverse Leakage Current	I_h			10	μA
					$I_p=20 \text{ mA}$
					$V_{rh}=3 \text{ V}$

FIGURE 12 – $V_0 = 1.6V$

La loi des mailles nous donne :

$$R_4 * I_4 = V_{CE} + R_d * I_c + V_0$$

La loi des noeuds permet d'écrire :

$$I_4 = I_E - I_S = (190 - 0.1)10^{-6} \text{ A}$$

en remplaçant dans la loi des mailles :

$$R_4 = \frac{V_{CE} + R_d * I_c + V_0}{(190 - 0.1)10^{-6}} = 14.398K\Omega$$

4 Schématisation et routage

Cette partie se fait sous le logiciel eagle, éditeur de schéma et logiciel de routage afin de concevoir le schéma de la carte, et ce en utilisant la bibliothèque de composant fourni. Le résultat est comme suit :

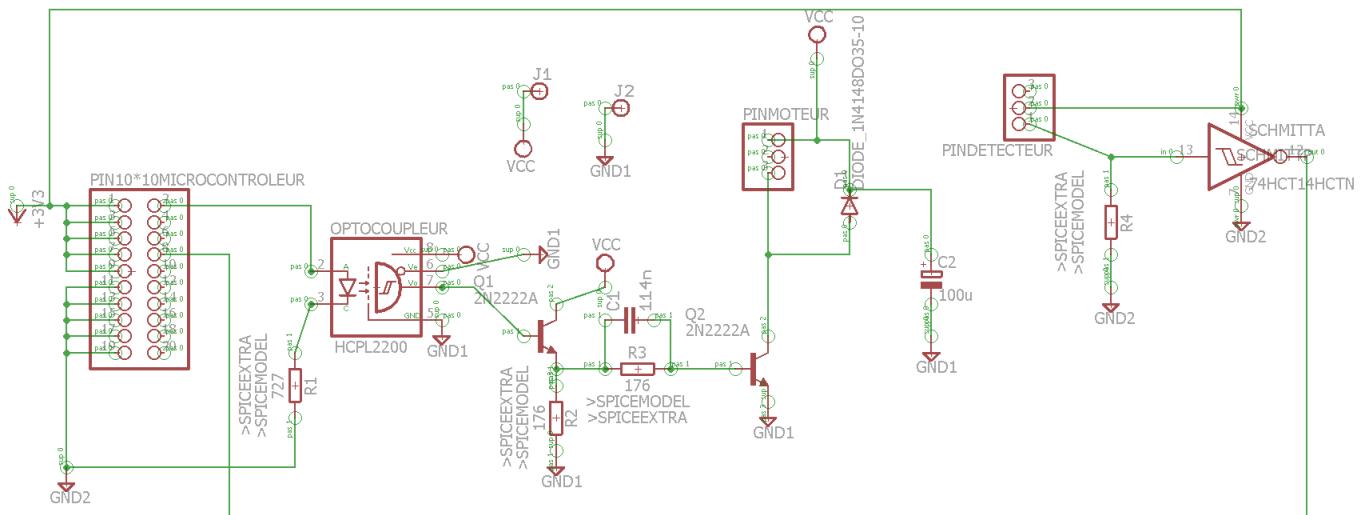


FIGURE 13 – Schéma de la carte

Après conception de la carte, on passe au routage de toutes les pistes du circuit, cela permet de relier physiquement les différents composants électroniques entre eux en utilisant des pistes de cuivre sur la carte du circuit imprimé.

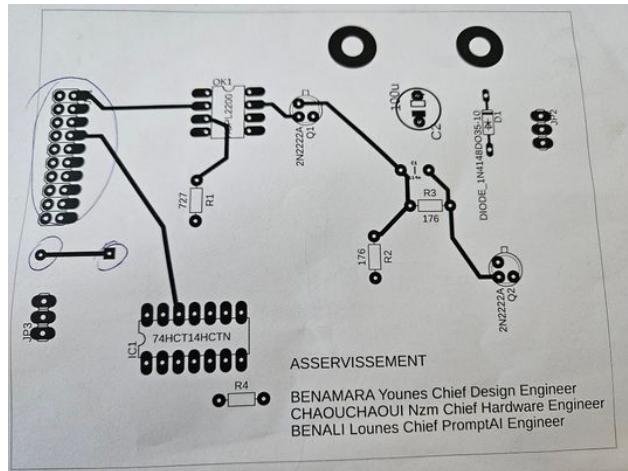


FIGURE 14 – Routage de la carte

5 Impression et soudure

Après avoir conçu notre carte sur logiciel, vient la conception Hardware. Cela passe par plusieurs étapes :

1. On imprime tout d'abord sur papier transparent les parties Top et Bottom, puis on glisse entre les deux la carte qui pour l'instant n'est qu'une plaque de cuivre et de résine photosensible protégée par une fine couche de plastique que l'on ôtera à cette étape.
2. Exposition à la lumière ultraviolette : La carte électronique recouverte de résine photosensible est exposée à la lumière ultraviolette à travers un film photographique. Les zones exposées à la lumière durcissent la résine photosensible, tandis que les zones non exposées restent solubles.
3. Développement de la résine photosensible : Après l'exposition à la lumière ultraviolette, la carte électronique est développée en utilisant une solution chimique pour enlever la résine photosensible soluble, laissant ainsi les zones de cuivre protégées.
4. Enlèvement du cuivre : Les zones de cuivre non protégées sont enlevées à l'aide d'une solution chimique, laissant ainsi les pistes de cuivre désirées.
5. Perçage des trous : Les trous nécessaires pour monter les composants électroniques sont percés à l'aide d'une machine de perçage.
6. Soudure des composants : Les composants électroniques sont soudés sur la carte électronique à l'aide d'une machine de soudure.

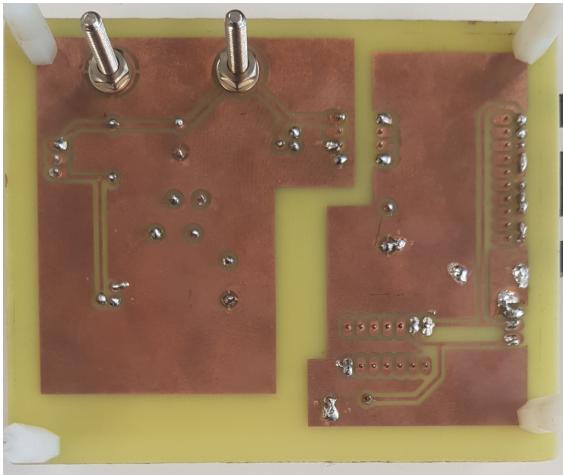


FIGURE 15 – Bottom de la carte

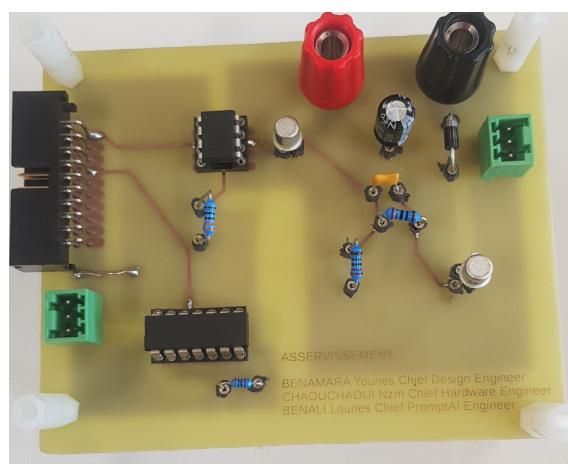


FIGURE 16 – Top de la carte

5.1 Mise en liaison

Après avoir conçu la carte on la relie dans un premier temps à un GBF produisant un signal carré au lieu du microcontrôleur, afin de s'assurer du bon fonctionnement de la carte.

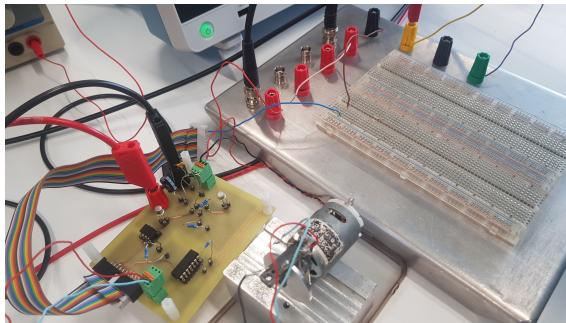


FIGURE 17 – Montage de test

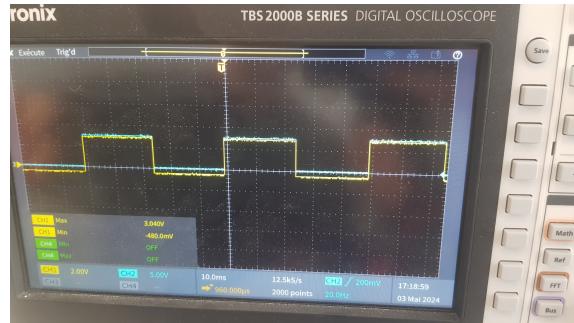


FIGURE 18 – Sortie de l'optocoupleur



FIGURE 19 – Sortie du trigger de schmitt

6 Asservissement et Programmation Mbed

6.1 Théorie

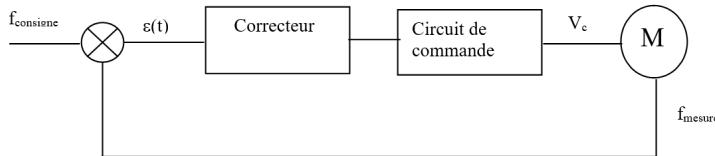


Schéma de la Boucle d'Asservissement

Le contrôleur PI (Proportional-Integral) est une méthode de contrôle de rétroaction couramment utilisée pour contrôler la vitesse d'un moteur. Le contrôleur PI utilise une combinaison de correction proportionnelle et intégrale pour ajuster la sortie de commande du moteur en fonction de l'erreur de vitesse mesurée. La correction proportionnelle ajuste la sortie de commande en fonction de l'erreur de vitesse actuelle, tandis que la correction intégrale ajuste la sortie de commande en fonction de l'intégrale de l'erreur de vitesse sur le temps.

Cependant, la correction PI seule peut ne pas suffire à maintenir une vitesse constante du moteur en raison de perturbations externes ou de changements de charge. Dans ce cas, un correcteur de phase peut être ajouté au contrôleur PI pour améliorer la stabilité de la vitesse du moteur. Le correcteur de phase ajuste la sortie de commande en fonction de la différence de phase entre la référence de vitesse et la vitesse mesurée du moteur.

En utilisant une combinaison de correction PI et de correction de phase, le contrôleur peut maintenir une vitesse constante du moteur même en présence de perturbations externes ou de changements de charge.

6.2 Code

Le code commence par définir les entrées/sorties et les variables nécessaires au programme. 'speed' est la sortie PWM qui contrôle la vitesse du moteur. 'sensor' est l'entrée numérique qui lit les impulsions du capteur de vitesse. 'ref_gbf' est l'entrée numérique qui lit les impulsions de la référence de vitesse. 'counter' et 'counter_ref' sont des minuteries qui mesurent respectivement la période du capteur de vitesse et de la référence de vitesse. 'watchdog_timer_sensor' et 'watchdog_timer_ref' sont des minuteries de détection d'anomalie qui vérifient si les signaux du capteur de vitesse et de la référence de vitesse sont toujours présents. 'sensor_period' et 'ref_period' sont les périodes mesurées du capteur de vitesse et de la référence de vitesse. 'delta_t' est la différence de temps entre l'impulsion du capteur de vitesse et l'impulsion de la référence de vitesse, utilisée pour la correction de phase. 'sensor_signal_received' est un drapeau qui indique si une impulsion du capteur de vitesse a été reçue. 'delay' est la période d'échantillonnage du contrôleur.

```

1 void speed_sensor();
2 void speed_ref_rise();
3
4 PwmOut speed(D8);
5 InterruptIn sensor(D5);

```

```

6 InterruptIn ref_gbf(D3);
7 Timer counter;
8 Timer counter_ref;
9 Timer watchdog_timer_sensor;
10 Timer watchdog_timer_ref;
11
12 volatile float sensor_period = 1.0f;
13 volatile float ref_period = 0.0f;
14 volatile float delta_t = 0.0f;
15 volatile bool sensor_signal_received = false;
16
17 float delay = 0.02;

```

Les lignes suivantes du code définissent les paramètres du contrôleur PID et du correcteur de phase. ‘kp’ et ‘ki’ sont les gains proportionnel et intégral du contrôleur PID. ‘tau’ est la constante de temps du contrôleur PID. ‘erreur_phase’ est la différence de phase entre le capteur de vitesse et la référence de vitesse. ‘K_phi’ est le gain du correcteur de phase. ‘correction_phase’ est la sortie du correcteur de phase. ‘maxcorrphi’ est la valeur maximale de la sortie du correcteur de phase. ‘puissance’ est la sortie de commande du moteur. ‘sensor_frequency’ est la fréquence du capteur de vitesse.

```

1 // PID stuff
2 float kp = 0.6f;
3 float tau = 3; //temps de r ponse en secondes
4 float ki = delay / tau;
5
6 float current_error = 0.0f;
7 float last_error = 0.0f;
8 float integral = 0.0f;
9
10 // Correcteur de phase
11 float erreur_phase = 0.0f;
12 float K_phi = 0.45f;
13 float correction_phase = 0.0f;
14
15 float maxcorrphi = 0.02;
16
17 float puissance = 0.2;
18
19 float sensor_frequency = 0.0f;

```

La fonction ‘main()’ utilise une boucle infinie pour contrôler la vitesse du moteur. Elle commence par vérifier si une impulsion du capteur de vitesse a été reçue en utilisant le drapeau ‘sensor_signal_received’. Si une impulsion a été reçue, la fonction calcule la fréquence du capteur de vitesse en utilisant la période mesurée ‘sensor_period’. Elle calcule ensuite la fréquence de la référence de vitesse en utilisant la période mesurée ‘ref_period’. Si la référence de vitesse n’a pas été reçue, la fonction utilise une valeur nulle pour la fréquence de référence. La fonction calcule ensuite l’erreur de vitesse en utilisant la fréquence de référence et la fréquence du capteur de vitesse. Elle calcule également l’erreur de phase en utilisant ‘delta_t’ et ‘ref_period’. La fonction utilise ensuite les paramètres du contrôleur PID et du correcteur de phase pour calculer la sortie de commande du moteur ‘puissance’. La fonction applique ensuite la sortie de commande au moteur en utilisant la fonction ‘speed.write()’. La

fonction attend ensuite la période d'échantillonnage ‘delay’ avant de recommencer la boucle.

Si aucune impulsion du capteur de vitesse n'a été reçue, la fonction vérifie si la minuterie de détection d'anomalie ‘watchdog_timer_sensor’ a expiré. Si c'est le cas, la fonction applique une commande de moteur fixe en utilisant la fonction ‘speed.write()’ pour éviter que le moteur ne s'arrête brutalement. La fonction attend ensuite la période d'échantillonnage ‘delay’ avant de recommencer la boucle.

```

1 int main()
2 {
3     float ref_freq = 0.0f;
4     counter.start();
5     counter_ref.start();
6     watchdog_timer_sensor.start();
7
8     sensor.rise(&speed_sensor);
9     ref_gbf.rise(&speed_ref_rise);
10
11    speed.period(0.0005f); // fr quence PWM
12    speed.write(0.0f); // Rapport cyclique
13
14    while (1)
15    {
16        if (sensor_signal_received)
17        {
18            sensor_frequency = 1.0 / (sensor_period * 1e-6); // en tr/min
19
20            if (ref_period != 0.0f)
21            {
22                ref_freq = 1.0f / (ref_period * 1e-6);
23            }
24            else ref_freq = 0;
25
26            current_error = (ref_freq - sensor_frequency) / 100;
27
28            integral += ki * current_error;
29
30            if (integral > 1) integral = 1.f;
31            if (integral < -1) integral = -1;
32
33            // Correction de phase
34            if (delta_t < ref_period / 2)
35                erreur_phase = delta_t / ref_period; // accelere
36            else
37                erreur_phase = (delta_t - ref_period) / ref_period; // ralenti
38
39            correction_phase = K_phi * erreur_phase;
40
41            if (correction_phase > maxcorrphi)
42                correction_phase = maxcorrphi;
43            else if (correction_phase < -maxcorrphi)
44                correction_phase = -maxcorrphi;
45

```

```

46     puissance = kp * current_error + integral + correction_phase;
47
48     if (puissance > 0.9) puissance = 0.9;
49     if (puissance < 0.1) puissance = 0.1;
50
51     speed.write(puissance);
52
53
54     // 2eme watchdog si on debranche le gbf
55     if (counter_ref.read_us() > 1e6)
56         speed = 0.0f;
57
58
59 }
60 else
61 {
62     if (watchdog_timer_sensor.read_us() > 1 * 1e6)
63     {
64         speed.write(0.5f);
65     }
66 }
67
68 wait_ms(delay * 1000);
69 }
70 }
```

La fonction ‘speed_sensor()‘ est le gestionnaire d’interruption pour l’entrée numérique du capteur de vitesse. Elle est appelée chaque fois qu’une impulsion du capteur de vitesse est reçue. Elle mesure la période du capteur de vitesse en utilisant la minuterie ‘counter‘ et réinitialise la minuterie. Elle met également à jour la variable ‘delta_t‘ pour la correction de phase. Elle définit le drapeau ‘sensor_signal_received‘ à ‘true‘ pour indiquer qu’une impulsion a été reçue. Enfin, elle réinitialise la minuterie de détection d’anomalie ‘watchdog_timer_sensor‘.

La fonction ‘speed_ref_rise()‘ est le gestionnaire d’interruption pour l’entrée numérique de la référence de vitesse. Elle est appelée chaque fois qu’une impulsion de la référence de vitesse est reçue. Elle mesure la période de la référence de vitesse en utilisant la minuterie ‘counter_ref‘ et réinitialise la minuterie. Elle réinitialise également la variable ‘delta_t‘ pour la correction de phase. Enfin, elle réinitialise la minuterie de détection d’anomalie ‘watchdog_timer_ref‘.

```

1 void speed_sensor()
2 {
3     sensor_period = counter.read_us();
4     counter.reset();
5
6     delta_t = counter_ref.read_us(); // correction de phase
7
8     sensor_signal_received = true;
9     watchdog_timer_sensor.reset();
10 }
11
12 void speed_ref_rise()
13 {
```

```

14     ref_period = counter_ref.read_us();
15     counter_ref.reset();
16     delta_t = 0;
17 }
```

7 Pour Conclure

Le circuit est composé de plusieurs blocs, notamment le bloc transmission et amplification du signal, le bloc moteur et le bloc détection.

Le bloc transmission et amplification du signal est composé d'un optocoupleur, d'un transistor et d'une résistance. L'optocoupleur est utilisé pour isoler le microcontrôleur du reste du circuit, ce qui permet d'éviter les interférences électromagnétiques et les surtensions qui pourraient endommager le microcontrôleur. Le transistor est utilisé pour amplifier le signal de sortie de l'optocoupleur, ce qui permet d'obtenir une tension suffisante pour commander le moteur. La résistance est utilisée pour polariser le transistor et régler le courant de base.

Le bloc moteur est composé d'un moteur à courant continu, d'une diode de roue libre et d'une capacité de lissage. La diode de roue libre est utilisée pour protéger le transistor contre les surtensions générées par le moteur lors de la commutation. La capacité de lissage est utilisée pour réduire les variations de tension d'alimentation du moteur, ce qui permet d'améliorer la stabilité de la vitesse du moteur.

Le bloc détection est composé d'un capteur optique, d'un trigger de Schmitt et d'une résistance. Le capteur optique est utilisé pour détecter la position du moteur. Le trigger de Schmitt est utilisé pour convertir le signal analogique du capteur optique en un signal numérique, ce qui permet de réduire les erreurs de mesure et d'améliorer la précision de la détection. La résistance est utilisée pour régler le courant de sortie du capteur optique.

Une fois le circuit conçu et testé sur une breadboard, nous avons routé le circuit sur une carte de développement. Pour cela, nous avons utilisé le logiciel de conception de circuits imprimés Eagle. La conception de la carte de développement a nécessité une attention particulière aux règles de conception des circuits imprimés, telles que le respect des largeurs et des espacements minimaux des pistes, l'utilisation de plans de masse et d'alimentation pour réduire les interférences électromagnétiques, et la disposition des composants pour optimiser l'espace disponible sur la carte.

Pour la partie software on utilise un régulateur PI pour ajuster la vitesse du moteur en fonction des différences de fréquence entre le capteur et la référence. Un correcteur de phase est également employé pour compenser les écarts de phase. Des interruptions sont utilisées pour détecter les signaux du capteur et de la référence, et des watchdogs sont mis en place pour assurer un fonctionnement sûr, même en l'absence de signaux.

————— *Fin* ————