



ÉCOLE UNIVERSITAIRE
DE PHYSIQUE ET D'INGÉNIERIE
Université Clermont Auvergne

Master 1 AUTOMATIQUE et ROBOTIQUE

Compte rendu de la première semaine du TER



Encadrés par :

Benoît THUILOT

Sébastien LENGAGNE

Fabrice DUMAS

Réalisée par :

Sarah AOUCHE

Année 2021/2022

Rapport 01 : Compte rendu de la première semaine de TER

Récapitulatif de la 1ère semaine :

Le travail fait pour cette 1^{ère} semaine m'a permis de découvrir deux parties de la maquette MS 15 DC Motor Module, la 1^{ère} c'était la partie de la commande analogique et la 2^{ème} c'était celle de la commande numérique PWM.

L'exploration de la maquette, de sa notice, manuel ainsi que le circuit électronique de la partie commande m'a aidé à comprendre son fonctionnement de base et de faire quelques essais en tension continue et en signal PWM.

Ce qui m'intéressait plus dans le contrôle de cette maquette c'était la commande numérique (PWM), et pour l'exploiter j'ai utilisé une carte ARDUINO UNO R3 pour générer la PWM et un oscilloscope pour visualiser les signaux.

Les petites manipulations et celle effectuées à Polytech sur la nouvelle maquette n'ont pas aboutis à des résultats satisfaisants concernant le mode de marche du moteur qui est censé tourner à vitesse maximale dans le sens inverse des aiguille d'une montre quand j'applique un signal PWM de période 22ms avec un état haut 2ms, dans le sens des aiguille de la montre quand j'applique une PWM avec un état haut de 1 ms et de se mettre à l'arrêt à 1.5 ms.

Après quelques recherches effectuées sur la maquette, l'échantillonnage, la quantification, les convertisseurs de signal, les Timer des microprocesseurs des carte ARDUINO et la fréquence des pins PWM, je suis arrivé à certaine conclusion sur les problèmes rencontrés.

Objectif à atteindre à la fin de TER :

Le contrôle numérique de la vitesse du moteur à courant continu avec la carte ARDUINO en utilisant le signal PWM.

La durée :

- ✓ Deux mois à partir de 06/05/2022 jusqu'au 4/07/2022 ;
- ✓ La soutenance en 11 au 12 /07/2022.

Tâches effectuées la 1^{ère} semaine :

Lundi 09/05/2022	1) Test en tension continue de la maquette ; 2) Etude de circuit électronique de la maquette et vérification des connexions par multimètre numérique ;
Mardi 10/05/2022	1) Test de deux programmes de la PWM et affichage sur moniteur de l'ARDUINO;
Mercredi 11/05/2022	1) Programmation de la PWM et Visualisation du signal par oscilloscope ; 2) Essai en PWM sur l'entrée (7) du commutateur (Vin / PWM) en négligeant le circuit filtre de la commande numérique PWM ; 3) Variation de la PWM ;
Jeudi 12/05/2022	1) Test de la PWM sur la nouvelle maquette à Polytech et visualisation par oscilloscope; 2) Variation de la PWM et test sur la maquette ;
Vendredi 13/05/2022	1) Vérification de la durée du signal PWM par visualisation sur oscilloscope ; 2) Modification de la fréquence de la Pin 3 de l'ARDUINO et visualisation par oscilloscope ; 3) Recherche et lecture de documents sur la maquette, ARDUINO et traitement du signal

Résultats :

- Après l'essai en tension continue $V_{in} = 5V$, j'ai constaté que le moteur tourne à sa vitesse maximale sans aucun problème.
- Le test effectué à Polytech, m'a permis de réaliser que des circuits intégrés (DG211BDJ) et (TL074CN) nécessaire à la commande PWM manquent sur la maquette que j'utilisais à l'EUPI.
- Après les tests de la commande numérique par le signal PWM, j'ai réalisé que le moteur tourne mais pas avec les exigences citées dans la notice du moteur (page 6).
- La vérification de la durée de la PWM et la lecture du manuel « Analog and Digital Motor Control Curriculum Manual CA06 » dédié à la maquette et d'autres recherches sur le traitement de signal (échantillonnage, quantification), codeur, convertisseur, processeur et Timer m'a permis d'aboutir à certaines hypothèses sur les problèmes de la PWM.



Figure 1: Visualisation du signal PWM d'une période 22ms et un état haut de 2ms par l'oscilloscope.

Hypothèses :

- L'incompatibilité du signal PWM généré par la carte ARDUINO et celui conçu pour commander le moteur de la maquette « MS15 DC MOTOR MODULE » ;
- Problèmes émanant d'une faible résolution et d'un échantillonnage à basse fréquence [1] du signal PWM généré par la carte ARDUINO UNO R3 qui entraîne une faible précision de quantification donc pertes d'information et vitesse maximale non atteignable ;
- La fréquence des Pins PWM de la carte ARDUINO ne correspond pas à la celle du signal PWM de la commande moteur ;

Défis et obstacles :

- Résoudre les problèmes de la PWM ;
- Trouver un moyen d'interface entre la carte ARDUINO et le bloc de commande PWM de la maquette ;

Solutions proposées :

La première solution qui m'ai venu en tête c'est de changer la fréquence des pins de la PWM en modifiant les registres des Timer du microprocesseur de L'ARDUINO UNO R3 car la fréquence du signal PWM est d'environ 980Hz pour les broches 5 et 6 et de 490 Hz pour les broches 3, 9, 10, 11. Mais les pré-diviseurs (diviseurs de fréquence propre à chaque Timer) donne des fréquences bien précises qui ne correspondent pas à la fréquence que je cherche ,de plus les TIMER sont soit sur 8bits ou 16 bits [4] et d'après le datasheet de la carte ARDUINO UNO R3 et celui de microprocesseur ATmega328P ainsi que le manuel «Analog and Digital Motor Control Curriculum Manual CA06 » (pages 180) :

« La technique PWM utilisée fonctionne bien à condition que le processeur puisse fournir la précision de synchronisation. De nombreux processeurs ont des registres spéciaux qui fourniront aux sorties PWM une synchronisation précise.

Pour la maquette de moteur à courant continu utilisée, un taux de mise à jour de 10 millisecondes (0,01 seconde) a été utilisé. Pour obtenir l'équivalence d'une résolution 12 bits ($2^{12} = 4096$), les temps ON et OFF doivent être contrôlés à 0,01/4000 secondes, soit 2,5 millièmes de seconde 2,5 μ s). C'est bien dans les capacités de nombreux processeurs modernes.

Des sorties PWM sont disponibles sur certains microprocesseurs et des circuits intégrés sont disponibles pour piloter des moteurs à courant continu. Ceux-ci prennent un signal logique de faible puissance et le convertissent en un entraînement PWM haute puissance vers les bobines du moteur. »

J'ai constaté que Le signal PWM de la carte Arduino a une résolution (8bits c'est-à-dire $2^8=256$ seulement), du coup, l'échantillonnage, la quantification et le codage ne sont pas les mêmes que celui destiné pour le moteur, ce qui donne une précision et une rapidité un peu loin de celle qu'on doit appliquer au moteur.

Et en consultant le site du PRODIDAC [1] de la commercialisation des équipements pédagogiques pour l'enseignement technique, j'ai trouvé dans la rubrique Exploitations pédagogiques la possibilité d'avoir des problèmes émanant d'une faible résolution et d'un échantillonnage à basse fréquence.

Afin de remédier à ça, j'ai pensé à utiliser un PCA9685 16 canaux, 12 bits PWM Servo Driver compatible avec la carte ARDUINO ou un Shield PWM/Servo 16 canaux pour Arduino.

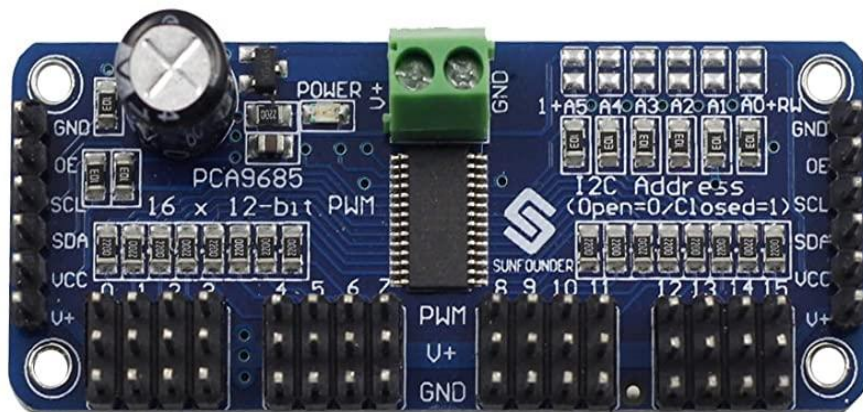


Figure 2: PCA9685 16 canaux, 12 bits PWM Servo Driver.

Caractéristiques: [7]

- 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685
- Il s'agit d'un contrôleur PWM incluant sa propre horloge et disposant d'une interface I2C.

Cela signifie que contrairement à la famille TLC5940, vous n'avez pas besoin que votre microcontrôleur produise constamment un signal (ce qui compliquerait l'écriture du programme).

Une fois la commande envoyée, les sorties PWM de ce breakout fonctionnent en totale indépendance. Votre programme peut donc se tourner librement vers l'exécution d'autres tâches!

- Le Breakout board est compatible 5V (5V compliant), ce qui signifie que vous pouvez le contrôler depuis un microcontrôleur 3.3V tout en pilotant des sorties 6V en toute sécurité (C'est bien pratique lorsque l'on veut contrôler des LEDs blanches et bleues 3.4V Forward Voltage depuis un microcontrôleur 3.3V)
- Un sélecteur d'adresse à 6 plots (6 broches) ce qui permet de raccorder jusqu'à 62 breakout board sur un seul bus I2C, soit un total de 992 sorties (ca fait une tonne de servo ou de LEDs).

- Fréquence PWM réglable (jusqu'à ~ 1.6 KHz)
- Une résolution 12-bit pour chaque sortie - pour des Servo Moteur, cela signifie une résolution de 4µs pour une fréquence de rafraichissement de 60Hz.
- Sortie configurable en push-pull ou open-drain
- Une broche "OE" (Output Enable) pour rapidement désactiver toutes les sorties.

AdaFruit a inclus le PCA9685 sur un breakout board sympa incluant quelques extras

- Bloc connecteur pour la connexion de l'alimentation (ou vous pouvez utiliser le connecteur 2.54mm disponible sur le côté du breakout board)
- Protection contre l'inversion de polarité sur le bloc connecteur.
- Une LED indiquant lorsque le breakout est alimenté.
- Des connecteurs 3 broches groupés par 4 vous permettant de connecter jusqu'à 16 servos en même temps (les connecteurs servo sont sensiblement plus larges que 2.54mm, il n'est possible que d'en brancher 4 côte-à-côte sur un connecteur à l'empatement de 2.54mm)
- Conçus pour être raccordés en "Chaîne"
- Un emplacement pour placer une grande capacité sur la ligne V+ (au cas où vous en auriez besoin)
- Des résistances 220 ohm en série sur les sorties PWM pour les protéger mais aussi pour permettre de raccorder facilement des LEDs.
- 6 plots à souder (Solder jumpers) permettant la modification de l'adresse du Breakout (l'adresse par défaut est 0x40)

Finalement quelques caractéristiques complémentaires:

- Dimensions (sans headers et bloc connecteur): 62.5mm x 25.4mm x 3mm
- Poids (sans headers et bloc connecteur): 5.5 gr
- Poids (avec headers et bloc connecteur): 9 gr

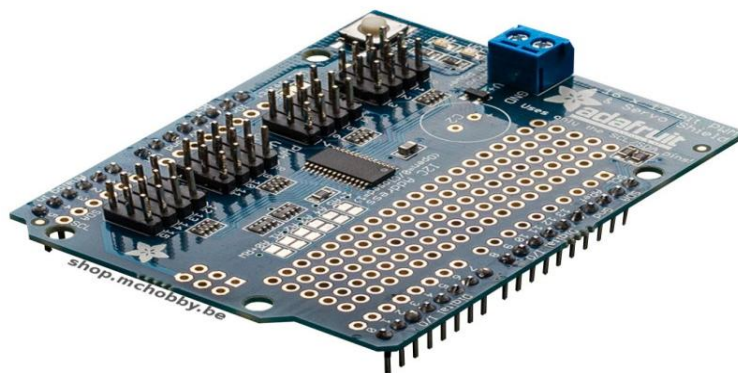


Figure 3: Shield PWM/Servo 16 canaux pour Arduino.

Caractéristiques: [8]

- Ce Shield est équipé qu'un contrôleur PWM avec un interface I2C. Le contrôleur dispose de sa propre horloge. Grâce à elle, la carte peut générer un signal PWM en toute autonomie (le microcontrôleur n'a pas besoin d'envoyer continuellement des informations au contrôleur). Le contrôleur reçoit les instructions concernant le signal PWM via le bus I2C puis se débrouille tout seul pour générer le signal en sortie.
- est compatible 5V. Vous pouvez le contrôler avec un Arduino en 3.3V et piloter des sorties jusqu'à une tension de 6 volts (pratique pour piloter des LEDs bleues et blanches qui ont une tension directe de 3.4V)
- dispose de 6 broches d'adresses. En modifiant l'adresse, vous pourrez empiler jusqu'à 62 shield (soit 992 sorties PWM, cela fait beaucoup de servo et de LEDs)
- dispose d'une fréquence PWM ajustable (jusqu'à ~1.6 KHz)
- Dispose d'une sortie avec une résolution 12-bits (0 à 4096). Pour les servos, cela signifie une résolution de ~4µs à un débit de 60 Hz (update rate = 60Hz)
- est configurable en sortie push-pull ou sortie à drain ouvert

Le PCA9685 n'est pas le seul élément intéressant du shield PWM / Servo d'Adafruit. Ce shield dispose également :

- D'un bornier pour l'alimentation de puissance -ou- une entrée breakout à l'empattement 2.54mm
- Protection contre la polarisation inversée (sur le bornier de puissance)
- Des leds d'alimentation verte et rouge (rouge indiquant une alimentation de puissance)
- Des groupes de 3 connexions pour raccorder plus facilement les servo-moteurs (jusqu'à 16 servos en une fois).
- Conçu pour être empiler. Si vous empilez les shields, il faudra cependant opter pour des connecteurs 3 broches coudées (à 90°) pour pouvoir brancher des servos sur le shield PWM.
- Un point de connexion pour ajouter une grosse capacité de régulation sur la ligne V+.
- Cela permettra de réguler les pointes d'appel de courant.
- Les sorties sont protégées par une résistance de 220 Ohms montée en série sur la sortie. Il est donc possible de contrôler des LEDs directement avec le contrôleur.
- 6 broches d'adresse (avec des cavaliers à souder)
- Une zone de prototypage de 5x20 trous pour tous les composants supplémentaires que vous voudriez ajouter.

➤ Ils peuvent même être utilisés pour commander des moteurs à courant continu, mais la chose dont je ne suis pas très sûre c'est la résolution de 4µs (0.016/4000 secondes) qui est supérieur à celle du signal du moteur 2.5 µs(0.01/4000 secondes), si elle donnera la PWM adéquate ou non ! ça reste à tester.

Références :

[1]: <https://prodidac.fr/R%E9gulation%20automatisme%20%E9lectrotechnique/R%E9gulation/Banc%20d%27%E9tude%20de%20r%E9gulation%20analogique%20et%20num%E9rique%20d%27un%20moteur%20%E0%20courant%20continu-207-00>

[2] : <https://github.com/terryjmyers/PWM>

[3]: <https://code.google.com/archive/p/arduino-pwm-frequency-library/downloads>

[4] : <https://passionelectronique.fr/introduction-timer-arduino/#les-differents-registres-utiles-et-lies-aux-timer-arduino>

[5]: Analog and Digital Motor Control Curriculum Manual CA06

[6]: <http://culturesciencesphysique.ens-lyon.fr/ressource/principe-numerisation.xml>

[7]: <https://shop.mchobby.be/fr/cartes-breakout/89-adafruit-controleur-pwm-servo-16-canaux-12-bits-i2c-interface-pca9685-3232100000896-adafruit.html>

[8]: <https://shop.mchobby.be/fr/shields-arduino/1160-shield-pwmservo-16-canaux-pour-arduino-3232100011601-adafruit.html>