

Contrôle de moteurs électriques avec Arduino

GNG1503 – Génie de la Conception

Objectif

Ce laboratoire donne une vue d'ensemble des moteurs électriques utilisés dans la plupart des projets de conception des cours GNG 1503 et 2501. En mettant l'accent sur le meilleur rapport qualité-prix et la simplicité d'utilisation le contrôle à l'aide d'une carte Arduino des moteurs à courant continu (*DC motors*), les moteurs pas-à-pas (*stepper motors*) et les servomoteurs (*servo-motors*) sera abordé. A la fin de ce laboratoire, vous serez en mesure de :

- Régler la direction de rotation du moteur
- Augmenter ou diminuer la vitesse de rotation du moteur
- Déterminer la distance parcourue en fonction du nombre de rotation
- Identifier les scénarii où l'utilisation d'un contrôleur/pilote de moteur (motor controller/driver) est nécessaire.

Contexte

Un moteur électrique est une machine qui permet de transformer l'énergie électrique en énergie mécanique. Il y a une variété de moteurs électriques disponibles sur le marché actuellement. Les *moteurs à courant continu* sont des plus populaires de la catégorie des moteurs électriques. Traditionnellement, ils sont constitués de l'inducteur (stator), de l'induit (rotor), du collecteur et des balais. La vitesse d'un moteur à courant continu peut-être contrôlée en variant la tension à ses bornes. Les deux types de moteurs CC les plus courants sont : les moteurs à balais (*Brushed*) et les moteurs sans balais (*brushless*). Les moteurs à courant continu sont largement adoptés pour les applications qui requièrent une RPM (rotation par minute) élevée. Par exemple ; les roues mobiles, ventilateurs, etc.

Le *servomoteur* est une sous-catégorie des moteurs à courant continu sans balais. Il est composé typiquement d'un moteur à courant continu, d'un axe de rotation (variation standard entre 0 et 180°), d'un capteur de position d'angle d'orientation de l'axe (très souvent un potentiomètre) et une carte électronique pour le contrôle de la position de l'axe et le pilotage du moteur à courant continu. Un servomoteur est capable d'atteindre des positions prédéterminées dans les instructions qui lui ont été données, puis de les maintenir. Toutefois, il faut souligner que ce qui rend un moteur *servo* ou non n'est pas le type de moteur, mais plutôt la boucle de rétroaction. Les servomoteurs sont idéaux pour les applications qui nécessitent un couple élevé et de la précision telles que : les bras robotiques et le contrôle du gouvernail d'un bateau. Ils sont considérés comme une alternative de haute performance au moteur pas-à-pas.

Un *moteur pas-à-pas* est essentiellement un servomoteur qui utilise une technique de motorisation différente. Dans ces types de moteur, on change la position du rotor en modifiant la direction d'un champ magnétique créée par les bobinages de l'inducteur. Ils ont généralement un grand nombre de paires de pôles magnétiques (50 à 100) générés soit par des aimants permanents ou le courant électrique. Ils nécessitent un circuit de commande qui comporte une partie logique et un circuit de puissance. La partie logique détermine pour chaque *pas* quelles sont les bobines alimentées et le sens de rotation. La partie de puissance, contrôle la vitesse du moteur. Les avantages de cette technologie sont une longue durée de vie, peu d'entretien et un haut rendement (85-90 %). Les Moteurs pas-à-pas principalement sont utilisés dans le contrôle de position en boucle ouverte, avec des utilisations allant des imprimantes 3D, des ventilateurs, pompes et compresseurs, jusqu'à des applications industrielles telles que les équipements de placement à grande vitesse.

Dans le cadre de ce laboratoire plusieurs expérimentations seront menées considérant chacun de ces moteurs couplés d'un microcontrôleur Arduino. Généralement, la tension et le courant à la sortie de la carte Arduino sont trop faibles par rapport aux caractéristiques des moteurs. Donc, il n'est pas possible de commander un moteur directement sur une sortie d'une carte Arduino. Par conséquent il sera impératif d'incorporer une interface de puissance pour assurer la liaison entre le microcontrôleur et une source d'alimentation externe. Ce composant pourrait être par exemple un transistor (bipolaire, MOSFET), un système de relais, un pilote de moteur (*motor driver*) ou un contrôleur de moteur. Le principe de fonctionnement de quelques-uns est expliqué dans les lignes qui suivent.

Pour qu'un moteur électrique puisse être opérationnel, il lui faut un *pilote de moteur* prévu à cet effet. Ce genre de pilote est souvent composé d'un pont en H, d'un circuit de régulation de tension et de quelques connecteurs pour relier les moteurs et une batterie. Un pilote de moteur peut être vu comme un genre d'amplificateur qui aide le microcontrôleur à commander le moteur en question.

En plus de remplir les mêmes fonctions qu'un pilote de moteur, un *contrôleur de moteur* a la capacité de communiquer les erreurs détectées, son statut et autres informations intéressantes. Il utilise plusieurs méthodes de communications standard telle que l'UART (également appelée port série) ou la MLI (Modulation de Largeur d'Impulsion)/PWL (Pulse Width Modulation) pour interagir avec le microcontrôleur. En outre, certains contrôleurs de moteur peuvent être commandés manuellement par une tension analogique (généralement créée au moyen d'un potentiomètre).

Enfin, un autre dispositif qui est très utilisé et propre aux cartes Arduino est l'interface moteur Arduino (motor shield) qui repose sur le double pilote (driver) de pont entier L293D. Ce shield contrôle la vitesse et la direction de façon indépendante. Il permet de connecter un ou plusieurs moteurs (4 moteurs CC, 2 moteurs pas à pas, ou 2 servomoteur à la fois) directement sur la carte Arduino sans avoir recours à des montages ou des circuits supplémentaires, comme c'est normalement le cas lors de l'utilisation d'un module Arduino.

Aperçu des Appareils et Équipement

L'équipement qui va être utilisé pour ce laboratoire inclus :

- 1 x Microcontrôleur Arduino Uno R3
- 1 x Câble USB 2.0 Type A à USB 2.0 Type B
- 1 x Ordinateur de Lab. ou Personnel (avec Windows, Macintosh, ou Linux OS)
- 1 x Shield Moteur Adafruit V1 (L293D)
- 1 x plaque d'essai
- 1 résistance de 220 Ohms
- 1 x potentiomètre
- 1 x Moteur DC dans un corps de Micro Servo
- 1 x MOSFET Canal-N
- 1 x servomoteur
- 1 x moteur pas-à-pas
- Des fils connecteurs
- Logiciel Arduino 1.8.2 IDE
- Bibliothèques correspondantes Arduino IDE (expliqué plus bas)

Préparation pré-lab

Avant d'arriver dans le laboratoire, vous devez lire ce manuel et vous familiariser avec les thèmes et les procédures de ce laboratoire. L'utilisation de votre ordinateur personnel est autorisé, à condition que vous ayez le logiciel Arduino IDE ainsi que les bibliothèques requises. La révision du syntaxe d'Arduino IDE ainsi qu'essayer d'écrire certains programmes à l'avance est encouragé. Une liste utile de commandes utilisées dans Arduino peut être trouvée ici :

<https://www.arduino.cc/en/Reference/HomePage>.

Questions pré-lab

Quels sont les deux grandes familles de moteurs à courant continu ?

Que représente l'induit (rotor) d'un moteur CC?

Pourquoi doit-on utiliser une interface de puissance?

Nommez quelques exemples d'interfaces de puissances entre un moteur et un microcontrôleur?

Quelle est la différence entre un *pilote de moteur* et un *contrôleur de moteur* ?

Avertissements

Des composantes électriques sensibles comme des plaques de contrôle et actionneurs utilisés dans ce laboratoire peuvent être endommagés ou brisés par une décharge électrostatique provenant d'une mauvaise manipulation pendant assemblage ou usage. Avant de préparer ce lab. Les étudiants devraient s'assurer que leur espace de travail est préparé de manière électrostatique (espace de travail non-métallique, pas de tapis/autres matériaux fibreux) et ils devraient se 'mettre à terre' en touchant un objet en métal large avant de manipuler les composantes. Il y a des accessoires pour améliorer l'espace de travail d'assemblage les décharges électrostatiques, comme un tapis antistatique et des bandes statiques de poignet/cheville, qui devraient être utilisés si disponibles. Avant de commencer un montage rassurez-vous que la carte Arduino n'est pas branchée à une source d'alimentation (murale ou ordinateur).

Partie A - Contrôle de moteur CC à l'aide d'un MOSFET

Cette section vous permettra de contrôler un moteur CC à l'aide d'un transistor MOSFET en utilisant une séquence de commandes prédéfinie. Les MOSFETs sont une catégorie de transistor qui est très utilisée dans les circuits analogues et numériques. Les transistors MOSFET sont des amplificateurs en tension qui aident à garder stable la tension aux bornes d'un moteur au moment du démarrage.

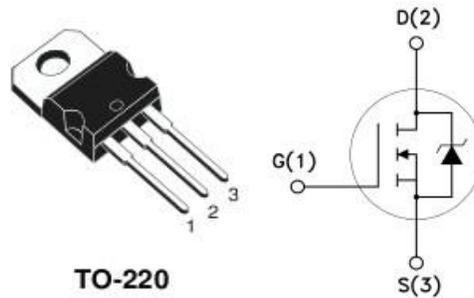


Figure 1: MOSFET Canal-N

- G ou 1 représente la borne GRILLE (GATE) : broche de commande
- D ou 2 représente la borne DRAIN : pour un canal-N on la branche à la charge (le moteur)
- S ou 3 représente la borne SOURCE : pour un canal-N on la connecte à la masse (*ground*).

Figure 2 montre le schéma de montage d'un MOSFET Canal-N avec un moteur et une carte Arduino. Parfois l'ordinateur n'est pas capable de donner assez de puissance aux moteurs. Dans ces cas-là on peut alimenter le moteur via la broche *Vin* de la carte Arduino lorsqu'elle est alimenté par la prise murale ou une source de tension d'alimentation séparée. Pour la protection du système, une *diode Schottky* peut être utilisée pour empêcher un retour de courant dû à l'inertie du moteur qui s'arrête.

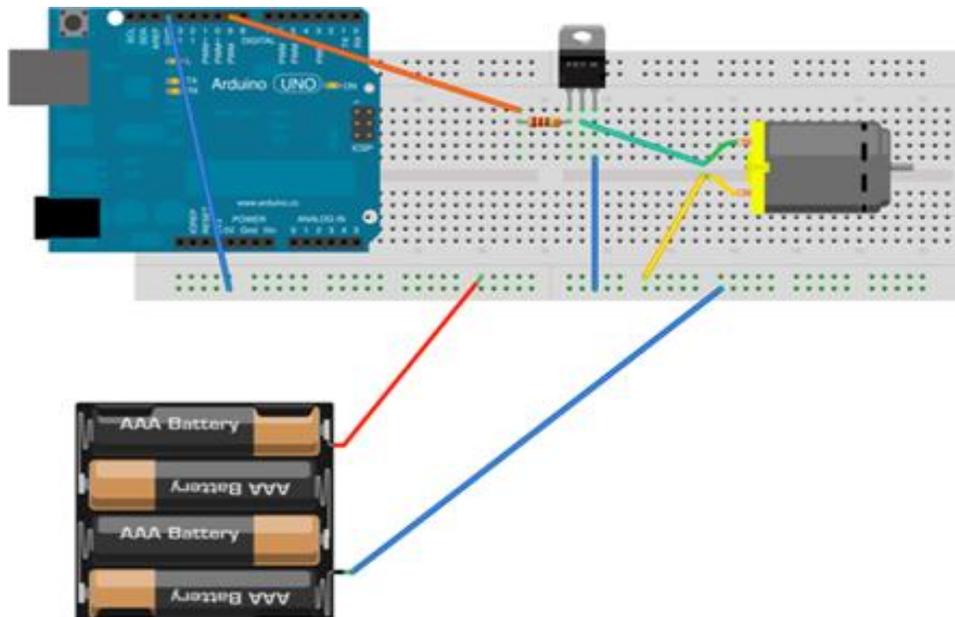


Figure 2: Plan de montage d'un moteur CC avec un MOSFET

Au-delà des commandes de base telles que "avancer et reculer", certaines applications pourraient nécessiter une variation de vitesse. Pour ce faire, on utilisera généralement certaines

broches de sorties digitales de l'Arduino. Elles sont notées PWM et on les reconnaît grâce au symbole " ~ " à côté de leur numéro. Ces broches offrent la possibilité d'envoyer un train d'impulsion variable entre 0V et du +5V. Dans ce montage, la broche *GATE* est montée en série avec une résistance de **220 Ohms** qui protège l'Arduino des courants élevés. Ensuite, le moteur est connecté à une source de courant externe et au *Drain* du MOSFET. Le montage est complété en connectant les mises à la terre. Aussi, le MOSFET communique avec l'Arduino à travers la broche **9**.

```
Lab_moteur_CC_MOSFET$
int pinMoteur=9; //pin de commande relié au MOSFET

void setup(){
  pinMode(pinMoteur,OUTPUT); // pin de commande en mode sortie
}

void loop(){

//===== Partie 1 =====

  for (int pourcentage=0;pourcentage<=100;pourcentage+=25){//boucle de variation en %
    int valeur=map(pourcentage,0,100,0,255); //conversion en valeur
    analogWrite(pinMoteur,valeur); // pin de commande en mode impulsion
    delay (2000); // attente de 2 secondes
  }

//===== Partie 2 =====

  digitalWrite(pinMoteur,HIGH); //le moteur se lance
  delay(1000);

//===== Partie 3 =====

  for (int *** ; *** ;pourcentage***25){//boucle de variation en %
    int valeur=map(pourcentage,0,100,0,255); //conversion en valeur
    analogWrite(pinMoteur,valeur); // pin de commande en mode impulsion
    delay (***) ; // ajouter une attente de 0.5 seconde
  }
  digitalWrite(pinMoteur,LOW); // le moteur s'arrête
  delay(1000);
}
```

Figure 3: Code Arduino pour le moteur CC

Décrivez le rôle de chaque partie du code de la figure précédente.

1. Montez le circuit comme l'indique la Figure 2.
2. Considérant qu'on veut réduire la vitesse graduellement, écrivez le code affiché dans la Figure 3 et remplacez les *** par les valeurs appropriées.
3. Montrez votre travail au TA/PM.

Si le travail est correct, compilez, téléchargez le code et notez vos observations.

Expliquez en vos propres termes la signification de l'expression "pourcentage+=25"

Partie B - Contrôle d'un servomoteur

Cette partie du laboratoire vous servira de guide pour comprendre le contrôle des servomoteurs. Nous l'avons vu plus haut, la connectique d'un servomoteur se résume à trois fils : deux pour l'alimentation positive et la masse et le dernier pour le signal de commande. Rappelons qu'un servomoteur accepte généralement une plage d'alimentation comprise entre 4.5V et 6V. Dans ce cas, n'oubliez pas de relier la masse de l'alimentation externe et celle de l'Arduino afin de garder un référentiel électrique commun. Étant donné que les servomoteurs ont déjà un système de contrôle embarqué, une interface de puissance n'est généralement plus nécessaire. Dans cet exemple, on va connecter la puissance et la masse directement aux broches 5V et GND de l'Arduino. Le signal PWM va être connecté à une broche de sortie digitale de l'Arduino.

Attention : Ne faites pas tourner manuellement le rotor du servomoteur pour ne pas l'endommager.

1. Connectez le servomoteur et le potentiomètre à la carte Arduino en utilisant le diagramme présenté à la Figure 4.
2. Le code à exécuter se trouve : File > Exemples > Servo > Knob.
3. Une fois terminé, montrez votre travail au TA/PM.

Quel est le nom de la librairie utilisée pour cette partie du laboratoire ?

Nommez toutes les variables déclarées dans ce code ?

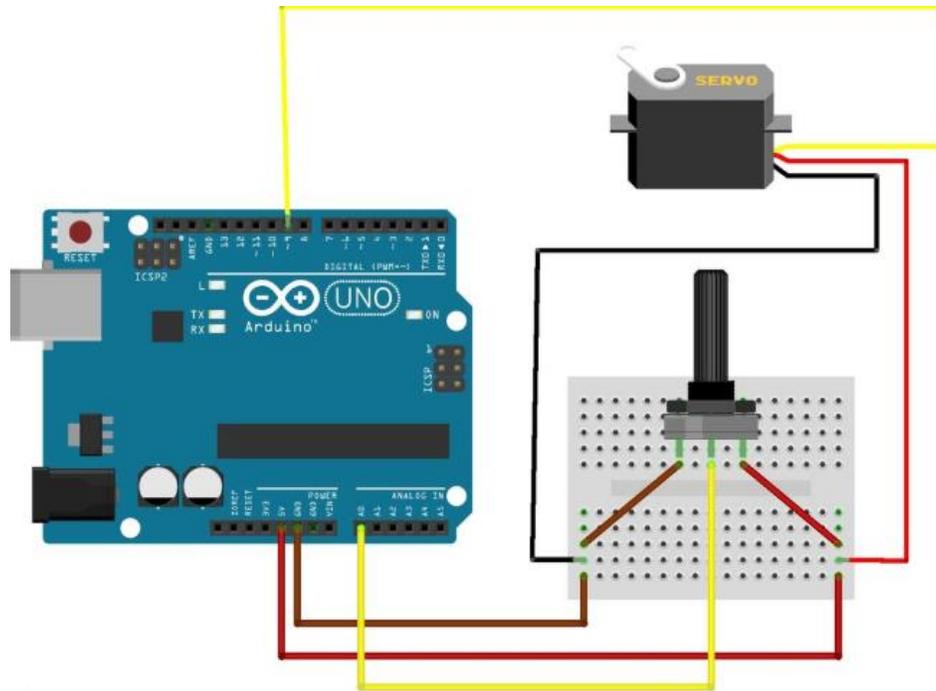


Figure 4: Montage servomoteur avec un potentiomètre

Partie C - Contrôle de moteur pas-à-pas à l'aide d'un shield de moteur

Ce programme va tester et contrôler deux moteurs pas-à-pas en utilisant une séquence de commandes prédéfinie à l'aide d'un shield de moteur. Les étapes du code sont les suivantes :

1. Installer la librairie *AFMotor.h*, <https://github.com/adafruit/Adafruit-Motor-Shield-library>
2. Déclarer un objet *AF_Stepper(steps, stepper#)* pour configurer le H-bridge. *Steps* indique le nombre de pas du moteur par révolution (un moteur de 7,5 degrés/pas a $360/7,5 = 48$ pas). *Stepper#* est le port sur lequel il est connecté. Si vous utilisez M1 et M2, c'est port 1. Si vous utilisez M3 et M4, c'est port 2.
3. Fixer la vitesse via la fonction *setSpeed(rpm)* avec l'argument *rpm* le nombre de tours par minutes voulu.
4. Utiliser la fonction *step(#steps, direction, steptype)* pour faire tourner le moteur. Le nombre de pas à exécuter est *#steps*. La *direction* est soit *AVANT (FORWARD)* ou *ARRIÈRE (BACKWARD)* et la précision, *steptype*, le type de pas est *SINGLE*, *DOUBLE*, *INTERLEAVE* ou *MICROSTEP*. "Simple" signifie activation d'une seule bobine, "double" signifie que 2 bobines sont activées en même temps (pour un couple plus élevé) et "interleave" signifie qu'il alterne entre simple et double pour obtenir une résolution double (mais bien sûr, sa vitesse est de moitié). "Microstep" est une méthode dans laquelle les bobines sont en PWM pour créer un mouvement fluide entre les étapes.

```

Stepper_motor_Code_Fr
#include <AFMotor.h> // Ajout de la librairie AFMotor

AF_Stepper motor(***, 1);

void setup() {
  Serial.begin(***); // Configuration du port série à 9600 bps

  motor.setSpeed(***); // fixer le pas à 60 rpm (tour par minute)

  motor.step(100, FORWARD, SINGLE);
  motor.release(); // Libère l'axe après un déplacement
  delay(1000);
}

```

5. Pour compléter cette section du laboratoire, veuillez télécharger le document « **Stepper_motor_Code_Fr.ino** » sur le campus virtuel et remplace les ******* par les valeurs appropriées basé sur ce que vous avez appris dans les sections précédentes.
6. Ensuite, compilez et téléchargez le code sur la carte Arduino et observez ce qui se passe lorsque la précision du pas change. Montrez le travail au TA/PM.

Il faut souligner aussi que la fonction *step()* est dite «bloquante», cela signifie que le code après *step()* ne sera exécuté qu'une fois le déplacement de *n pas* est effectué.

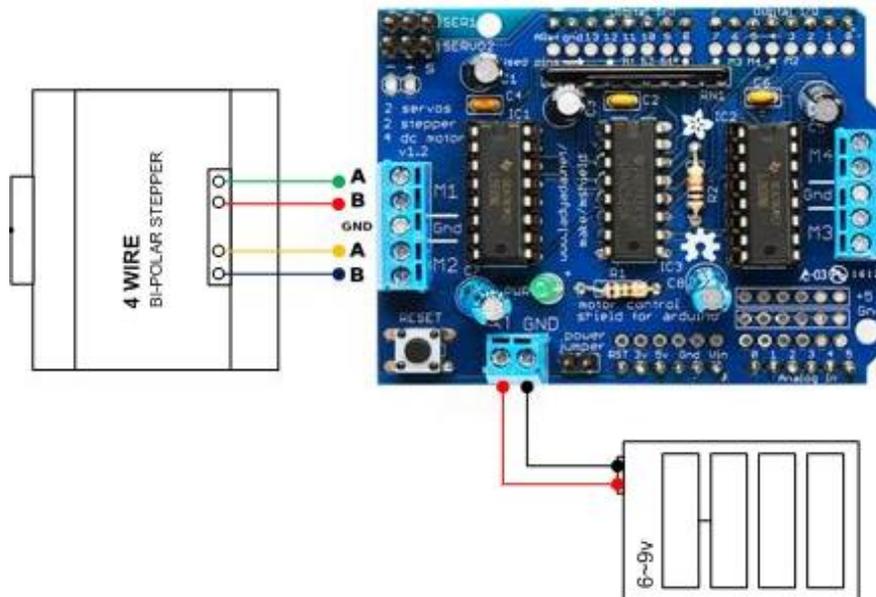


Figure 5: Montage avec stepper et un shield moteur Adafruit V1

Ressources additionnelles

- [Choisir un contrôleur de moteur](#)
- [Apprendre à contrôler un moteur pas à pas avec le shield moteur Arduino](#)
- [Manuel de Laboratoire de Programmation Arduino - GNG2501](#)
- [Using Stepper Motors](#)