Comment contrôler un moteur pas à pas

Par Kilian Sanfins

Introduction

Comme son nom l'indique, ce moteur avance de pas en pas, lui conférant une grande précision de mouvement et, de par sa construction, un couple à basse vitesse et de maintien important. Grâce au développement des imprimantes 3D, le moteur pas à pas et en particulier son contrôleur a vu son coût baisser et s'est démocratisé, au point où des bibliothèques lui sont dédiés, rendant le contrôle de ce moteur très aisé avec un arduino.
On utilisera ici le A4988 de Allegro, il existe également un équivalent de Texas Instrument le DRV8825.

Avantage

- Bon rapport couple/précision/coût
- Très facile d'utilisation si l'on souhaite contrôler un moteur en position.
- Couple de maintien, le moteur ne perds pas sa position, tant qu'on l'alimente
- Précis si dimensionné correctement

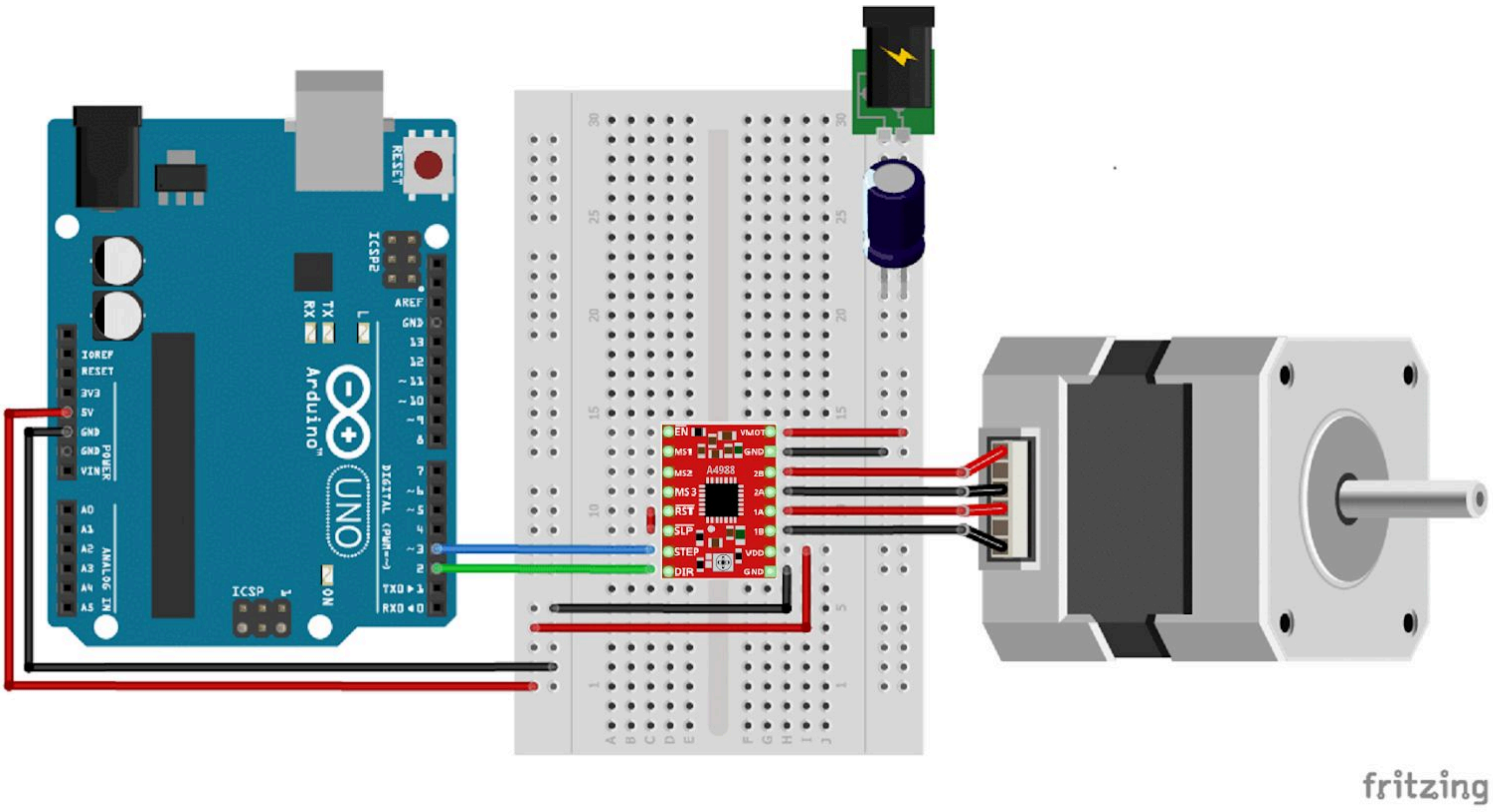
Désavantage

- A haute vitesse le couple diminue
- Le moteur pas à pas à tendance à être plus volumineux que d'autres technologies de moteur
- On le contrôle sous forme de pas, par incrément, il est possible de lisser le mouvement en le subdivisant en "micropas", mais en contrepartie, le couple fourni entre 2 micropas est réduit.
- Le moteur fonctionne en boucle ouverte, ainsi s'il est mal dimensionné et qu'il "saute un pas" (couple à fournir trop important) il est impossible de le détecter à moins d'utiliser un encodeur.
- Pour maintenir la position, le moteur doit être alimenté et donc consommer du courant.

Cas d'utilisation

- Lorsqu'il est nécéssaire d'obtenir un mouvement coupleux à basse vitesse tout en restant précis (peu de jeu)
- Si l'on souhaite contrôler avec précision la position du moteur

Schéma électrique



Code

```
/*
   Ce programme a pour but de contrôler un moteur pas à pas
   avec le driver A4988 de Allegro.
   la pin 3 est relié à STEP
   la pin 2 est relié à DIR
   On alimente la carte avec une tension maximum de 35V
   Attention à ce que le moteur supporte la tension, mais
   généralement un moteur pas à pas peut supporter une forte
   tension, à partir du moment où l'on ne dépasse pas le courant
   maximal autorisé (qui est de 2A pour le driver A4988).

   Code écrit par Kilian Sanfins
*/


#define STEP 3 // On renomme les pins de manière
#define DIR 2 // plus lisible

void setup() {
  pinMode(STEP, OUTPUT); // On initialise les pins comme
  pinMode(DIR, OUTPUT); // des sorties

  digitalWrite(STEP, LOW);  // On s'assure que l'état des
  digitalWrite(DIR, LOW); // pins est fixé et pas flottant
}



void loop() {

  Direction(1); // On sélectionne le sens horaire
  for (int i = 0; i > 100; i++) { // Et on fait avancer le moteur de 100 pas
       Step();
  }

  delay(1000); // On attend 1 sec

  Direction(0); // On tourne à présent dans l'autre sens (anti-horaire)
  for (int i = 0; i > 100; i++) { // On avance de 100 pas
       Step();
  }

  delay(1000);

  // Nous voilà à nouveau au point de départ!
}
```

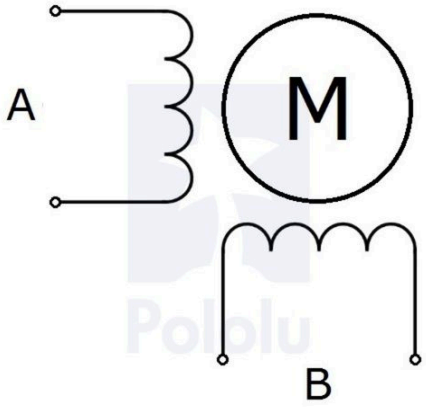
```
/*
   Fais avancer le moteur de 1 pas
*/
void Step() {
  digitalWrite(STEP, HIGH);
  delayMicroseconds(2); // Un délai de 1 microsecondes est requis par le driver
  digitalWrite(STEP, LOW);

  // Ce délai est là pour s'assurer que pour 2 appels consécutifs de Step,
  // le délai entre les 2 appels ne soit pas trop court pour le driver
  delayMicroseconds(2);
}
```

```
/*
   Envoie un signal au driver sur la pin DIR
   pour définir le sens de rotation.
   Si sens = 1 alors le moteur tourne dans le sens horaire.
   Si sens = 0 alors le moteur tourne dans le sens antihoraire.
   (Si ce n'est pas le cas, inverser le branchement du moteur)
*/
void Direction(bool sens) {
  if (sens == 1) {
       digitalWrite(DIR, HIGH);
  }
  else {
       digitalWrite(DIR, LOW);
  }
}
```

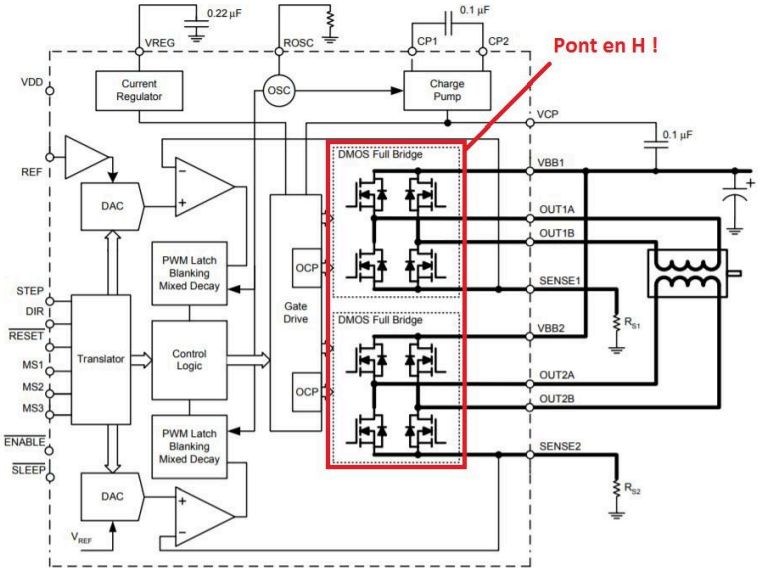
Explications

Le moteur pas à pas le plus répandu est souvent constitué de 2 paires de bobines que l'on nomme traditionnellement A et B.



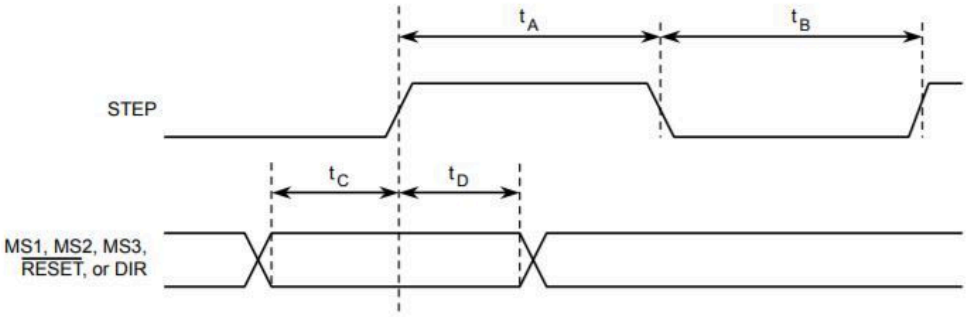
En fonction du sens du courant que l'on fait circuler au travers de ces bobines, il est possible de faire tourner le moteur dans un sens ou dans l'autre, de pas en pas.
Il s'agit donc de contrôler correctement la tension appliquée aux bornes de la bobine pour choisir dans quel sens circulera le courant (mais également son intensité, mais on y reviendra plus tard).
Pour cela, il y a en réalité dans le driver 2 pont en H (voir tutoriel sur le moteur à courant continu), un pour chaque paire de bobines, comme on peut le voir ci-dessous avec l'extrait de la datasheet du driver A4988:

Functional Block Diagram



La carte se charge alors de faire circuler le courant dans le bon sens, à chaque activation de la pin STEP, en fonction de l'état de la pin DIR.

Le driver requiert cependant une séquence particulière pour réaliser un pas. A nouveau, la datasheet nous fournit la réponse, à la page 6 cette fois ci:



Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t_A	1	µs
STEP minimum, LOW pulse width	t_B	1	µs
Setup time, input change to STEP	t_C	200	ns
Hold time, input change to STEP	t_D	200	ns

Figure 1: Logic Interface Timing Diagram

Ce tableau nous explique comment envoyer les signaux au driver pour qu'il exécute correctement les commandes, en particulier les délais maximums et minimums pour changer d'état les différentes pins.
Tout d'abord la ligne **STEP**:

- Initialement à l'état bas
- doit rester un minimum de 1 µs à l'état haut (délai t_A)
- puis doit repasser à l'état bas, et ce pendant au minimum 1 µs (délai t_B) avant de pouvoir relancer une seconde commande.

La fonction step() du code exécute cette séquence à chaque fois que l'on souhaite avancer d'un pas.

Pour la seconde ligne, cela se lit de la même manière et la consigne est valable pour les 5 pins DIR, RESET (la barre au-dessus d'un mot indique que le signal sera inversé), MS1, MS2 et MS3 (plus d'explication sur ces 3 pins juste après!):

- Initialement à l'état bas OU haut
- le délai t_C de 200 ns est le délai avant l'activation de STEP pendant lequel STEP prendra en compte l'état (mise en place de l'état)
- le délai t_D de 200 ns est le délai après l'activation de STEP pendant lequel STEP prendra en compte l'état (maintien de l'état)

Pour résumer, 200 ns avant et après avoir activé STEP, cette valeur ne doit pas être vhangée auquel cas le signal sera mal interprété!
Un arduino fonctionne à 16 Mhz (soit 16 million de cycles à la seconde ou ~62.5 ns/cycle), avec des instructions qui prennent 1 à 3 cycles en moyenne. Des appels de fonction comme digitalRead ou digitalWrite prennent ~60 cycles en comparaison.
A moins de changer constamment la direction du moteur avec un appel d'un pas dans la foulée (et encore…) il n'y a pas forcément de raison à placer un délai juste après le changement des pins.

Mais sachez que ce délai de changement existe pour de nombreuses entrées et sorties et qu'il peut poser des soucis de stabilité si on ne l'a pas pris en compte et que l'on pousse le composant à ses capacités maximales!

Micro-Pas

La technique du micro-pas consiste à subdiviser chaque pas en micro-pas. Ainsi, si l'on a configuré le driver à faire un "demi-pas" à chaque appel sur la pin STEP, le moteur n'avancera plus d'un pas mais d'une moitié de pas, et il nécessitera un second appel sur la pin STEP pour faire un pas complet.

Le A4988 peut réaliser jusqu'à 1/16ème de pas, il faudra donc faire 16 appels sur la pin STEP pour faire avancer le moteur de 1 pas complet.
On avancera alors de 1/16ème de pas à chaque appel sur la pin STEP, pour un moteur pas à pas typique de 200 pas/révolution, chaque pas correspond à une rotation de 1.8°, et chaque micro-pas (toujours en 1/16ème de pas) correspond à une rotation de 0.1125°!
Outre le gain en précision, le micro pas peut aussi lisser le mouvement à basse vitesse (à haute vitesse il est préférable de privilégier un fonctionnement en FULL STEP) et le rendre donc moins "rugueux" (on ressent moins les pas)

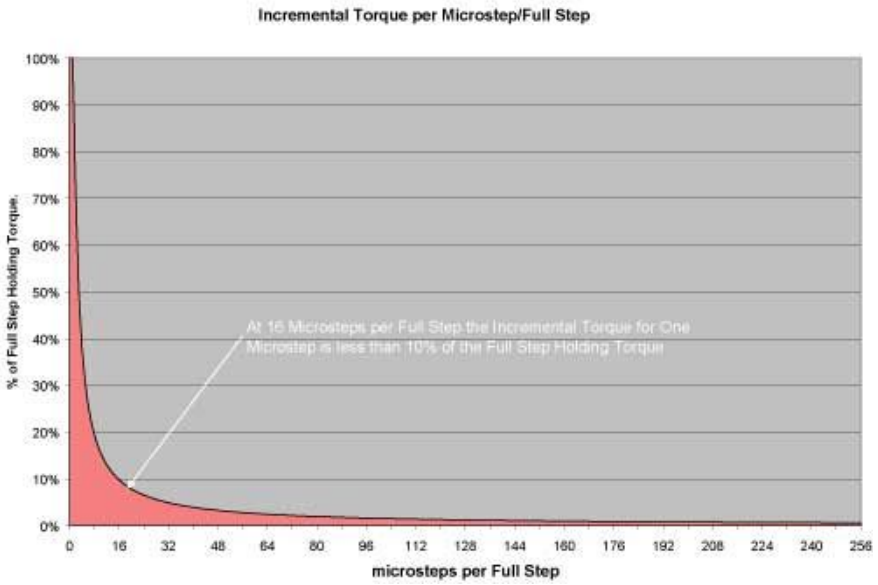
Comment activer ce mode de fonctionnement? Un petit tour page 6 de la datasheet nous fournit ce tableau:

Table 1: Microstepping Resolution Truth Table

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Il suffit de placer MS1, MS2, MS3 à l'état haut (5v) pour obtenir le fameux mode en 1/16ème de pas. Si l'on ne les connecte à rien, le driver fonctionnera en mode FULL-STEP (1 step = 1 pas complet).

Bien sûr il y a un revers à la médaille, le couple **incrémental** diminue drastiquement avec la division du pas.



Ceci signifie que si le couple résistant (l'effort au bout de l'arbre moteur) est trop grand, alors le moteur ne bougera pas! On dit que le moteur "perd des pas" car on lui demande d'avancer d'un pas et ne le fait pas. Ceci est détrimental pour la précision du mouvement à réaliser, on pense tourner 1 tour, alors que en réalité on n'a fait que 95% du tour (ou pire..). Au bout du compte, le système peut être complètement érroné!

De manière générale c'est le défaut d'un système que l'on dit en boucle ouverte, il n'y a pas de retour de l'information. Si une erreur apparaît, le système n'a aucun moyen de l'indiquer

C'est pourquoi il est très important de bien dimensionner son moteur pas à pas en fonction de son utilité, notamment réaliser une application avec du couple à la limite des capacités du moteurs, avec un positionnement précis.

Astuce!

Pour trouver quels fils du moteur vont de pair (pour constituer les bobines), il suffit de s'équiper d'un multimètre, de le mettre en mode "test de continuité" et de chercher les 2 fils qui vont ensemble!

Sources pour approfondir

- HowToMechatronics.com, site en anglais, avec une vidéo explicative.
  https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/

- Non l'arduino n'est pas lent! (anglais)
  https://cybergibbons.com/uncategorized/arduino-misconceptions-2-arduino-is-slow/

- PDF à télécharger en bas de la page, détail du couple en microstepping (anglais)
  https://www.micromo.com/technical-library/stepper-motor-tutorials/microstepping-myths-and-realities

- La datasheet du A4988. (anglais)
  https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf

- La datasheet du DRV8825. (anglais)
  http://www.ti.com/lit/ds/symlink/drv8825.pdf