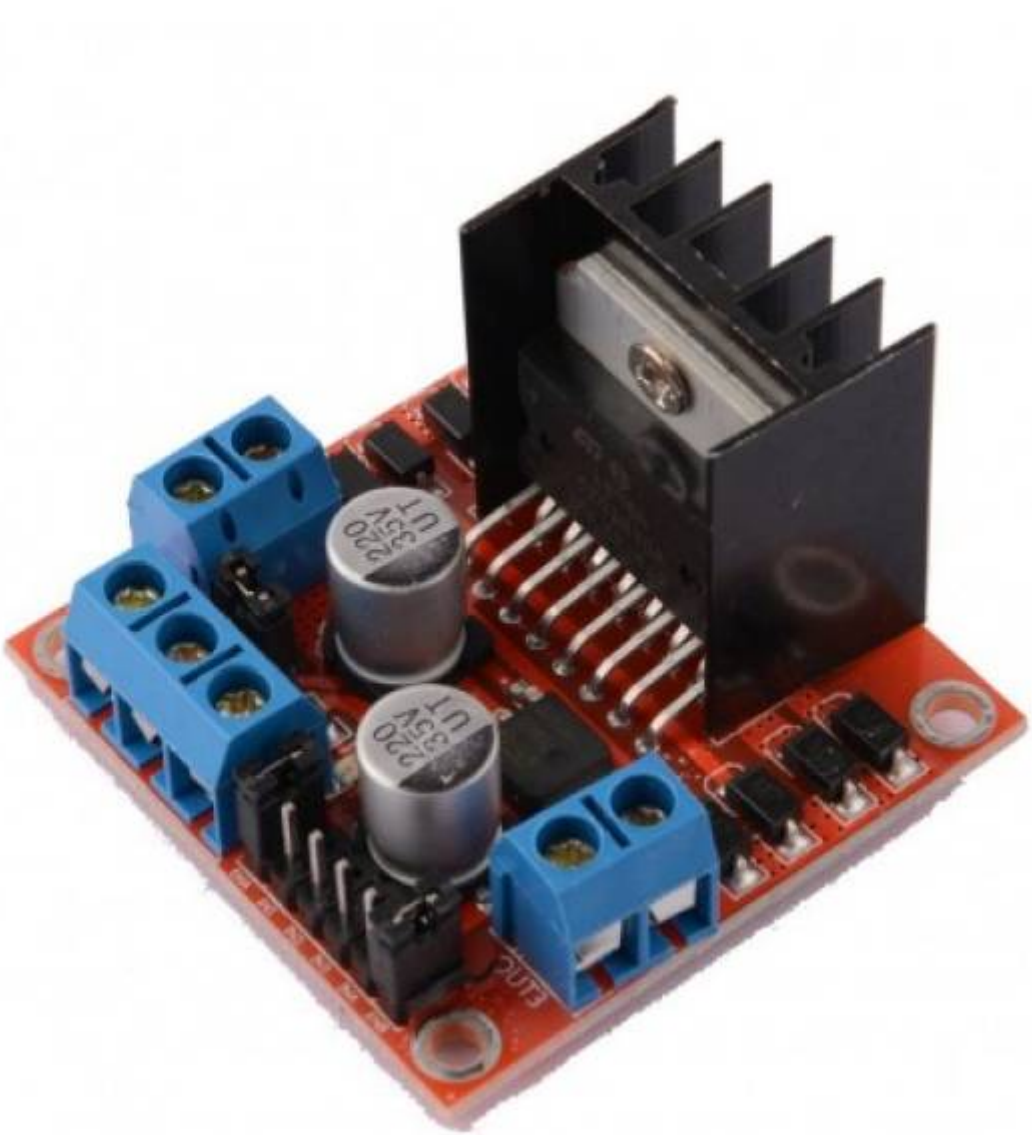
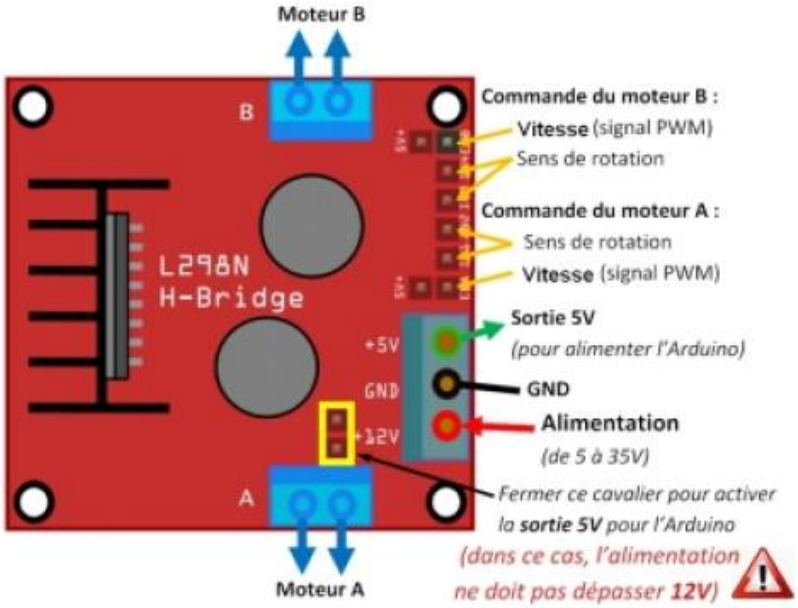
Contrile de Moteur pas à pas

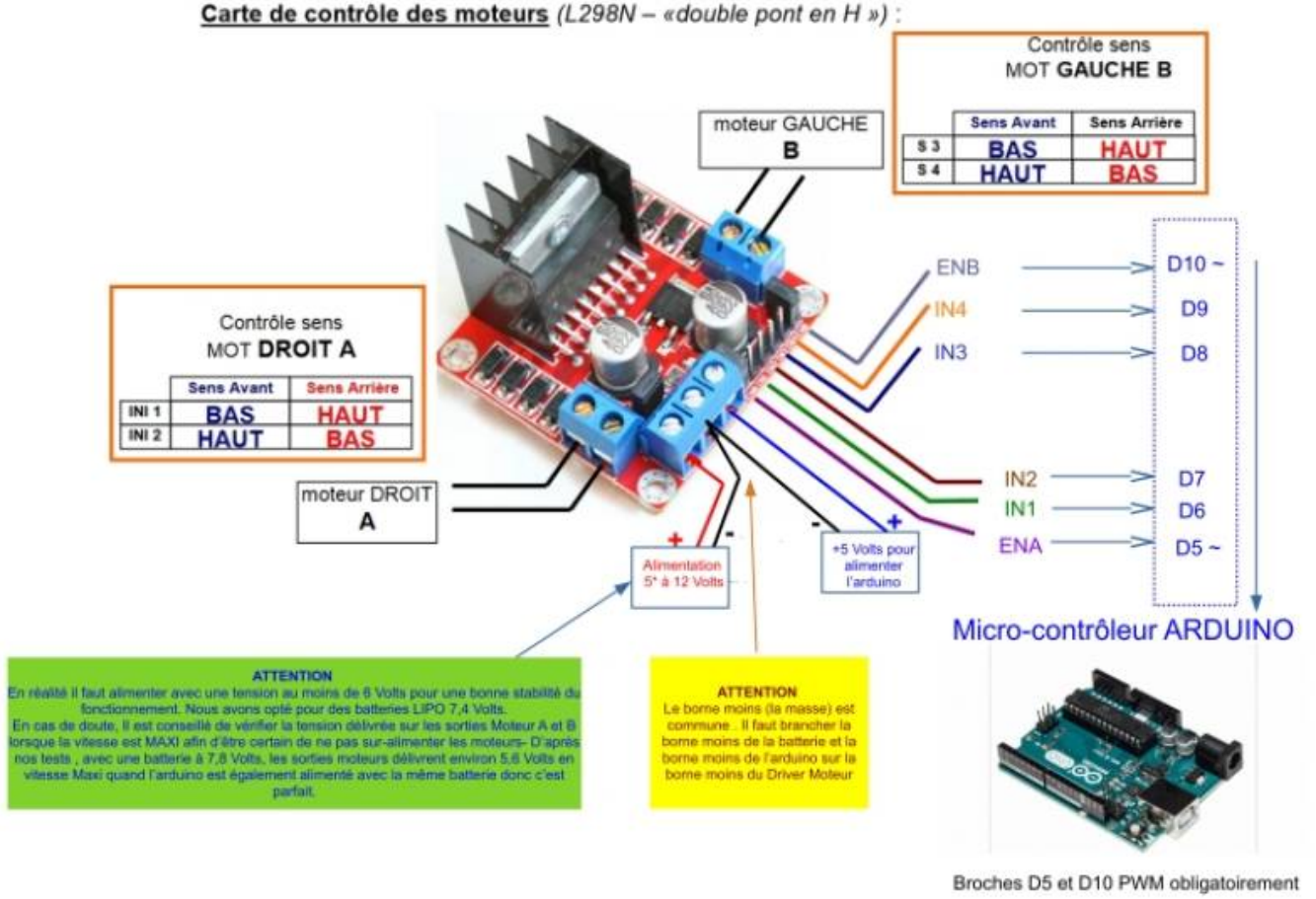
L298



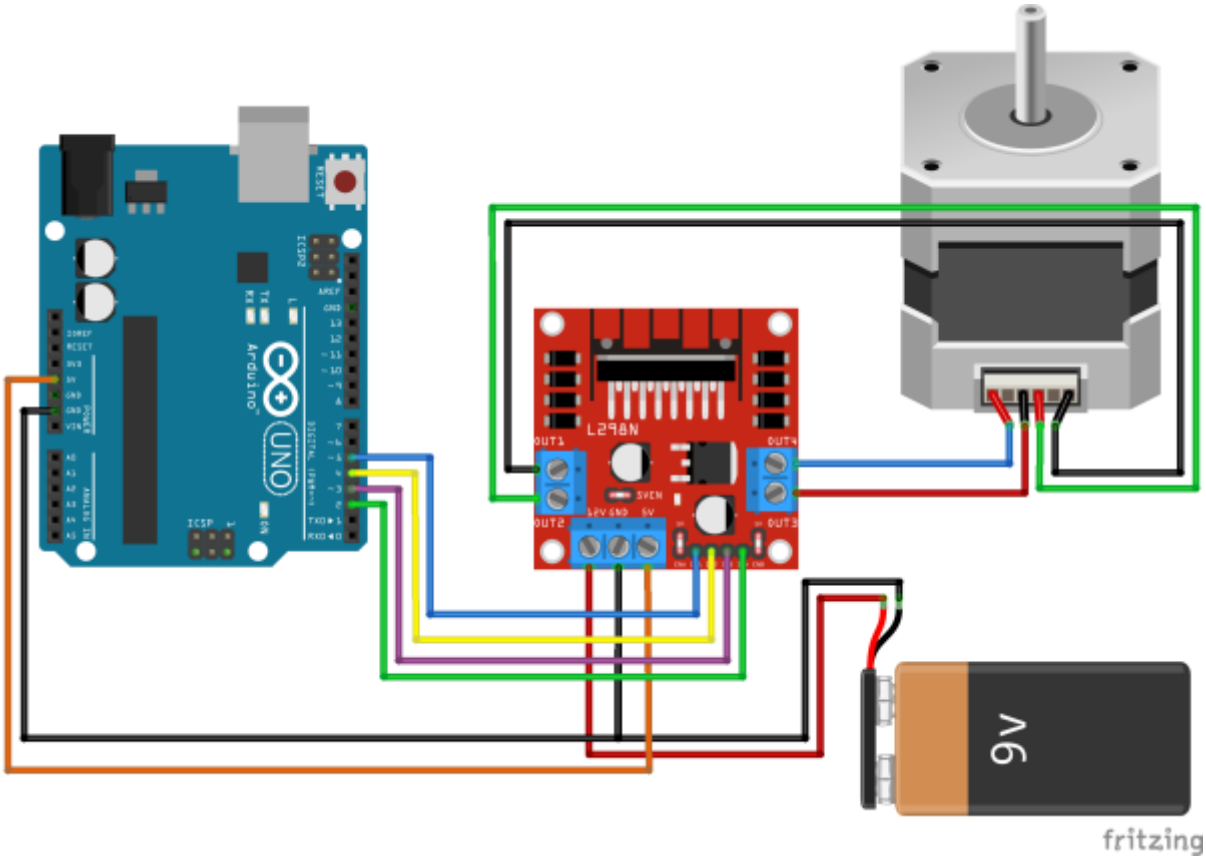
Programmer le Driver Moteur L298N (double pont en H)



Carte de contrôle des moteurs *(L298N – «double pont en H ») :*



Broches D5 et D10 PWM obligatoirement

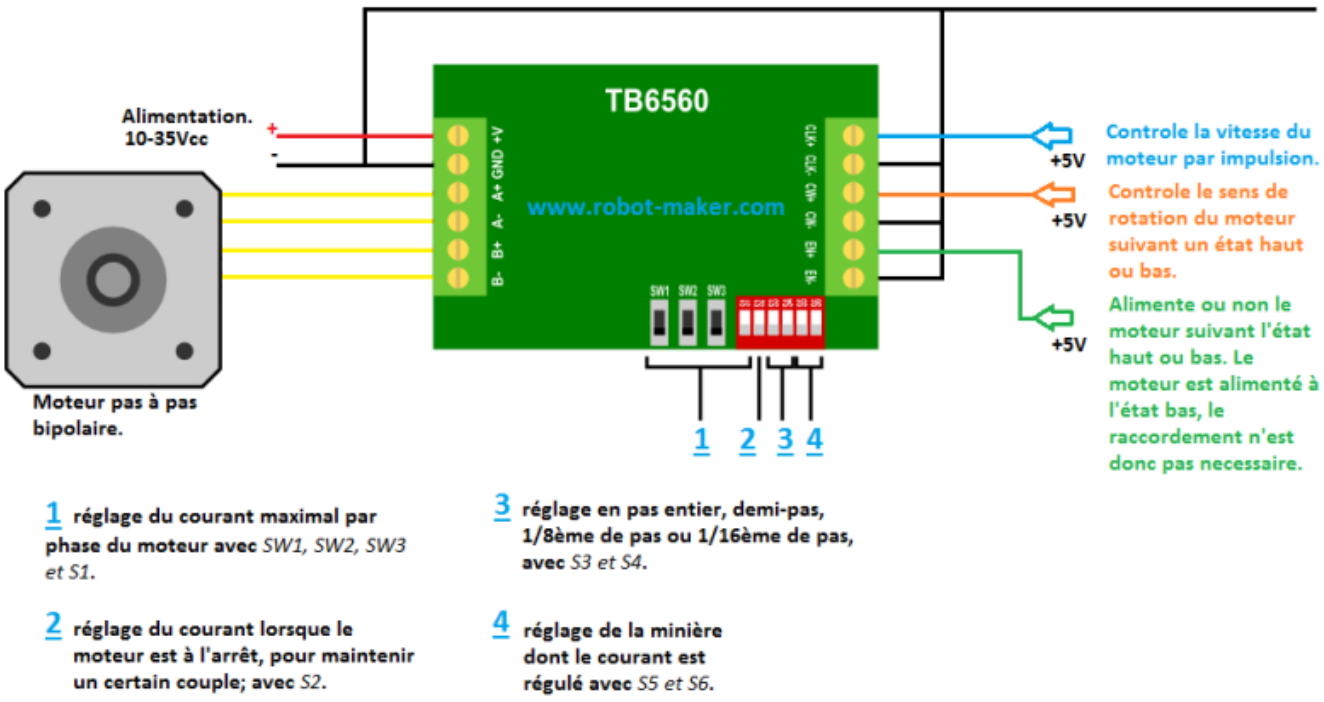


TB6560

Équipé d'un large dissipateur thermique, ce driver de moteur pas à pas permet de piloter un moteur pas à pas consommant jusqu'à 3A par phase avec à minima 2 fils donnant les informations d'avance et de direction.

De plus, cette carte est équipée de dip switch permettant un réglage aisé et pratique des différents paramètres de pilotage du moteur dont le courant par phase et la méthode d'excitation pouvant en théorie aller jusqu'à du 16ème de pas. Cependant en pratique, il est préférable de piloter le moteur en 8ème voir en quart de pas.

Les 3 entrées de la cartes sont protégées par 3 optocoupleurs et peuvent être connectées aussi bien en cathode commune qu'en anode commune si on veut inverser la logique de fonctionnement.



Datasheet simplifie du TB6560 EN

Datasheet complet TB6560 EN

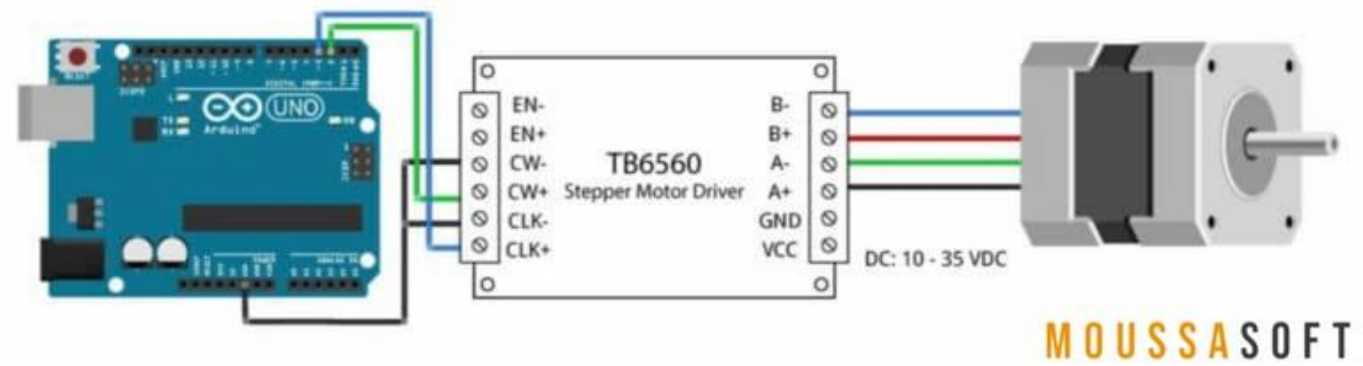
Câblage du TB6560 3A avec Arduino

Pour tirer pleinement parti du contrôleur TB6560 avec Arduino, il est essentiel de comprendre les connexions appropriées. Voici comment réaliser les connexions de base :

Connexions Requises

Les connexions essentielles à effectuer sont les suivantes :

- VCC (Tension d'Alimentation) : Connectez une source d'**alimentation de 10 à 35 VDC à la broche VCC du TB6560** pour alimenter le moteur.
- GND (Masse) : Assurez-vous de connecter la masse de l'alimentation à la broche GND pour une référence de tension commune.
- EN- et EN+ : Ces broches permettent de désactiver le moteur pas à pas en reliant EN- à GND. Si vous ne les utilisez pas, le moteur sera toujours activé.
- CW- et CW+ : Connectez CW- à la masse (GND) d'Arduino et CW+ à une broche d'Arduino (par exemple, broche 2) pour contrôler la direction de rotation.
- CLK- et CLK+ : Connectez CLK- à la masse (GND) d'Arduino et CLK+ à une broche d'Arduino (par exemple, broche 3) pour générer les impulsions de pas.
- A- et A+ ainsi que B- et B+ : Ces broches sont utilisées pour connecter les bobines du moteur pas à pas. La polarité n'a pas d'importance, il suffit de connecter les fils d'une bobine à A- et A+ et les fils de l'autre bobine à B- et B+.



Comment utiliser TB6560 Contrôleur Moteur pas à pas avec Arduino

Exemple de code Arduino TB6560

Maintenant que vous avez connecté le pilote et réglé les commutateurs DIP, il est temps de connecter l'Arduino à l'ordinateur et de télécharger du code. Vous pouvez télécharger l'exemple de code suivant sur votre Arduino à l'aide de l' IDE Arduino . Pour cet exemple spécifique, vous n'avez pas besoin d'installer de bibliothèques.

Ce croquis contrôle à la fois la vitesse, le nombre de tours et le sens de rotation du moteur pas à pas.

Vous pouvez copier le code en cliquant sur le bouton dans le coin supérieur droit du champ de code.

exemple001.ino

```
/* Example sketch to control a stepper motor with TB6560 stepper motor
driver and Arduino without a library. More info:
https://www.makerguides.com */

// Define stepper motor connections and steps per revolution:
#define dirPin 2
#define stepPin 3
#define stepsPerRevolution 1600

void setup() {
  // Declare pins as output:
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop() {
  // Set the spinning direction clockwise:
  digitalWrite(dirPin, HIGH);

  // Spin the stepper motor 1 revolution slowly:
  for (int i = 0; i < stepsPerRevolution; i++) {
    // These four lines result in 1 step:
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2000);
```

```
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2000);
  }

  delay(1000);

  // Set the spinning direction counterclockwise:
  digitalWrite(dirPin, LOW);

  // Spin the stepper motor 1 revolution quickly:
  for (int i = 0; i < stepsPerRevolution; i++) {
    // These four lines result in 1 step:
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(1000);
  }

  delay(1000);

  // Set the spinning direction clockwise:
  digitalWrite(dirPin, HIGH);

  // Spin the stepper motor 5 revolutions fast:
  for (int i = 0; i < 5 * stepsPerRevolution; i++) {
    // These four lines result in 1 step:
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
  }

  delay(1000);

  // Set the spinning direction counterclockwise:
  digitalWrite(dirPin, LOW);

  // Spin the stepper motor 5 revolutions fast:
  for (int i = 0; i < 5 * stepsPerRevolution; i++) {
    // These four lines result in 1 step:
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(500);
  }

  delay(1000);
}
```

Comment fonctionne le code :

Tutoriel sur le pilote de moteur pas à pas TB6560 avec Arduino

Le croquis commence par définir les broches de pas (CLK-) et de direction (CW-). Je les ai connectées aux broches 3 et 2 de l'Arduino.

L'instruction #define est utilisée pour donner un nom à une valeur constante. Le compilateur remplacera toutes les références à cette constante par la valeur définie lors de la compilation du programme. Ainsi, partout où vous mentionnez dirPin, le compilateur la remplacera par la valeur 2 lors de la compilation du programme.

J'ai également défini une stepsPerRevolution constante. Comme j'ai réglé le pilote sur le mode micropas de 1/8, je l'ai réglé sur 1600 pas par tour. Modifiez cette valeur si votre configuration est différente.

001.ino

```
// Définir les connexions du moteur pas à pas et les pas par tour :
#define dirPin 2
#define stepPin 3
#define stepsPerRevolution 1600
```

Dans la setup() section du code, toutes les broches de contrôle du moteur sont déclarées comme SORTIE numérique avec la fonction pinMode().

002.ino

```
void setup() {
  // Declare pins as output:
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}
```

Dans cette loop() section du code, nous laissons le moteur faire un tour lentement dans le sens horaire et un tour rapide dans le sens antihoraire. Ensuite, nous laissons le moteur faire 5 tours dans chaque sens à grande vitesse. Alors, comment contrôler la vitesse, le sens de rotation et le nombre de tours ?

003.ino

```
 // Set the spinning direction clockwise:
  digitalWrite(dirPin, HIGH);

  // Spin the stepper motor 1 revolution slowly:
  for(int i = 0; i < stepsPerRevolution; i++)
  {
    // These four lines result in 1 step:
```

```
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2000);
  }
```

Contrôler le sens de rotation :

Pour contrôler le sens de rotation du moteur pas à pas, nous définissons la broche DIR (direction) sur HIGH ou LOW. Pour cela, nous utilisons la fonction digitalWrite(). Selon la manière dont vous avez connecté le moteur pas à pas, le réglage de la broche DIR sur high permettra au moteur de tourner dans le sens horaire ou antihoraire.

Contrôler le nombre de pas ou de tours :

Dans cet exemple de schéma, les boucles for contrôlent le nombre d'étapes que le moteur pas à pas doit effectuer. Le code de la boucle for génère 1 (micro)pas du moteur pas à pas. Étant donné que le code de la boucle est exécuté 1600 fois (stepsPerRevolution), cela génère 1 tour. Dans les deux dernières boucles, le code de la boucle for est exécuté 8000 fois, ce qui génère 8000 (micro)pas ou 5 tours.

Notez que vous pouvez modifier le deuxième terme de la boucle for selon le nombre d'étapes souhaité. for(int i = 0; i < 800; i++) Cela donnerait 800 étapes ou un demi-tour.

Contrôler la vitesse :

La vitesse du moteur pas à pas est déterminée par la fréquence des impulsions que nous envoyons à la broche STEP. Plus la fréquence est élevée, plus le moteur tourne vite. Vous pouvez contrôler la fréquence des impulsions en modifiant delayMicroseconds() le code. Plus le délai est court, plus la fréquence est élevée, plus le moteur tourne vite.

Installation de la bibliothèque AccelStepper

La bibliothèque AccelStepper écrite par Mike McCauley est une bibliothèque géniale à utiliser pour votre projet. L'un de ses avantages est qu'elle prend en charge l'accélération et la décélération, mais elle possède également de nombreuses autres fonctions intéressantes.

Vous pouvez télécharger la dernière version de cette bibliothèque ici