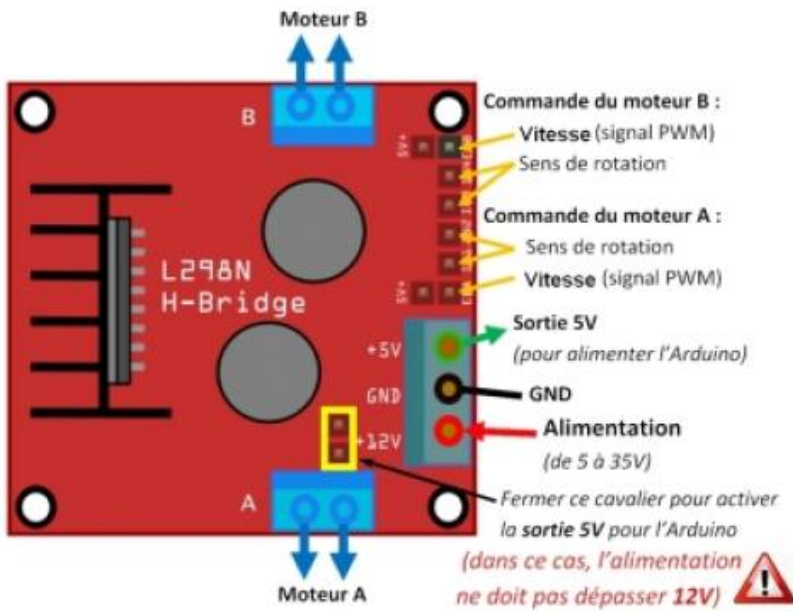


# moteur pas à pas L298

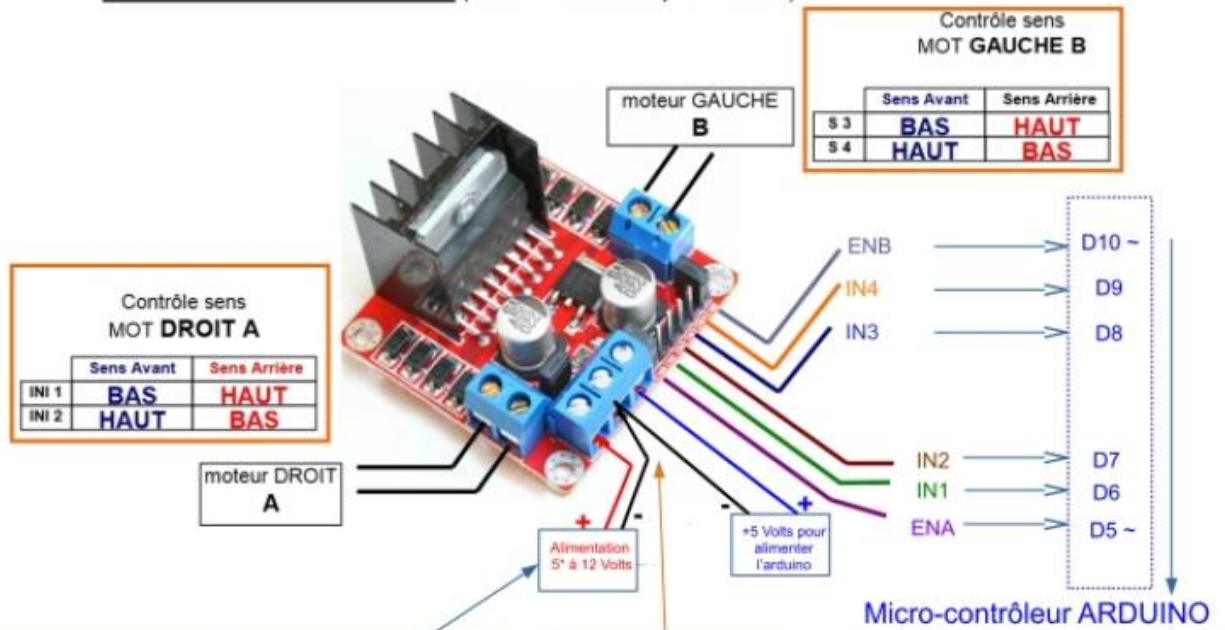
## L298



# Programmer le Driver Moteur L298N (double pont en H)



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



**ATTENTION**  
 En réalité il faut alimenter avec une tension au moins de 6 Volts pour une bonne stabilité du fonctionnement. Nous avons opté pour des batteries LIPO 7,4 Volts.  
 En cas de doute, il est conseillé de vérifier la tension délivrée sur les sorties Moteur A et B lorsque la vitesse est MAXI afin d'être certain de ne pas sur-alimenter les moteurs. D'après nos tests, avec une batterie à 7,8 Volts, les sorties moteurs délivrent environ 5,8 Volts en vitesse Maxi quand l'arduino est également alimenté avec la même batterie donc c'est parfait.

**ATTENTION**  
 Le borne moins (la masse) est commune. Il faut brancher la borne moins de la batterie et la borne moins de l'arduino sur la borne moins du Driver Moteur



Broches D5 et D10 PWM obligatoirement

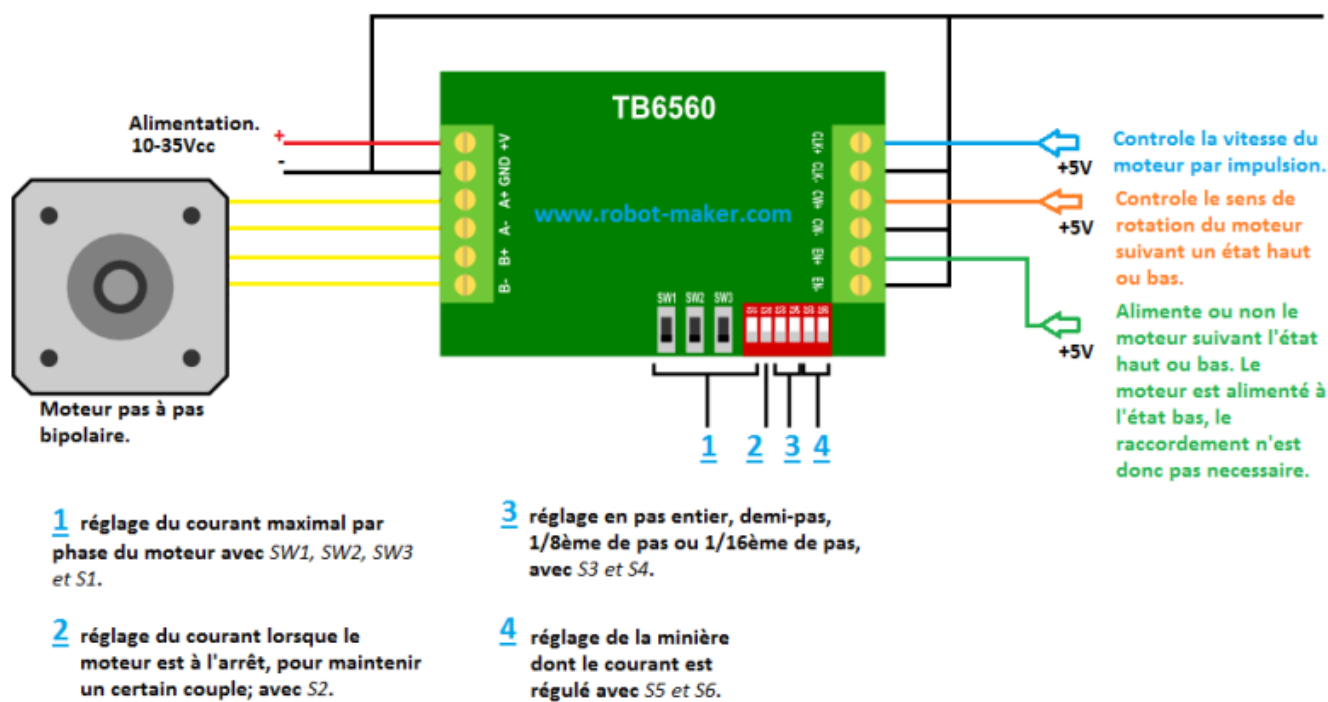


### TB6560

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 simplifié 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);
}
```

From:

<https://chanterie37.fr/fablab37110/> - **Castel'Lab le Fablab MJC de Château-Renault**

Permanent link:

<https://chanterie37.fr/fablab37110/doku.php?id=start:arduino:moteur:paspas&rev=1738436094>

Last update: **2025/02/01 19:54**

