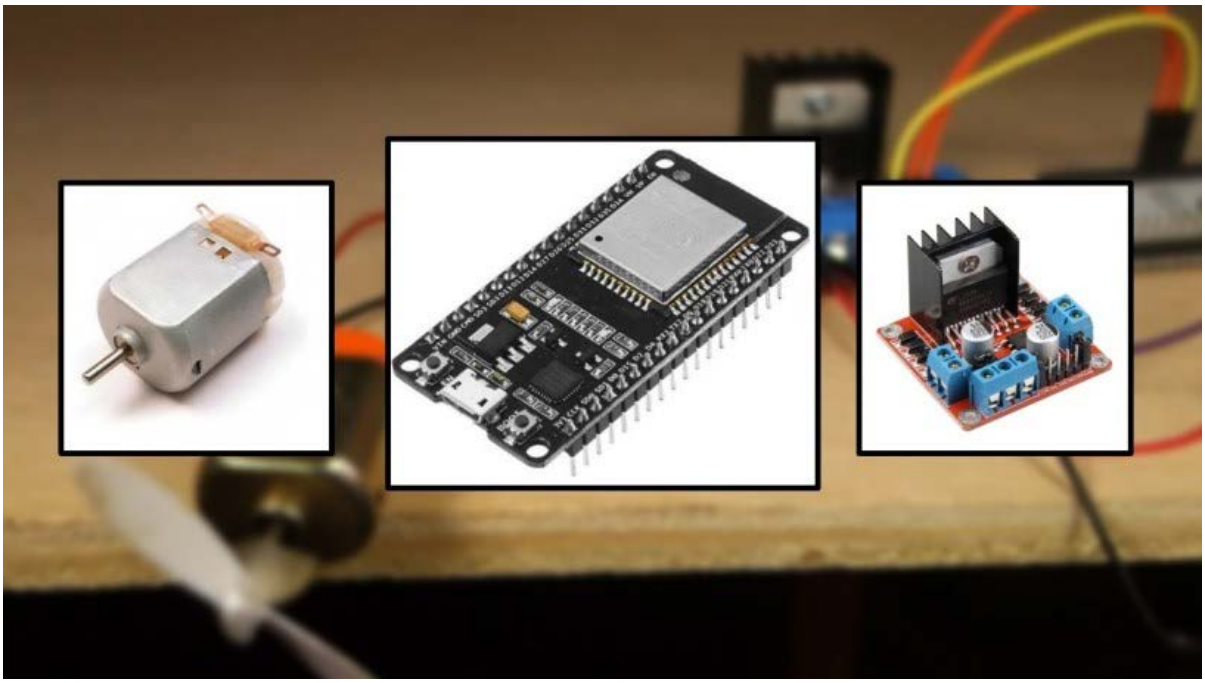
ESP32 avec moteur à courant continu et pilote de moteur L298N - Contrôle de la vitesse et de la direction

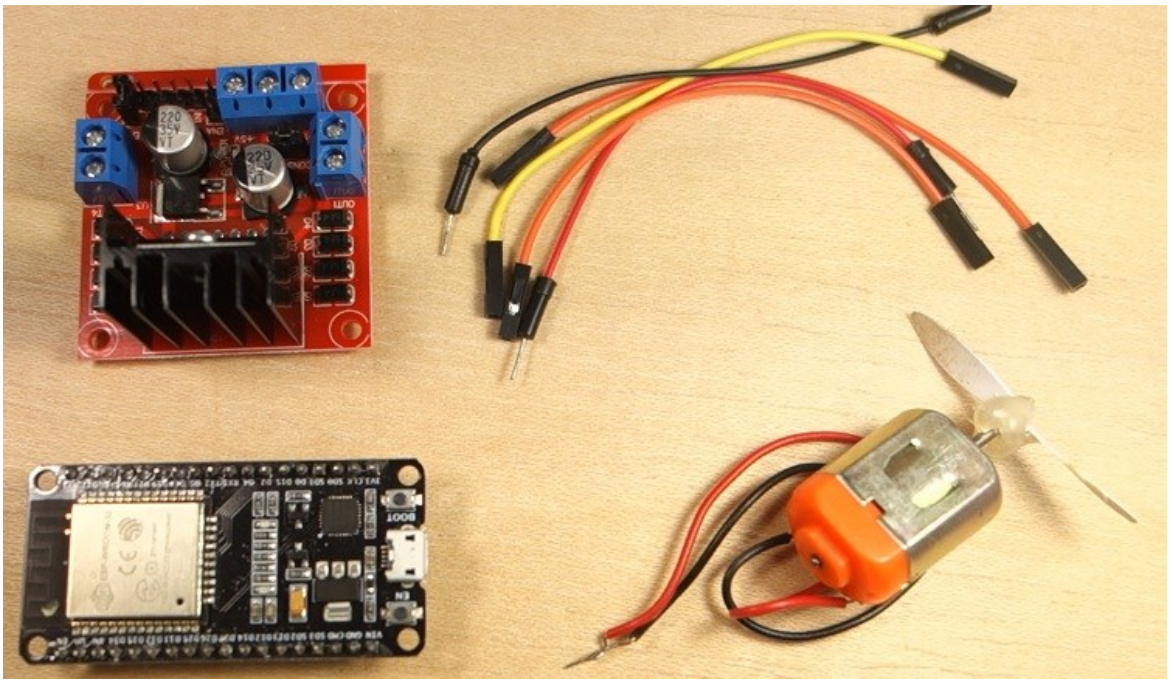
Ce tutoriel montre comment contrôler la direction et la vitesse d'un moteur à courant continu à l'aide d'un ESP32 et du driver de moteur L298N. Dans un premier temps, nous allons jeter un coup d'œil sur le fonctionnement du pilote de moteur L298N. Ensuite, nous allons vous montrer un exemple sur la façon de contrôler la vitesse et la direction d'un moteur à courant continu à l'aide de l'ESP32 avec Arduino IDE et du pilote de moteur L298N.



Remarque : il existe de nombreuses façons de contrôler un moteur à courant continu. Nous utiliserons le pilote de moteur L298N. Ce tutoriel est également compatible avec des modules de pilote de moteur similaires

Pièces requises

Pour compléter ce tutoriel, vous avez besoin des éléments suivants:



Carte ESP32 DOIT DEVKIT V1 -

Moteur

Contrôleur L298N

Source d'alimentation: 4x piles 1.5 AA ou bloc d'alimentation

2x condensateurs en céramique 100nF (en option)

1x interrupteur à glissière SPDT (en option)

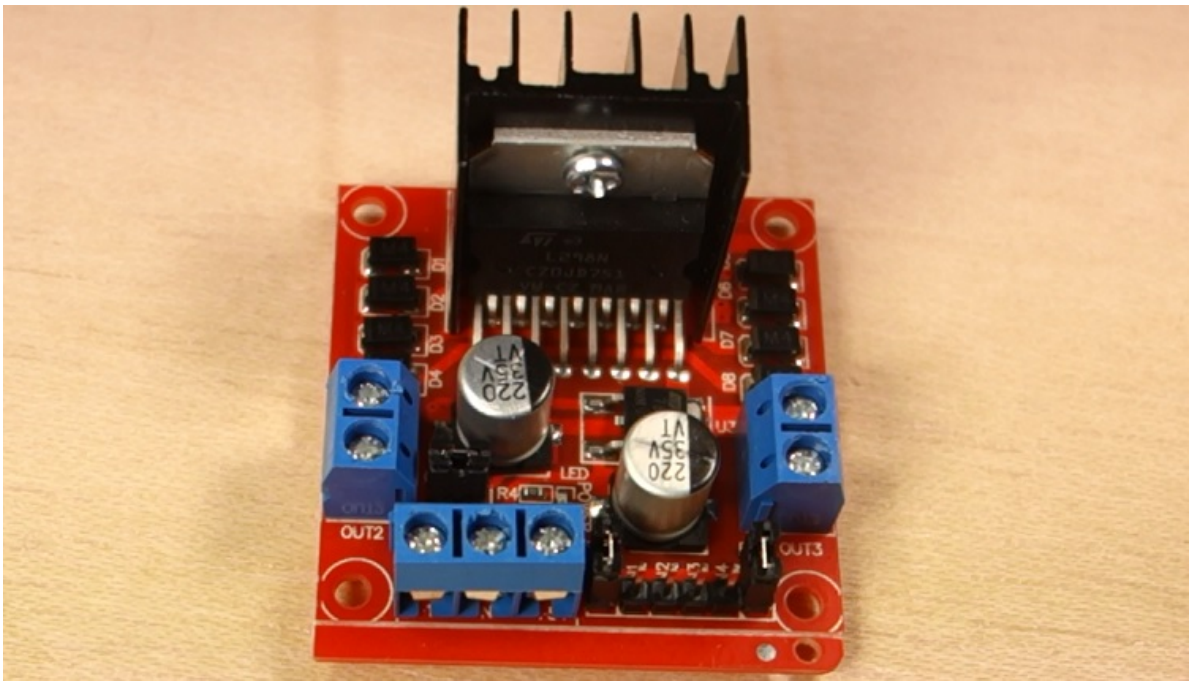
Fils de connexion

Présentation du driver de moteur L298N

Il existe de nombreuses façons de contrôler un moteur à courant continu. La méthode que nous allons utiliser ici convient à la plupart des moteurs amateurs nécessitant un fonctionnement en 6V ou 12V.

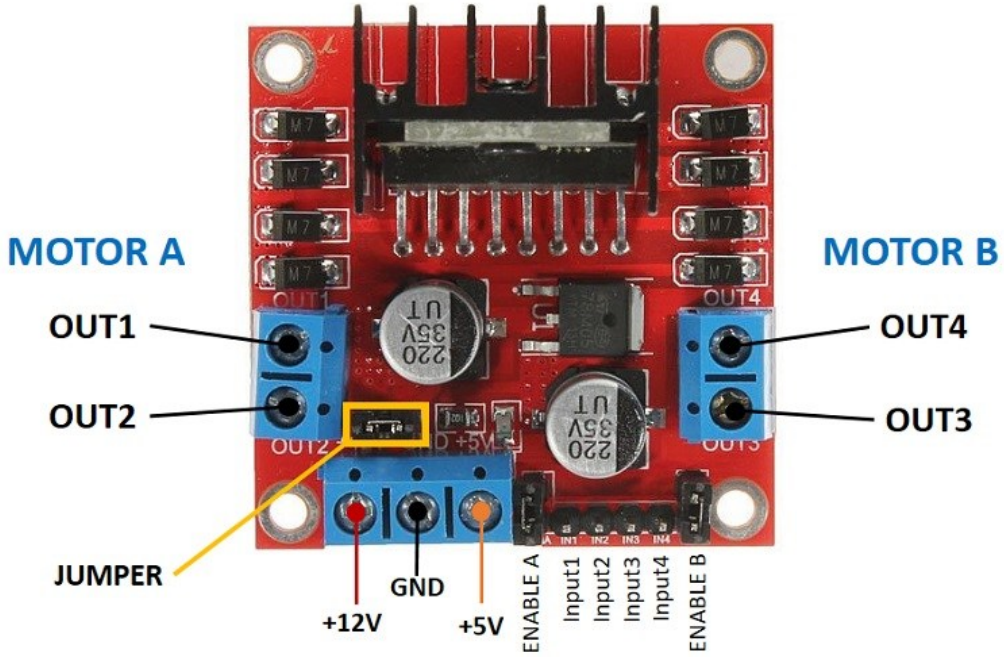
Nous allons utiliser le pilote de moteur L298N qui peut gérer jusqu'à 3A à 35V. De plus, cela nous permet de piloter simultanément deux moteurs à courant continu, ce qui est parfait pour construire un robot.

Le pilote de moteur L298N est illustré à la figure suivante:



Brochage du pilote de moteur L298N

Jetons un coup d'œil au brochage du pilote de moteur L298N et voyons comment cela fonctionne.



Le pilote de moteur a deux borniers de chaque côté pour chaque moteur. OUT1 et OUT2 à gauche et OUT3 et OUT4 à droite.

OUT1 : borne du moteur à courant continu A +

OUT2 : moteur à courant continu A - borne

OUT3 : borne moteur B + CC

OUT4 : moteur B courant continu - borne

En bas, vous avez un bornier à trois bornes avec + 12V , GND et + 5V . Le bornier + 12V sert à alimenter les moteurs. Le terminal + 5V est utilisé pour alimenter la puce L298N. Toutefois, si le cavalier est en place, la puce est alimentée à l'aide de l'alimentation du moteur et vous n'avez pas besoin de fournir une tension de 5V via la borne + 5V .

Remarque : si vous fournissez plus de 12V, vous devez retirer le cavalier et fournir 5V à la borne + 5V.

Il est important de noter que malgré le nom du terminal + 12V, avec la configuration que nous allons utiliser ici (avec le cavalier en place), vous pouvez fournir une tension comprise entre 6V et 12V. Dans ce didacticiel, vous utiliserez 4 piles AA 1,5 V dont la sortie combinée est d'environ 6 V, mais vous pouvez utiliser toute autre source d'alimentation appropriée. Par exemple, vous pouvez utiliser un bloc d'alimentation pour tester ce didacticiel.

En résumé:

+ 12V : Le terminal + 12V est l'endroit où vous devez connecter votre alimentation

GND : alimentation GND

+ 5V : fournir 5V si le cavalier est retiré. Agit comme une sortie 5V si le cavalier est en place

Cavalier : cavalier en place - utilise l'alimentation du moteur pour alimenter la puce. Cavalier enlevé: vous devez fournir 5V à la borne + 5V. Si vous fournissez plus de 12V, vous devez retirer le cavalier

En bas à droite, vous avez quatre broches d'entrée et deux bornes d'activation. Les broches d'entrée permettent de contrôler la direction de vos moteurs à courant continu, et les broches d'activation, de contrôler la vitesse de chaque moteur.

IN1: Entrée 1 pour le moteur A

IN2 : Entrée 2 pour le moteur A

IN3 : Entrée 1 pour le moteur B

IN4 : Entrée 2 pour le moteur B

EN1 : Activer la broche pour le moteur A

EN2 : Activer la broche pour le moteur B

Il y a des cavaliers sur les broches d'activation par défaut. Vous devez retirer ces cavaliers pour contrôler la vitesse de vos moteurs.

Contrôler les moteurs à courant continu avec le L298N

Maintenant que vous connaissez le driver de moteur L298N, voyons comment l'utiliser pour contrôler vos moteurs à courant continu.

Activer les pins

Les broches d'activation sont comme un commutateur ON et OFF pour vos moteurs. Par exemple:

Si vous envoyez un **signal HIGH** à la broche enable 1, le moteur A est prêt à être contrôlé et à la vitesse maximale;

Si vous envoyez un **signal LOW** à la broche enable 1, le moteur A s'éteint;

Si vous envoyez un **signal PWM** , vous pouvez contrôler la vitesse du moteur. La vitesse du moteur est proportionnelle au cycle de service. Toutefois, notez que, pour les cycles de fonctionnement courts, les moteurs risquent de ne pas tourner et de produire un bourdonnement continu.

SIGNAL SUR LE PIN D'ACTIVATION	ETAT MOTEUR
HAUTE	Moteur activé
FAIBLE	Moteur non activé
PWM	Moteur activé: vitesse proportionnelle au facteur de marche

Broches d'entrée

Les broches d'entrée contrôlent le sens de rotation des moteurs. Entrée 1 et entrée 2 moteur de commande A, et entrée 3 et 4 moteur de commande B.

Si vous appliquez LOW à l'entrée 1 et HIGH à l'entrée 2, le moteur tournera en avant.

Si vous appliquez l'alimentation inversement: HIGH pour entrer 1 et LOW pour entrer 2, le moteur tournera en arrière. Le moteur B peut être commandé en utilisant la même méthode mais en appliquant HIGH ou LOW à l'entrée 3 et à l'entrée 4.

Contrôle de moteur CC avec ESP32 - Vitesse et Direction

Maintenant que vous savez comment contrôler un moteur à courant continu avec le pilote de moteur L298N, construisons un exemple simple pour contrôler la vitesse et la direction d'un moteur à courant continu.

Schématique

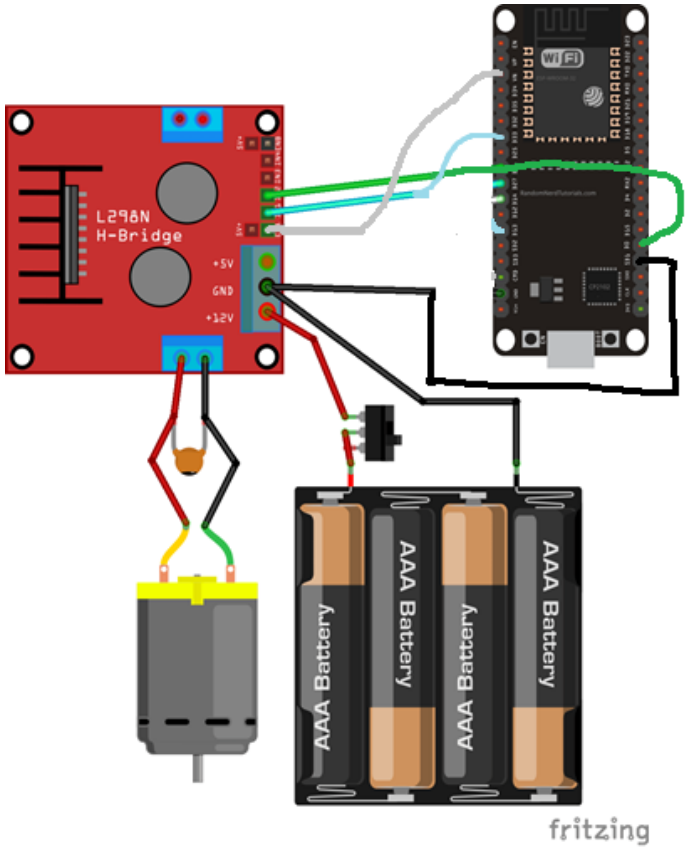
Le moteur que nous contrôlerons est connecté aux broches de sortie du moteur A; nous devons donc connecter les broches ENABLEA, INPUT1 et INPUT2 du pilote de moteur à l'ESP32. Suivez le schéma suivant pour connecter le moteur à courant continu et le pilote de moteur L298N à l'ESP32.

Contrôle de moteur CC avec ESP32 - Vitesse et Direction

Maintenant que vous savez comment contrôler un moteur à courant continu avec le pilote de moteur L298N, construisons un exemple simple pour contrôler la vitesse et la direction d'un moteur à courant continu.

Schématique

Le moteur que nous contrôlerons est connecté aux broches de sortie du moteur A; nous devons donc connecter les broches ENABLEA, INPUT1 et INPUT2 du pilote de moteur à l'ESP32. Suivez le schéma suivant pour connecter le moteur à courant continu et le pilote de moteur L298N à l'ESP32.



Le moteur à courant continu nécessite un grand saut de courant pour pouvoir être déplacé. Les moteurs doivent donc être alimentés par une source d'alimentation externe de l'ESP32. Par exemple, nous utilisons des piles 4AA, mais vous pouvez utiliser n'importe quelle autre source d'alimentation appropriée. Dans cette configuration, vous pouvez utiliser une alimentation de 6V à 12V.

L'interrupteur entre le support de batterie et le pilote de moteur est facultatif, mais il est très pratique de couper et d'appliquer de l'énergie. De cette façon, vous n'avez pas besoin de connecter en permanence, puis de déconnecter les câbles pour économiser de l'énergie.

Nous vous recommandons de souder un condensateur en céramique de 0,1 µF aux bornes positive et négative du moteur à courant continu, comme indiqué sur le schéma, afin d'aplanir les pointes de tension. (Remarque: les moteurs fonctionnent également sans le condensateur.)

Code de téléchargement

Le code suivant contrôle la vitesse et la direction du moteur à courant continu. Ce code n'est pas utile dans le monde réel, c'est un exemple simple pour mieux comprendre comment contrôler la vitesse et la direction d'un moteur à courant continu avec l'ESP32.

Déclaration des broches du moteur

Tout d'abord, vous définissez les GPIO auxquels les broches du moteur sont connectées. Dans ce cas, l'entrée 1 du moteur A est connectée à GPIO 27, l'entrée 2 à GPIO 26 et la broche Enable à GPIO 14.

```
int motor1Pin1 = 27;
int motor1Pin2 = 26;
int enable1Pin = 14;
```

Définition des propriétés PWM pour contrôler la vitesse

Comme nous l'avons vu précédemment, vous pouvez contrôler la vitesse du moteur à courant continu en appliquant un signal PWM à la broche d'activation du pilote de moteur L298N. La vitesse sera proportionnelle au cycle de service. Pour utiliser PWM avec l'ESP32, vous devez d'abord définir les propriétés du signal PWM.

```
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 200;
```

Dans ce cas, nous générons un signal de 30000 Hz sur le canal 0 avec une résolution de 8 bits. Nous commençons avec un cycle de travail de 200 (vous pouvez définir une valeur de cycle de travail de 0 à 255).

Pour la fréquence que nous utilisons, lorsque vous appliquez des cycles de travail inférieurs à 200, le moteur ne bouge pas et produit un bourdonnement étrange. C'est pourquoi nous avons défini un cycle de travail de 200 au début.

Remarque : les propriétés PWM que nous définissons ici ne sont qu'un exemple. Le moteur fonctionne bien avec d'autres fréquences.

setup()

Dans le `setup()`, vous commencez par définir les broches du moteur comme sorties.

```
pinMode(motor1Pin1, OUTPUT);
pinMode(motor1Pin2, OUTPUT);
pinMode(enable1Pin, OUTPUT);
```

Vous devez configurer un signal PWM avec les propriétés que vous avez définies précédemment à l'aide de la fonction `ledcSetup ()` qui accepte comme arguments, le `pwmChannel`, la `fréquence` et la `résolution`, comme suit:

```
ledcSetup(pwmChannel, freq, resolution);
```

Ensuite, vous devez choisir le GPIO qui génère le signal. Pour cela, utilisez la fonction `ledcAttachPin ()` qui accepte comme arguments le GPIO où vous voulez obtenir le signal et le canal qui le génère. Dans cet exemple, nous obtiendrons le signal dans le GPIO `enable1Pin`, qui correspond au GPIO 14. Le canal qui génère le signal est le `pwmChannel`, qui correspond au canal 0.

```
ledcAttachPin(enable1Pin, pwmChannel);
```

Déplacement du moteur à courant continu

Dans la `boucle (),` le moteur bouge. Le code est bien commenter ce que fait chaque partie du code. Pour faire avancer le moteur, définissez l'entrée 1 sur LOW et l'entrée 2 pintes sur HIGH. Dans cet exemple, le moteur avance en avant pendant 2 secondes (2000 millisecondes).

```
// Move the DC motor forward at maximum speed
Serial.println("Moving Forward");
digitalWrite(motor1Pin1, LOW);
digitalWrite(motor1Pin2, HIGH);
delay(2000);
```

Reculer le moteur à courant continu

Pour faire reculer le moteur à courant continu, vous appliquez l'alimentation inversée aux broches d'entrée du moteur. HIGH pour entrée 1 et LOW pour entrée 2.

```
// Move DC motor backwards at maximum speed
Serial.println("Moving Backwards");
digitalWrite(motor1Pin1, HIGH);
digitalWrite(motor1Pin2, LOW);
delay(2000);
```

Arrêtez le moteur à courant continu

Pour arrêter le moteur à courant continu, vous pouvez définir la broche d'activation sur LOW ou définir les broches d'entrée 1 et 2 sur LOW. Dans cet exemple, nous définissons les deux broches d'entrée sur LOW.

```
// Stop the DC motor
Serial.println("Motor stopped");
digitalWrite(motor1Pin1, LOW);
digitalWrite(motor1Pin2, LOW);
delay(1000);
```

Contrôler la vitesse du moteur à courant continu

Pour contrôler la vitesse du moteur à courant continu, nous devons modifier le cycle de service du signal PWM. Pour cela, utilisez la fonction `ledcWrite ()` qui accepte comme arguments le canal PWM qui génère le signal (pas le GPIO de sortie) et le cycle de travail, comme suit.

```
ledcWrite(pwmChannel, dutyCycle);
```

Dans notre exemple, nous avons une boucle while qui augmente l'incrément de 5 à chaque boucle.

```
// Move DC motor forward with increasing speed
digitalWrite(motor1Pin1, HIGH);
digitalWrite(motor1Pin2, LOW);
while (dutyCycle <= 255){
  ledcWrite(pwmChannel, dutyCycle);
  Serial.print("Forward with duty cycle: ");
  Serial.println(dutyCycle);
  dutyCycle = dutyCycle + 5;
  delay(500);
```

```
}
```

Lorsque la condition while n'est plus vraie, nous fixons à nouveau le cycle de travail à 200.

```
dutyCycle = 200;
```

Conclusion

Dans ce tutoriel, nous vous avons montré comment contrôler la direction et la vitesse d'un moteur à courant continu à l'aide d'un pilote de moteur ESP32 et L298N. En résumé:

Pour contrôler le sens de rotation du moteur à courant continu, utilisez les broches d'entrée 1 et 2;

Appliquez LOW à l'entrée 1 et HIGH à l'entrée 2 pour faire tourner le moteur en avant. Appliquez le pouvoir dans l'autre sens pour le faire tourner à l'envers;

Pour contrôler la vitesse du moteur à courant continu, vous utilisez un signal PWM sur la broche d'activation. La vitesse du moteur à courant continu est proportionnelle au facteur de marche.

Source

https://randomnerdtutorials.com/esp32-dc-motor-l298n-motor-driver-control-speed-direction/#more-61995

```
/*********
  Rui Santos
  Complete project details at http://randomnerdtutorials.com
*********/

// Motor A
int motor1Pin1 = 27;
int motor1Pin2 = 26;
int enable1Pin = 14;

// Setting PWM properties
const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 200;

void setup() {
  // sets the pins as outputs:
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);

  // configure LED PWM functionalities
  ledcSetup(pwmChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(enable1Pin, pwmChannel);

  Serial.begin(115200);

  // testing
  Serial.print("Testing DC Motor...");
}

void loop() {
  // Move the DC motor forward at maximum speed
  Serial.println("Moving Forward");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, HIGH);
  delay(2000);

  // Stop the DC motor
  Serial.println("Motor stopped");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  delay(1000);

  // Move DC motor backwards at maximum speed
  Serial.println("Moving Backwards");
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
  delay(2000);

  // Stop the DC motor
  Serial.println("Motor stopped");
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  delay(1000);
```

```
    // Move DC motor forward with increasing speed
    digitalWrite(motor1Pin1, HIGH);
    digitalWrite(motor1Pin2, LOW);
    while (dutyCycle <= 255)
    {
      ledcWrite(pwmChannel, dutyCycle);
      Serial.print("Forward with duty cycle: ");
      Serial.println(dutyCycle);
      dutyCycle = dutyCycle + 5;
      delay(500);
    }
    dutyCycle = 200;
  }

```