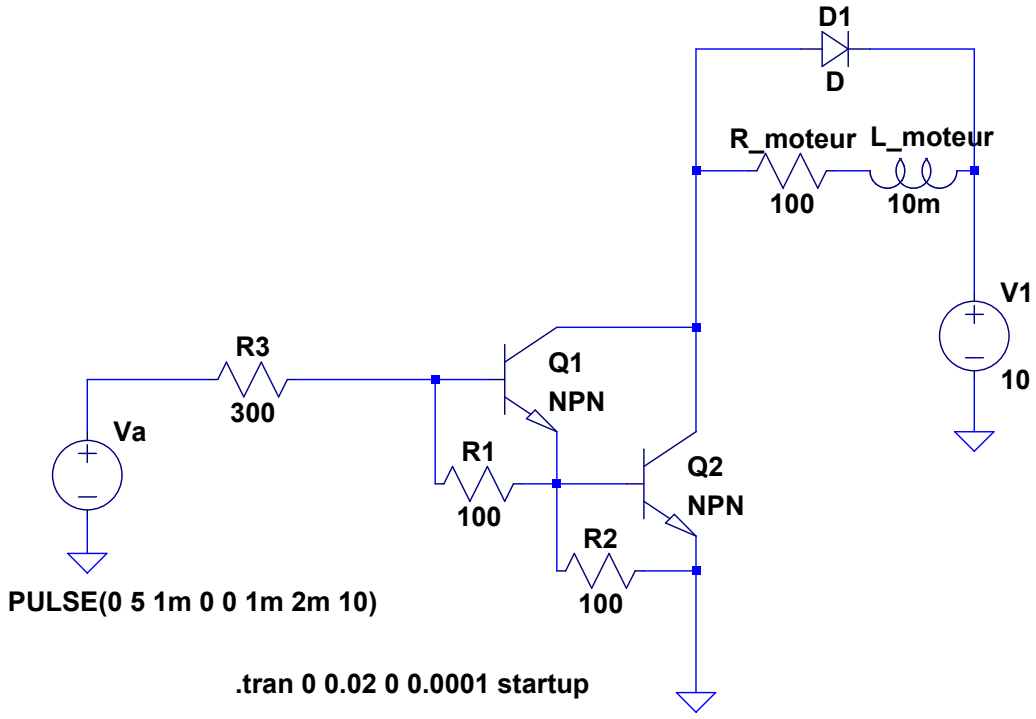
Commande d'un moteur à courant continu sur Arduino

1. Commande par un transistor

Lorsque le sens de rotation est fixé, un amplificateur à un transistor permet de commander un moteur à partir d'une sortie TTL délivrant quelques milliampères. Si le moteur consomme un courant de l'ordre de l'ampère ou plus, deux transistors en configuration Darlington permettent d'obtenir ce courant.



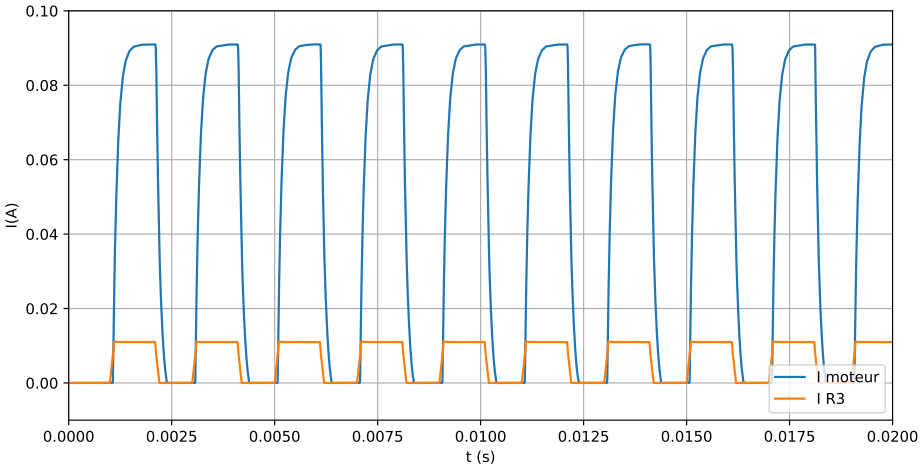
V_a est la tension de commande, qui vaut 0 ou 5 volts. V_1 est la source d'alimentation du moteur.

Le moteur est modélisé par une résistance de $100\,\Omega$ et une inductance de $10\,\mathrm{mH}$ en série, valeurs que l'on rencontre sur des petits moteurs. La force contre-électromotrice n'est pas prise en compte.

Pour alimenter le moteur, on utilise la technique du découpage, qui consiste à faire alterner des phases où une tension d'alimentation constante est fournie, à des phases où l'alimentation ne fournit aucune énergie.

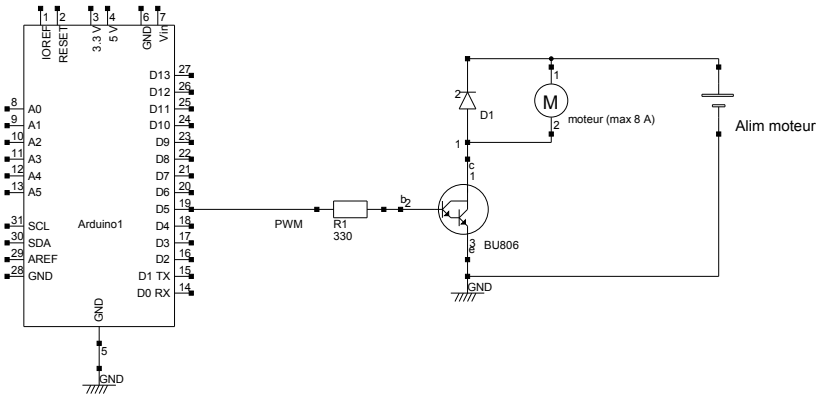
Lorsque les transistors sont à l'état saturé (Va au niveau haut), le moteur est alimenté. Lorsque les transistors sont à k'état bloqué (Va au niveau zéro), l'alimentation est déconnectée puisque le potentiel du collecteur des transistors n'est plus fixé : le bloc moteur + diode est alors en circuit ouvert. Juste après le passage de l'état saturé à l'état bloqué, le courant dans le moteur décroît continûment en s'écoulant dans la diode (appelée diode de roue libre).

La figure suivante montre le résultat d'une simulation effectuée avec LTSpice. La source Va délivre un signal TTL de période $2\ ms$.

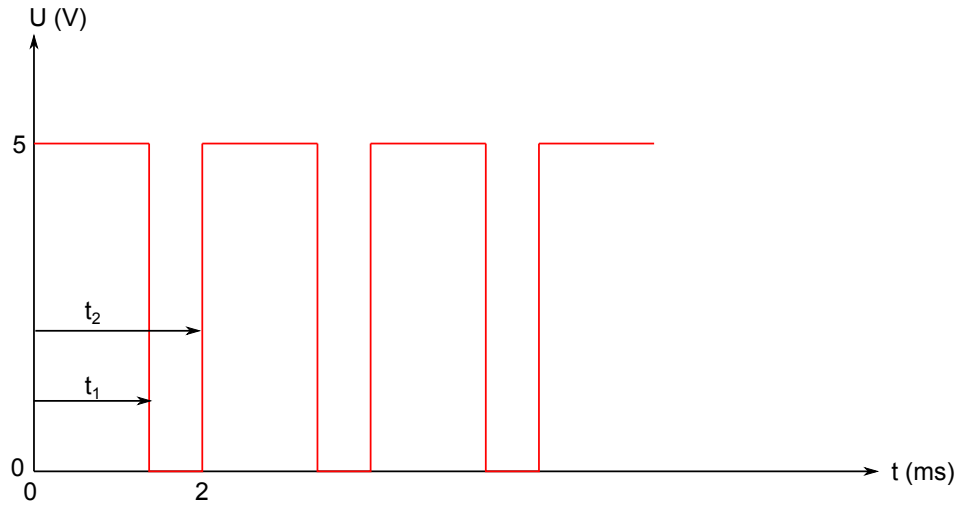


Avec un petit moteur, dont la résistance interne est élevée, la décroissance du courant l'alimentation est déconnectée est rapide. Elle se fait avec un temps caractéristique L/R. Pour les plus gros moteurs, la résistance interne est faible et il faudrait prendre en compte la force contre-électromotrice dans le modèle.

Voici un exemple de circuit avec un Arduino et un transistor Darlington :



Pour obtenir le découpage, le transistor est commandé par un signal PWM (par exemple délivré par la sortie D3 de l'arduino). Il s'agit d'un signal carré dont le rapport cyclique est variable, et dont la fréquence par défaut est de $490\,\mathrm{Hz}$ sur les sorties D3 et D11.



Le rapport cyclique est :

$$r = \frac{t_1}{t_2} \tag{1}$$

La valeur moyenne est égale au rapport cyclique multipliée par la tension maximale, ici $5\ V$. Pour obtenir un signal PWM sur la sortie 5 de l'Arduino, on utilise la commande :

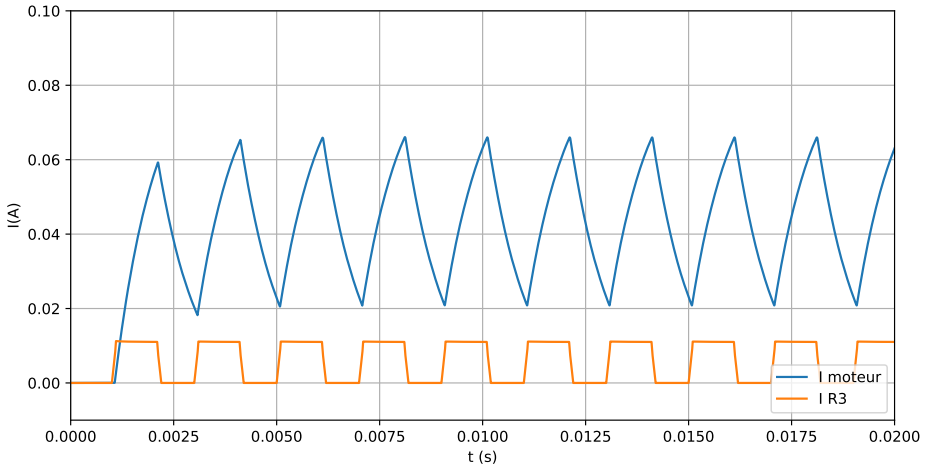
```
analogWrite(5,x)
```

où x est un entier 8 bits représentant le rapport cyclique. Par exemple, pour $x = 128$ on obtient un signal PWM de rapport cyclique $1/2$, dont la valeur moyenne est d'environ $2,5\ V$.

La résistance de $330\,\Omega$ placée dans le circuit base-émetteur permet de limiter le courant sortant de la borne D5 à environ $10\,\mathrm{mA}$.

Le transistor Darlington BU806 peut délivrer un courant de $8\ A$. Ce montage permet donc d'alimenter des petits moteurs relativement puissants. Si le sens de rotation ne doit pas être changé, c'est le montage à préférer en raison de son très faible coût. Le transistor fonctionne en commutation, donc il consomme peu de puissance comparé au moteur.

Lorsque le rapport L/R est plus grand, le courant dans le moteur ne descend pas à zéro, comme le montre la simulation ci-dessous faite avec $L = 100\,\mathrm{mH}$:

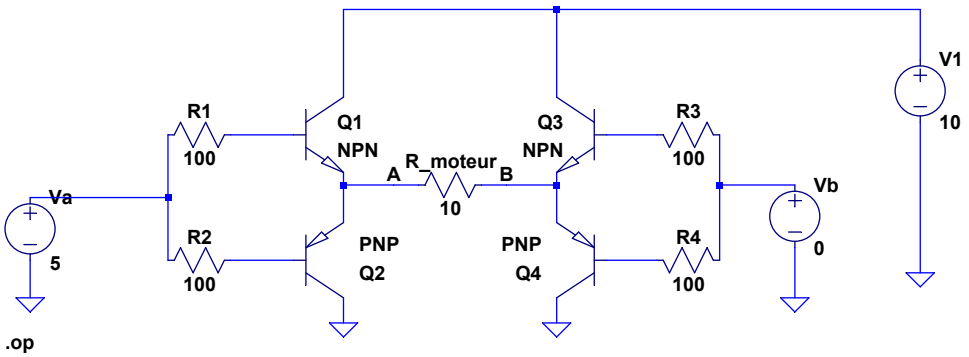


Cet effet peut être obtenu en ajoutant une bobine de forte inductance et de faible résistance en série avec le moteur. Lorsque le rapport cyclique est faible (faible vitesse de rotation), il est en effet souhaitable de réduire les variations de courant dans le moteur.

2. Commande bidirectionnelle

2.a. Pont en H à transistor à jonctions

Voici un circuit montrant le principe du pont en H, qui permet de changer le sens du courant dans le moteur :



Les transistors Q1 et Q3 sont de type NPN, alors que Q2 et Q4 sont PNP.

Il y a deux tensions de commande V_a et V_b. L'une est à 5 volts alors que l'autre est à 0 volt. V_1 est l'alimentation du moteur.

Voici le résultat d'une simulation de ce circuit avec LTSpice :

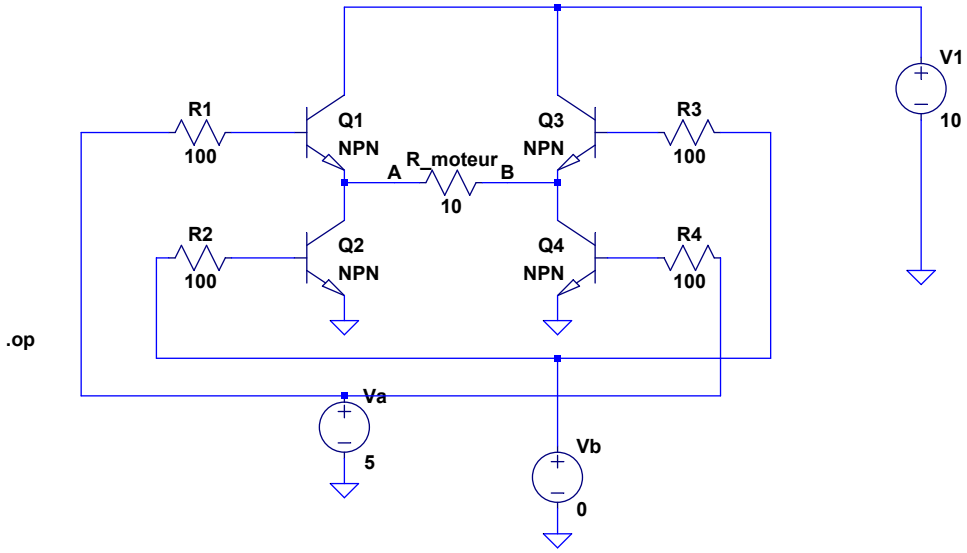
```
Q1 N001 N002 A 0 NPN
R1 N002 N005 100
```

```
R2 N006 N005 100
V1 N001 0 10
Va N005 0 5
Q2 0 N006 A 0 PNP
R3 N003 N004 100
R4 N007 N004 100
Q3 N001 N003 B 0 NPN
Q4 0 N007 B 0 PNP
Vb N004 0 0
R_moteur B A 10
```

```
V(n001):     10  voltage
V(n002):     4.73855     voltage
V(a):    3.82034     voltage
V(n005):     5   voltage
V(n006):     5   voltage
V(n003):     1.11798e-009    voltage
V(n004):     0   voltage
V(n007):     0.261454    voltage
V(b):    1.17966     voltage
Ic(Q4):  -0.261456   device_current
Ib(Q4):  -0.00261456     device_current
Ie(Q4):  0.264071    device_current
Ic(Q2):  -5.0001e-012    device_current
Ib(Q2):  6.17976e-012    device_current
Ie(Q2):  -1.17966e-012   device_current
Ic(Q3):  1.00001e-011    device_current
Ib(Q3):  -1.11798e-011   device_current
Ie(Q3):  1.17966e-012    device_current
Ic(Q1):  0.261456    device_current
Ib(Q1):  0.00261456  device_current
Ie(Q1):  -0.264071   device_current
I(R_moteur):     -0.264069   device_current
I(R4):   0.00261454  device_current
I(R3):   1.11798e-011    device_current
I(R2):   -6.17977e-012   device_current
I(R1):   -0.00261454     device_current
I(Vb):   0.00261454  device_current
I(Va):   -0.00261454     device_current
I(V1):   -0.261454   device_current
```

Le courant délivré par les sources de commande est de $26\,\mathrm{mA}$. Le courant traversant le moteur est de $260\,\mathrm{mA}$. L'amplification du courant n'est pas très forte car l'amplificateur n'a qu'un étage. Cependant, le sens du courant peut être inversé : il suffit d'appliquer $V_b = 5$ et $V_a = 0$. De plus, si $V_a = V_b$, on permet au courant délivré par le moteur de circuler, ce qui réalise un frein-moteur.

L'efficacité peut être améliorée avec un pont à 4 transistors NPN :



Dans ce cas, la commande se fait en rendant les transistors Q1 et Q4 passants et Q2 et Q3 bloquants, ou inversement. Il faut donc une porte logique pour appliquer deux niveaux contraires sur les résistances R1 et R2 (R3 et R4).

Voici la description SPICE et le résultat de la simulation :

```
Q1 N001 N003 A 0 NPN
R1 N003 N002 100
R2 N006 N005 100
V1 N001 0 10
Va N002 0 5
R3 N004 N005 100
R4 N007 N002 100
Q3 N001 N004 B 0 NPN
Vb N005 0 0
R_moteur B A 10
Q2 A N006 0 0 NPN
Q4 B N007 0 0 NPN
```

```
V(n001):     10  voltage
V(n002):     4.63882     voltage
V(a):    3.71226     voltage
V(n005):     5   voltage
V(n006):     3.71236e-010    voltage
V(n003):     1.00645e-009    voltage
V(n007):     0.931513    voltage
V(n004):     5   voltage
V(b):    0.0643716   voltage
Ic(Q4):  0.36479     device_current
Ib(Q4):  0.0406849   device_current
Ie(Q4):  -0.405474   device_current
Ic(Q2):  3.71246e-012    device_current
Ib(Q2):  -3.71236e-012   device_current
```

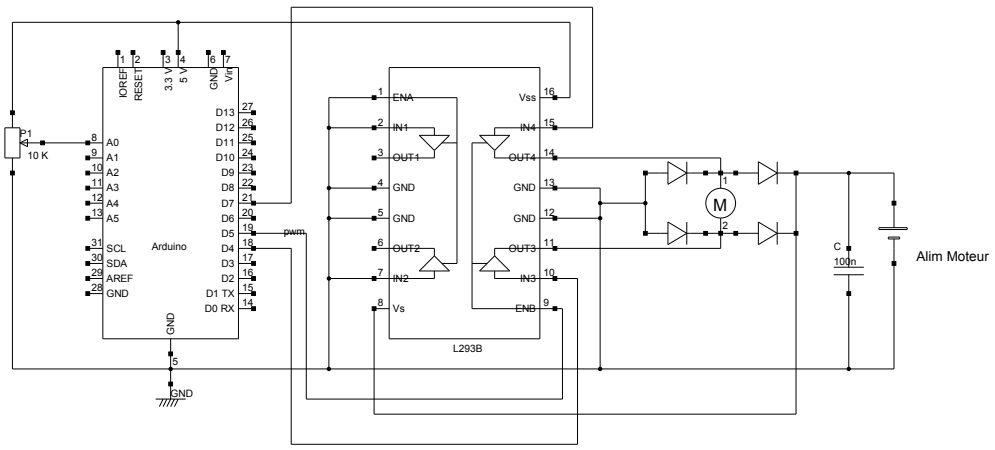
```
Ie(Q2):  -1e-016     device_current
Ic(Q3):  1.00001e-011    device_current
Ib(Q3):  -1.00645e-011   device_current
Ie(Q3):  6.43643e-014    device_current
Ic(Q1):  0.361177    device_current
Ib(Q1):  0.00361177  device_current
Ie(Q1):  -0.364789   device_current
I(R_moteur):     -0.364789   device_current
I(R4):   -0.0406849  device_current
I(R3):   1.00645e-011    device_current
I(R2):   3.71236e-012    device_current
I(R1):   -0.00361177     device_current
I(Vb):   -0.0406849  device_current
I(Va):   -0.00361177     device_current
I(V1):   -0.361177   device_current
```

Le courant traversant le moteur est de $365\,\mathrm{mA}$, nettement plus élevé qu'avec le premier circuit.

En pratique, on utilise un pont en H vendu sous forme de circuit intégré, qui possède plusieurs étages d'amplification de manière à fournir un gain en courant beaucoup plus grand. Nous allons voir deux exemples de circuit-intégrés prévus pour la commande des petits moteurs électriques : le L293 et le L298.

2.b. Commande par un circuit L293

Le circuit intégré L293B est un circuit à faible coût permettant de commander un moteur jusqu'à environ $1\,\mathrm{A}$. Il comporte deux ponts en H, ce qui permet de commander deux moteurs à courant continu, ou bien un seul moteur pas-à-pas dipolaire. Chaque pont peut délivrer au maximum $1\,\mathrm{A}$. Voici un exemple de montage avec un Arduino, dans lequel on utilise un seul pont :



Le circuit comporte deux bornes d'alimentation. La borne $V_s(8)$ est pour l'alimentation des transistors de l'étage de sortie, qui fournissent le courant au moteur. On relie donc cette borne

à l'alimentation du moteur (alim DC ou accumulateur). La borne $V_{ss}(16)$ est pour la partie logique du circuit ; on relie cette borne à l'alimentation 5 volts de l'arduino.

On utilise ici le pont B, commandé par les bornes de 9 à 15. Les deux tensions de commande sont appliquées en $IN3$ et $IN4$. Pour faire tourner le moteur dans un sens, on applique $IN3 = 5$ et $IN4 = 0$, et inversement pour l'autre sens. Pour freiner le moteur, on applique la même tension sur ces deux entrées. L'entrée ENB (enable B), permet d'activer ou de désactiver le pont B. En pratique, on envoit un signal PWM sur cette entrée pour moduler l'intensité reçue par le moteur. On doit donc la relier à une sortie de l'Arduino pouvant délivrer un signal PWM, ici la sortie D5.

Voici un programme Arduino permettant de tester ce montage :

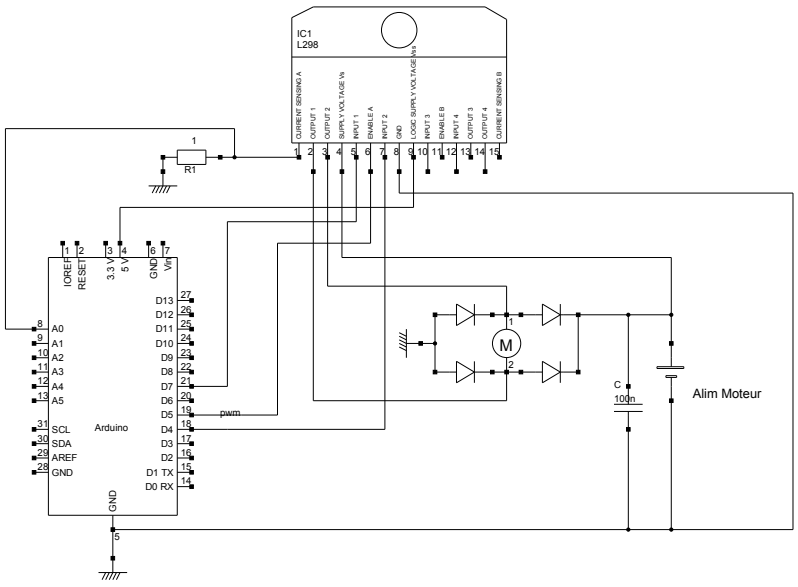
```
const byte enB = 5;
const byte in3 = 4;
const byte in4 = 7;
const byte control = 0;

int compte;

void mouv_sens_1(byte vitesse) {
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
  analogWrite(enB,vitesse);
}

void mouv_sens_2(byte vitesse) {
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
  analogWrite(enB,vitesse);
}

void setup() {
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
  pinMode(enB,OUTPUT);
  mouv_sens_1(0);
  Serial.begin(9600);
}

void loop() {
   int a0;
   int vitesse;
   a0 = analogRead(control);
   vitesse = (a0/4-128)*2;
   Serial.println(vitesse);
   if (vitesse > 0) mouv_sens_1(vitesse);
   else if (vitesse <=0) mouv_sens_2(-vitesse);
   delay(1000);

}
```

Le potentiomètre de $10\,\mathrm{k}\Omega$ permet de faire varier la tension sur l'entrée analogique A0. Le programme lit cette tension, la convertit en vitesse positive ou négative et commande en conséquence le moteur. La valeur de la vitesse est aussi envoyée par le port série pour affichage sur la console.

Pour les faibles valeurs de vitesse, le moteur ne tourne pas, ou nécessite d'être lancé manuellement. Pour le faire tourner lentement, on peut d'abord appliquer une vitesse importante puis la réduire. À la vitesse maximale, lorsque le rapport cyclique du signal PWM est 1, la tension aux bornes du moteur est notablement inférieure à la tension délivrée par l'alimentation. La chute de tension est environ celle de deux diodes, de l'ordre de $1,5$ à 2 V. Il faut si possible tenir compte de cette chute de tension pour le choix de l'alimentation. Par exemple, si le moteur est prévu pour une tension nominale de $6\ V$, il faut utiliser une alimentation de $7,5\ V$ ou un peu plus.

2.c. Commande avec un L298

Le L298 est un double pont similaire au L293, mais qui peut délivrer jusqu'à $1\ \mathrm{A}$ par pont. Il possède en plus, pour chaque pont, une borne à relier à la masse dans laquelle le courant moteur s'écoule (le courant qui traverse les deux transistors de l'étage de sortie). En reliant par une petite résistance, on peut mesurer le courant moteur. On considère ici le L298N, dont le boitier Multiwatt15 peut acceuillir un radiateur. Voici un exemple de montage avec un Arduino, dans lequel ont utilise seulement le pont A. Une résistance de $1\ \Omega$ relie la borne SENSA(1) à la masse. La tension de cette borne est lue sur l'entrée analogique A0, ce qui donne une échelle de 1 V par ampère. Cette résistance doit pouvoir dissiper au moins $4\ \mathrm{W}$. Si l'on ne souhaite pas mesurer le courant, on relie cette borne directement à la masse. Comme précédemment, l'entrée ENA est reliée à une sortie de l'Arduino pouvant délivrer un signal PWM.



2.d. Arduino Motor-Shield

Il existe des cartes pour Arduino avec un L298 prêt à l'emploi, comportant les diodes de protection, les résistances de mesure et les borniers de connexion pour les moteurs et l'alimentation. On considère le Motor-Shield officiel Arduino.

Par défaut, le moteur est alimenté par la borne Vin de l'arduino, ce qui signifie qu'il faut alimenter celui-ci avec une source assez puissante pour le moteur utilisé, et qui doit délivrer

une tension comprise entre 7 et 12 V. Pour plus de souplesse dans le choix de l'alimentation du moteur, il est préférable de couper le pont *Vin connect* situé à l'arrière de la platine, ce qui permet d'alimenter le moteur par le bornier. L'arduino est alors alimenté de son côté, soit par une pile 9 V, soit par le port USB.

Le double pont est un L298 à boitier PowerSO20. Il est commandé par l'intermédiaire d'un circuit logique d'interface. Chaque pont (A ou B) est commandé par 3 bornes :

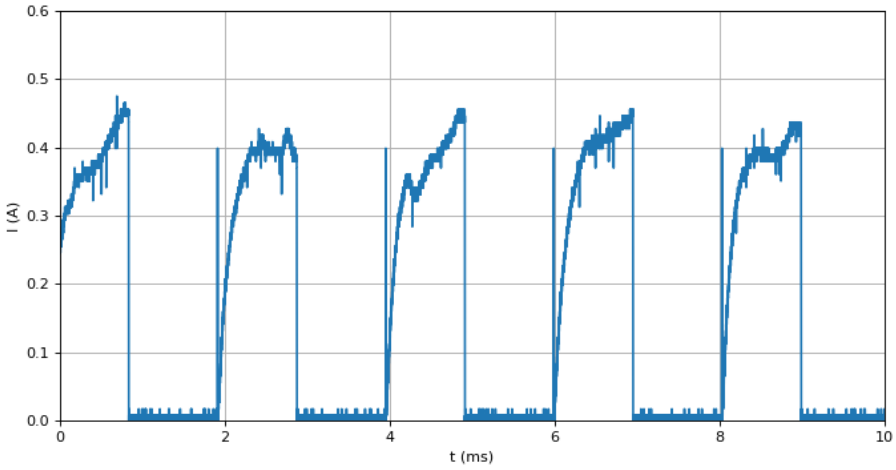
- ▷ DIR : la direction (0 ou 5 V)
- ▷ PWM : signal envoyé sur l'entrée EN (enable) du pont, pour ajuster la vitesse du moteur.
- ▷ BRAKE : commande du frein moteur (0 = inactif, 5 V = actif).

La sortie DIRA est reliée directement à l'entrée INPUT1 du L298. Lorsque BRAKEA=0, NON(DIRA) est appliqué sur l'entrée INPUT2. Ainsi lorsque BRAKE=0, les deux entrées INPUT1 et INPUT2 ont des niveaux contraires. Lorsque BRAKEA=1, INPUT1=INPUT2 et le moteur est donc freiné.

Voici un programme de test, qui fournit un PWM de rapport cyclique $120/256$:

```
const int pwm_a = 3;
const int pwm_b = 11;
const int dir_a = 12;
const int dir_b = 13;
const int brake_a = 9;
const int brake_b = 8;

void setup() {
  pinMode(pwm_a,OUTPUT);
  pinMode(pwm_b,OUTPUT);
  pinMode(dir_a,OUTPUT);
  pinMode(dir_b,OUTPUT);
  pinMode(brake_a,OUTPUT);
  pinMode(brake_b,OUTPUT);
  analogWrite(pwm_a,0);
  analogWrite(pwm_b,0);
  digitalWrite(dir_a,LOW);
  digitalWrite(dir_b,LOW);
  digitalWrite(brake_a,LOW);
  digitalWrite(brake_b,LOW);

  analogWrite(pwm_a,120);

}

void loop() {
  delay(1000);
}
```

La figure ci-dessous montre un enregistrement du courant effectué avec un petit moteur (tension nominale 6 V, courant maximal en pleine charge 700 mA). La tension aux bornes de la résistance de mesure est sur l'entrée analogique A0, à raison de $1,65$ volts par ampère. Le moteur entraîne un réducteur.



Pour ce moteur, on voit que le temps d'établissement du courant est de l'ordre de $0,5$ ms. Pour des rapports cycliques faibles (par exemple $40/256$), la durée de maintient du courant est trop faible pour démarrer le moteur. Pour remédier à cela, on peut d'abord appliquer un courant permettant au moteur de démarrer pendant $100\ ms$:

```
analogWrite(pwm_a,200);
delay(100);
analogWrite(pwm_a,40);
```

La fréquence par défaut du signal PWM est 490 Hz. Un signal périodique binaire comme le PWM est généré par un compteur. Celui-ci est incrémenté à chaque top de l'horloge et sa valeur est comparée à un seuil. Lorsque le seuil est atteint le compteur est initialisé. Il est possible d'incrémenter le compteur non pas tous les tops d'hologe, mais tous les 8, 64, 256 ou 1024 tops. On peut ainsi modifier la période du signal. Pour faire ce changement, il faut tout d'abord savoir quels sont les timers utilisés pour les différentes sorties PWM. Sur l'arduino MEGA (ATmega2560), les timers utilisés sont :

- ▷ Sorties 3,4 : timer 0
- ▷ Sorties 12,11 : timer 1
- ▷ Sorties 10,9 : timer 2
- ▷ Sorties 5,3,2 : timer 3
- ▷ Sorties 8,7,6 : timer 4

Pour changer le facteur multiplicatif de l'horloge (clock prescaler), il faut agir sur le registre `TCCRnB`, où n est le numéro du timer. Dans le cas présent, il faut agir sur `TCCR3B` (sortie 3). Le réglage de l'horloge du timer se fait avec les 3 premiers bits de ce registres (nommés CSn0, CSn1, CSn2). Les valeurs à placer sur ces 3 bits sont 1,2,3,4,5, respectivement pour les facteurs 1,8,64,256,1024 (sur ATmega 2560). La valeur par défaut est 3, qui donne une fréquence de $490\ Hz$. Voici ce qu'il faut faire pour appliquer le facteur 4, qui permet d'obtenir une fréquence de $120\ Hz$:

```
TCCR3B &= ~7;
TCCR3B |= 4; // prescaler = 256
```

Pour plus d'informations sur les signaux PWM, voir la page Secrets of Arduino PWM. Pour le bon usage des registres de configuration des timers, il faut consulter la documentation officielle des microcontroleurs ATmega.

Il est aussi possible de régler plus finement la fréquence du PWM en utilisant le module `TimerThree`, de la manière suivante :

```
#include <TimerThree.h>

Timer3.initialize();
microseconds = 10000;
duty = 512;
output = 3;
Timer3.pwm(output,duty,microsecondes);
```

Le rapport cyclique est codé sur 10 bits (0 à 1024). La période est en microsecondes.