TP Véhicule autonome

Capacités exigibles du programme :

Conversion électromécanique de puissance continu.
— Mettre en œuvre une machine à courant

Liste du matériel :

- Alimentation stabilisée (~ 5 V)
- Plaquette de montage de composants
- Oscilloscope
- Résistances de 10 kΩ et 100 Ω ($\times 4$)
- Condensateur de 100 nF
- Diodes silicium ($\times 5$)
- Transistor MOSFET « Canal n IRF 640 »
- Transistors **NPN** BDX33C ($\times 2$)
- Transistors PNP BDX34C ($\times 2$)
- Petit moteur à courant continu (avec hélice)
- Module Arduino (avec logiciel installé)
- Véhicule (jouet) avec connexions aux moteurs déportée

1 Pilotage en vitesse d'un moteur à courant continu

Pilotage de la mise en route et de l'arrêt :
Déterminer et mettre en œuvre un protocole expérimental permettant de réaliser le pilotage de la mise en route et de l'arrêt d'un moteur à courant continu depuis le programme (fourni) d'un module Arduino.

Pilotage de la vitesse de rotation :
Déterminer et mettre en œuvre un protocole expérimental permettant de réaliser le pilotage de la vitesse de rotation d'un moteur à courant continu depuis le programme (fourni) d'un module Arduino.

2 Pilotage du sens de rotation d'un moteur à courant continu

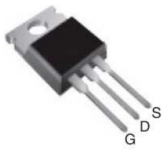
Déterminer et mettre en œuvre un protocole expérimental permettant de réaliser le pilotage du sens de rotation d'un moteur à courant continu depuis le programme (fourni) d'un module Arduino.

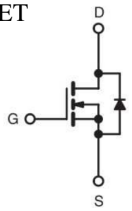
3 Pilotage d'un véhicule

Déterminer et mettre en œuvre un protocole expérimental permettant de réaliser le pilotage du petit véhicule (jouet) à disposition pour lui faire effectuer un créneau depuis le programme d'un module Arduino.

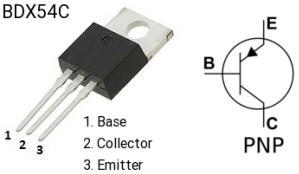
A Annexe 1 - Quelques données sur les transistors à disposition

Transistor MOSFET
Canal n IRF 640





N-Channel MOSFET



BDX53B, BDX53C (NPN), BDX54B, BDX54C (PNP)

Plastic Medium-Power Complementary Silicon Transistors

These devices are designed for general−purpose amplifier and low−speed switching applications.

Features

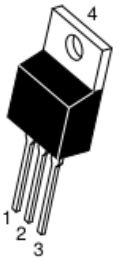
- High DC Current Gain −
  h_{FE} = 2500 (Typ) @ I_C = 4.0 Adc
- Collector Emitter Sustaining Voltage − @ 100 mAdc
  $V_{CEO(sus)}$ = 80 Vdc (Min) − BDX53B, 54B
  = 100 Vdc (Min) − BDX53C, 54C
- Low Collector−Emitter Saturation Voltage −
  $V_{CE(sat)}$ = 2.0 Vdc (Max) @ I_C = 3.0 Adc
  = 4.0 Vdc (Max) @ I_C = 5.0 Adc
- Monolithic Construction with Built−In Base−Emitter Shunt Resistors
- These Devices are Pb−Free and are RoHS Compliant*

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Collector−Emitter Voltage BDX53B, BDX54B BDX53C, BDX54C	V_{CEO}	 80 100	Vdc
Collector−Base Voltage BDX53B, BDX54B BDX53C, BDX54C	V_{CB}	 80 100	Vdc
Emitter−Base Voltage	V_{EB}	5.0	Vdc
Collector Current − Continuous − Peak	I_C	8.0 12	Adc
Base Current	I_B	0.2	Adc
Total Device Dissipation @ T_C = 25°C Derate above 25°C	P_D	65 0.48	W W/°C
Operating and Storage Junction Temperature Range	T_J, T_{stg}	−65 to +150	°C

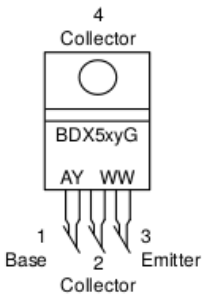
Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

DARLINGTON
8 AMPERE
COMPLEMENTARY SILICON
POWER TRANSISTORS
80−100 VOLTS, 65 WATTS



TO−220
CASE 221A
STYLE 1

MARKING DIAGRAM & PIN ASSIGNMENT



BDX5xy = Device Code
x = 3 or 4
y = B or C
A = Assembly Location
Y = Year
WW = Work Week
G = Pb−Free Package

B Annexe 2 - Extrait d'un tutoriel [1] concernant l'alimentation d'un moteur à courant continu

maximale autorisée pour donner un coup de fouet à votre moteur, mais ne restez jamais dans une plage trop élevée ! Une deuxième conséquence de cette relation concerne le moment du démarrage du moteur. En effet, la relation entre tension et vitesse n'est pas tout à fait linéaire pour les tensions faibles, elle est plutôt "écrasée" à cet endroit. Du coup, cela signifie que le moteur n'arrivera pas à tourner pour une tension trop basse. C'est un peu comme si vous aviez une tension de seuil de démarrage. En dessous de cette tension, le moteur est à l'arrêt, et au dessus il tourne correctement avec une relation de type "100 trs/min/volts" (autrement dit, le moteur tournera à 100 tours par minutes pour 1 volt, puis 200 tours par minutes pour 2 volts et etc etc... bien entendu le 100 est pris comme un exemple purement arbitraire, chaque moteur a sa caractéristique propre).

7.1.1.3.2 Lien entre courant et couple Comme nous venons de le voir, la vitesse est une sorte d'image de la tension. Passons maintenant à une petite observation : Lorsque l'on freine l'axe du moteur, par exemple avec le doigt, on sent que le moteur insiste et essaye de repousser cette force exercée sur son axe. Cela est du au courant qui le traverse et qui augmente car le moteur, pour continuer de tourner à la même vitesse, doit fournir plus de couple. Hors, le couple et le courant sont liés : si l'un des deux augmente alors l'autre également. Autrement dit, pour avoir plus de couple le moteur consomme plus de courant. Si votre alimentation est en mesure de le fournir, il pourra éventuellement bouger, sinon, comme il ne peut pas consommer plus que ce qu'on lui donne, il restera bloqué et consommera le maximum de courant fourni.

[[attention]] |Si vous faites circuler trop de courant dans un moteur pour trop longtemps, il va chauffer. Les moteurs sont des composants sans protection. Même s'ils chauffent ils ne feront rien pour s'arrêter, bien au contraire. Cela peut mener à une surchauffe et une destruction du moteur (les bobines à l'intérieur sont détruites). Attention donc à ne pas trop le faire forcer sur de longues périodes continues.

7.1.2 Alimenter un moteur

Bon, et si nous voyions un peu comment cela se passe dans la pratique ? Je vais vous montrer comment alimenter les moteurs électriques à courant continu. Vous allez voir que ce n'est pas aussi simple que ça en a l'air, du moins lorsque l'on veut faire quelque chose de propre. Vous allez comprendre de quoi je parle...

7.1.2.1 Connecter un moteur sur une source d'énergie : la pile

Faisons l'expérience la plus simple qui soit : celle de connecter un moteur aux bornes d'une pile de 9V :

[[question]] |C'est tout ? o_O

Ben oui, quoi de plus ? Le moteur est connecté, son axe tourne, la pile débite du courant... Ha ! Voilà ce qui nous intéresse dans l'immédiat : la pile débite du courant. Oui et pas des moindres car les moteurs électriques sont bien généralement de véritables gloutons énergétiques. Si vous avez la chance de posséder un ampèremètre, vous pouvez mesurer le courant de consommation de votre moteur. En général, pour un petit moteur de lecteur CD on avoisine la centaine de milliampères. Pour un moteur un peu plus gros, tel qu'un moteur de modélisme, on trouve plusieurs centaines de milliampères de consommation. Pour des moteurs encore plus gros, on peut se retrouver avec des valeurs dépassant largement l'ampère voire la dizaine d'ampères. Revenons à notre moteur.

1. Les documents ci-dessous sont des extraits du tutoriel « Arduino : premiers pas en informatique embarquée » réalisé par `Eskimon` et `olite` pour le site `https://zestedesavoir.com/`. Le tutoriel complet est disponible à l'adresse `https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/`.

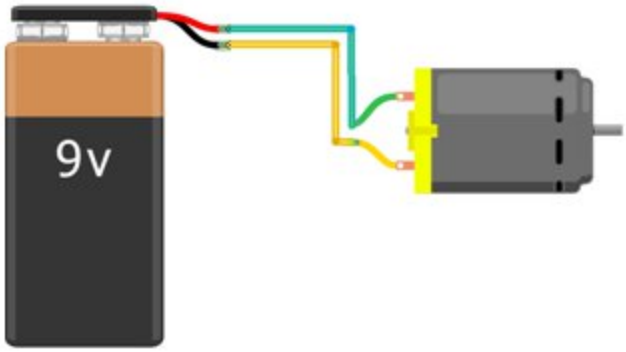


Figure 7.23 – C'est beau, ça tourne.

Lui ne consomme pas plus de 100mA à vide. Mais pour une simple pile c'est beaucoup. Et je vous garantis qu'elle ne tiendra pas longtemps comme ça ! De plus, la vitesse n'est pas réglable, le moteur tourne toujours à son maximum (si c'est un moteur fait pour tourner à 9V). Enfin, pour allumer ou arrêter le moteur, vous êtes obligé de le connecter ou le déconnecter de la pile. En somme, utiliser un moteur dans cette configuration, par exemple pour faire avancer votre petit robot mobile, n'est pas la solution la plus adaptée.

7.1.2.2 Avec la carte Arduino

Vous vous doutez bien que l'on va utiliser la carte Arduino pour faire ce que je viens d'énoncer, à savoir commander le moteur à l'allumage et à l'extinction et faire varier sa vitesse.

[[attention]] | **Ne faites surtout pas le montage qui suit, je vous expliquerai pourquoi !**

Admettons que l'on essaie de brancher le moteur sur une sortie de l'Arduino :

Avec le programme adéquat, le moteur va tourner à la vitesse que l'on souhaite, si l'on veut, réglable par potentiomètre et s'arrêter ou démarrer quand on le lui demande. C'est mieux. C'est la carte Arduino qui pilote le moteur. Malheureux ! Vous ne croyez tout de même pas que l'on va se contenter de faire ça ?! Non, oulaaaa. C'est hyper ultra dangereux... pour votre carte Arduino ! Il est en effet impensable de réaliser ce montage car les moteurs à courant continu sont de véritables sources de parasites qui pourraient endommager, au point de vue matériel, votre carte Arduino ! Oubliez donc tout de suite cette idée de connecter directement le moteur sur une sortie de votre Arduino. Les moteurs, quand ils tournent, génèrent tout un tas de parasites qui peuvent être des surtensions très grandes par rapport à leur tension d'alimentation. De plus, le courant qu'ils demandent est bien trop grand par rapport à ce que peut fournir une sortie numérique d'une carte Arduino (environ 40 mA). Ce sont deux bonnes raisons de ne pas faire le montage précédent.

[[question]] |Mais alors, on fait comment si on peut pas piloter un moteur avec notre carte Arduino ?

Je n'ai pas dis que l'on ne pouvait pas piloter un moteur avec une carte Arduino. J'ai bien précisé *dans cette configuration*. Autrement dit, il faut faire quelque chose de plus pour pouvoir mener à terme cet objectif.

7 Le mouvement grâce aux moteurs

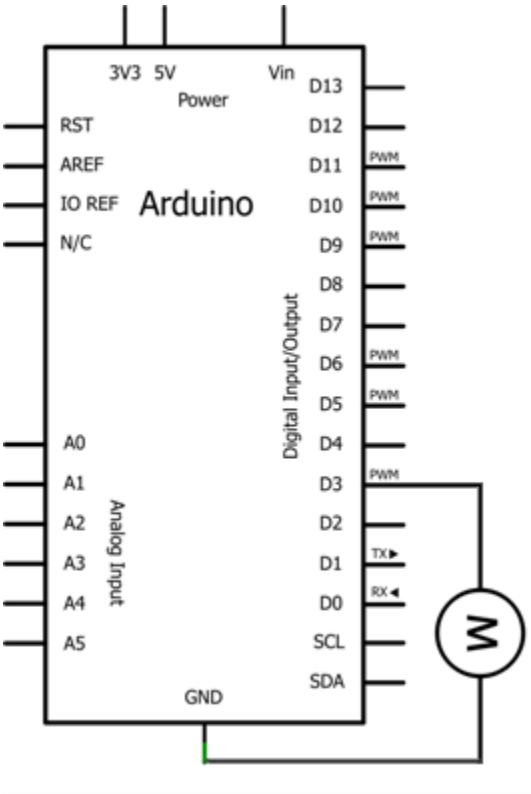


Figure 7.24 – Moteur branché sur une sortie de l'Arduino

7.1.2.3 Une question de puissance : le transistor

Souvenez-vous, nous avons parlé d'un composant qui pourrait convenir dans ce chapitre. Il s'agit du **transistor**. Si vous vous souvenez de ce que je vous avais expliqué, vous devriez comprendre pourquoi je vous en parle ici. Car, à priori, on ne veut pas allumer un afficheur 7 segments. ^^ En fait, le transistor (bipolaire) est comme un interrupteur que l'on commande par un courant. Tout comme on avait fait avec les afficheurs 7 segments, on peut allumer, saturer ou bloquer un transistor pour qu'il laisse passer le courant ou non. Nous avions alors commandé chaque transistor pour allumer ou éteindre les afficheurs correspondants. Essayons de faire de même avec notre moteur :

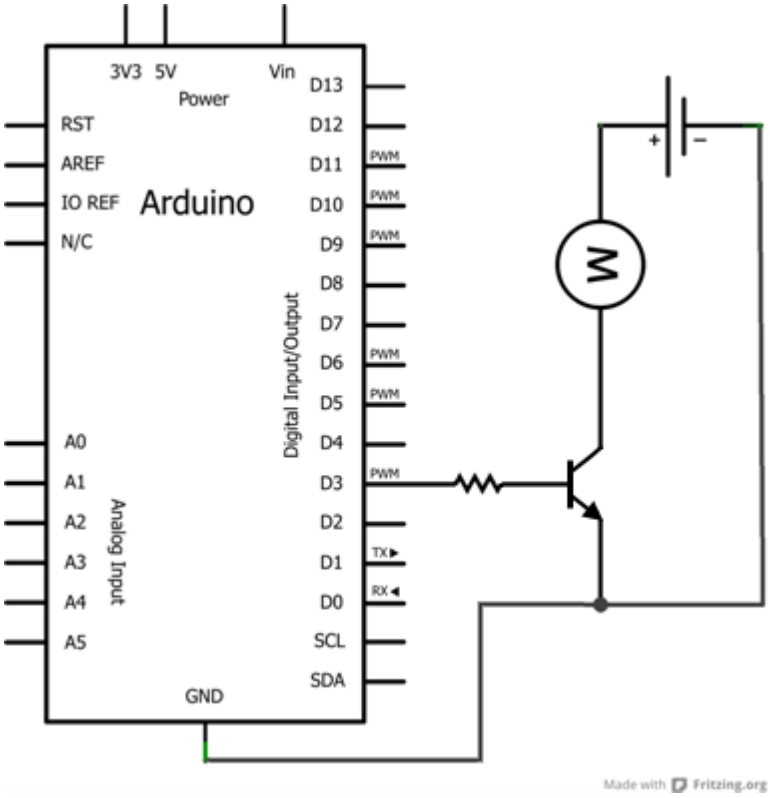


Figure 7.25 – Avec un transistor

Ici, le transistor est commandé par une sortie de la carte Arduino via la résistance sur la base. Lorsque l'état de la sortie est au niveau 0, **le transistor est bloqué et le courant ne le traverse pas**. Le moteur ne tourne pas. Lorsque la sortie vaut 1, le transistor est commandé et devient saturé, c'est-à-dire qu'il laisse passer le courant et le moteur se met à tourner. Le problème, c'est que tout n'est pas parfait et ce transistor cumule des inconvénients qu'il est bon de citer pour éviter d'avoir de mauvaises surprises :

— parcouru par un grand courant, il chauffe et peut être amené à griller s'il n'est pas refroidi

— il est en plus sensible aux parasites et risque d'être endommagé
— enfin, il n'aime pas les "hautes" tensions

Pour répondre à ces trois contraintes, trois solutions. La première consisterait à mettre un transistor qui accepte un courant assez élevé par rapport à la consommation réelle du moteur, ou bien d'adjoindre un *dissipateur* sur le transistor pour qu'il refroidisse. La deuxième solution concernant les parasites serait de mettre un condensateur de filtrage. On en a déjà parlé avec les boutons poussoirs . Pour le dernier problème, on va voir que l'on a besoin d'une diode.

7.1.2.3.1 Le "bon" transistor Comme je viens de vous l'expliquer, il nous faut un transistor comme "interface" de puissance. C'est lui qui nous sert d'interrupteur pour laisser passer ou non le courant. Pour l'instant, nous avons beaucoup parlé des transistors "bipolaires". Ils sont sympas, pas chers, mais il y a un problème : ils ne sont pas vraiment faits pour faire de la commutation, mais plutôt pour faire de l'amplification de courant. Le courant qu'il laisse passer est proportionnel au courant traversant sa base. Pour les petits montages comme celui des 7 segments ce n'est pas vraiment un problème, car les courants sont faibles. Mais pour des montages avec un moteur, où les courants sont bien plus élevés, votre transistor bipolaire va commencer à consommer. On retrouvera jusqu'à plusieurs volts de perdus entre son émetteur et son collecteur, autant de volts qui ne profiteront pas à notre moteur.

[[question]] |Mais alors on fait comment pour pas perdre tout ça ?

Eh bien c'est facile ! On change de transistor ! L'électronique de puissance a donné naissance à d'autres transistors, bien plus optimaux pour les questions de fonctionnement à fort courant et en régime saturé/bloqué. Ce sont les transistors MOSFET (appelés aussi "transistor à effet de champ"). Leur symbole est le suivant :

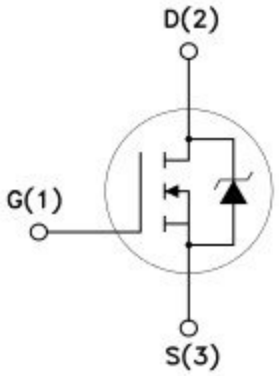


Figure 7.26 – Symbole du transistor MOSFET (canal N)

Il ressemble évidemment à un bipolaire, cela reste un transistor. Par contre il est fait pour faire de l'amplification de tension. Autrement dit, sa broche de commande (que l'on appelle "Gate") doit recevoir une commande, une tension, donc plus besoin de résistance entre Arduino et le transistor. Son fonctionnement est simple : une différence de potentiel sur la gate et il commute (laisse passer le courant entre D (Drain) et S (Source)) sinon il bloque le courant. Facile non ? Un inconvénient cependant : ils coûtent plus chers que leurs homologues bipolaires (de un à plusieurs euros selon le modèle, le courant qu'il peut laisser passer et la tension qu'il peut bloquer). Mais

en contrepartie, ils n'auront qu'une faible chute de tension lorsqu'ils laissent passer le courant pour le moteur, et ça ce n'est pas négligeable. Il existe deux types de MOSFET, le *canal N* et le *canal P*. Ils font la même chose, mais le comportement est inversé (quand un est passant l'autre est bloquant et vice versa). Voici un schéma d'exemple de branchement (avec une résistance de pull-down, comme ça si le signal n'est pas défini sur la broche Arduino, le transistor sera par défaut bloqué et donc le moteur ne tournera pas) :

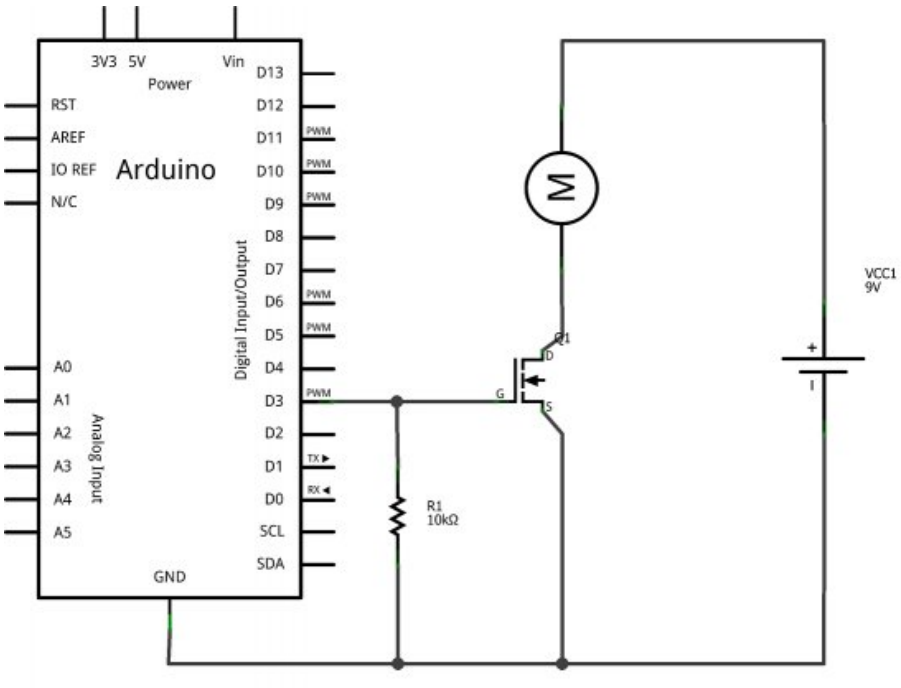


Figure 7.27 – Schéma simple de branchement

7.1.2.4 Protégeons l'ensemble : la diode de roue libre

Une diode, qu'est-ce que c'est ? Nous en avons déjà parlé à vrai dire, il s'agissait des diodes électroluminescentes (LED) mais le principe de fonctionnement reste le même sans la lumière. Une diode, dont voici le symbole :

...est un composant électronique qui ne laisse passer le courant que dans un sens (cf. ce chapitre). Vos souvenirs sont-ils à nouveau en place ? Alors, on continue ! Reprenons le schéma précédent avec le transistor piloté par l'Arduino et qui commande à son tour le moteur. Saturons le transistor en lui appliquant une tension sur sa base. Le moteur commence à tourner puis parvient à sa vitesse de rotation maximale. Il tourne, il tourne et là... je décide de couper l'alimentation du moteur en bloquant le transistor. Soit. Que va-t-il se passer ?

[[question]] |Le moteur va continuer de tourner à cause de son inertie !

7 Le mouvement grâce aux moteurs

Figure 7.28 – Symbole d'une diode

Très bien. Et que cela va t-il engendrer ? Une tension aux bornes du moteur. En effet, je l'ai dit plus tôt, un moteur est aussi un générateur électrique car il est capable de convertir de l'énergie mécanique en énergie électrique même si son rôle principal est de faire l'inverse. Et cette tension est très dangereuse pour le transistor, d'autant plus qu'elle est très haute et peut atteindre plusieurs centaines de Volts (phénomène physique lié aux bobines internes du moteur qui vont se charger). En fait, le moteur va générer une tension à ses bornes et un courant, mais comme le transistor bloque la route au courant, cette tension ne peut pas rester la même et est obligée d'augmenter pour conserver la relation de la loi d'Ohm. Le moteur arrive à un phénomène de **charge**. Il va, précisément, se charger en tension. Je ne m'étends pas plus sur le sujet, il y a bien d'autres informations plus complètes que vous pourrez trouver sur internet. La question : comment faire pour que le moteur se décharge et n'atteigne pas des tensions de plusieurs centaines de Volts à ses bornes (ce qui forcerait alors le passage au travers du transistor et détruirait ce dernier) ? La réponse : par l'utilisation d'une diode. Vous vous en doutiez, n'est-ce pas ? ;) Il est assez simple de comprendre comment on va utiliser cette diode, je vous donne le schéma. Les explications le suivent :

Reprenons au moment où le moteur tourne. Plus de courant ne circule dans le transistor et la seule raison pour laquelle le moteur continue de tourner est qu'il possède une inertie mécanique. Il génère donc cette fameuse tension qui est orientée vers l'entrée du transistor. Comme le transistor est bloqué, le courant en sortie du moteur va donc aller traverser la diode pour revenir dans le moteur. C'est bien, car la tension induite (celle qui est générée par le moteur) restera proche de la tension d'alimentation du moteur et n'ira pas virevolter au voisinage des centaines de Volts. Mais ça ne s'arrête pas là. Pour ceux qui l'auraient remarqué, la tension induite par le moteur est opposée à celle que fournit l'alimentation de ce dernier. Or, étant donné que maintenant on fait un bouclage de la tension induite sur son entrée (vous me suivez toujours ?), eh bien cela alimente le moteur. Les deux tensions s'opposent et cela a pour effet de ralentir le moteur. La **diode de roue libre**, c'est comme ça qu'on l'appelle, sert donc à deux choses : d'une part elle protège le transistor de la surtension induite par le moteur, d'autre part elle permet au moteur de "s'autofreiner".

[[question]] |Et on met quoi comme diode ? o_O

Excellente question, j'allais presque oublier ! La diode que nous mettrons sera une diode *Schottky*. Ne vous laissez pas impressionner par ce nom barbare qui signifie simplement que la diode est capable de basculer (passer de l'état bloquant à passant) de manière très rapide. Dès lors qu'il y a une surtension engendrée par le moteur lorsque l'on le coupe de l'alimentation, la diode va l'absorber aussitôt avant que le transistor ait le temps d'avoir des dommages. On pourra également

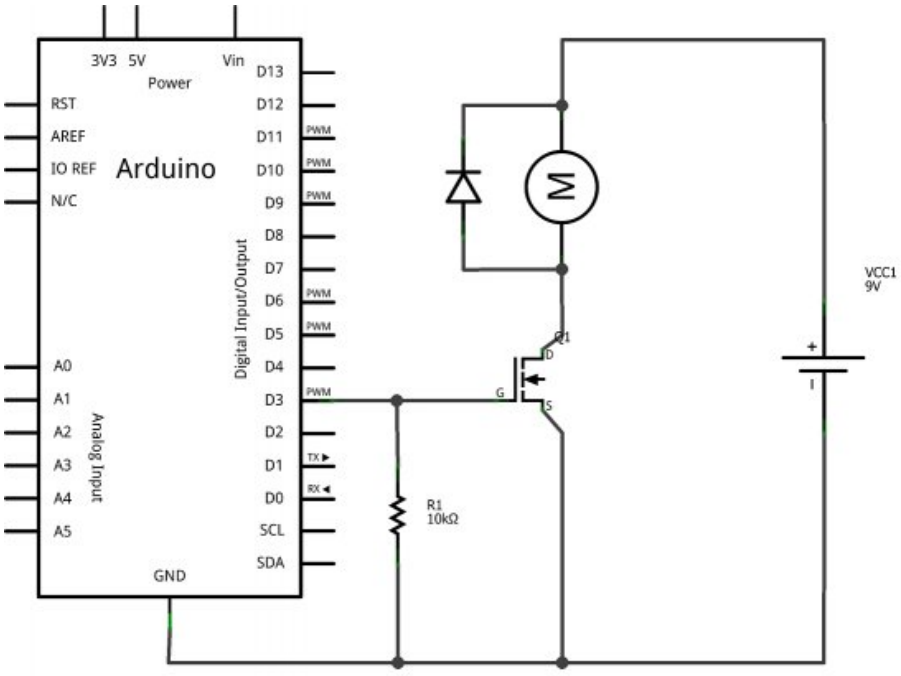


Figure 7.29 – Schéma d'utilisation de la diode

rajouter aux bornes de la diode un condensateur de déparasitage pour protéger le transistor et la diode contre les parasites. Au final, le schéma ressemble à ça :

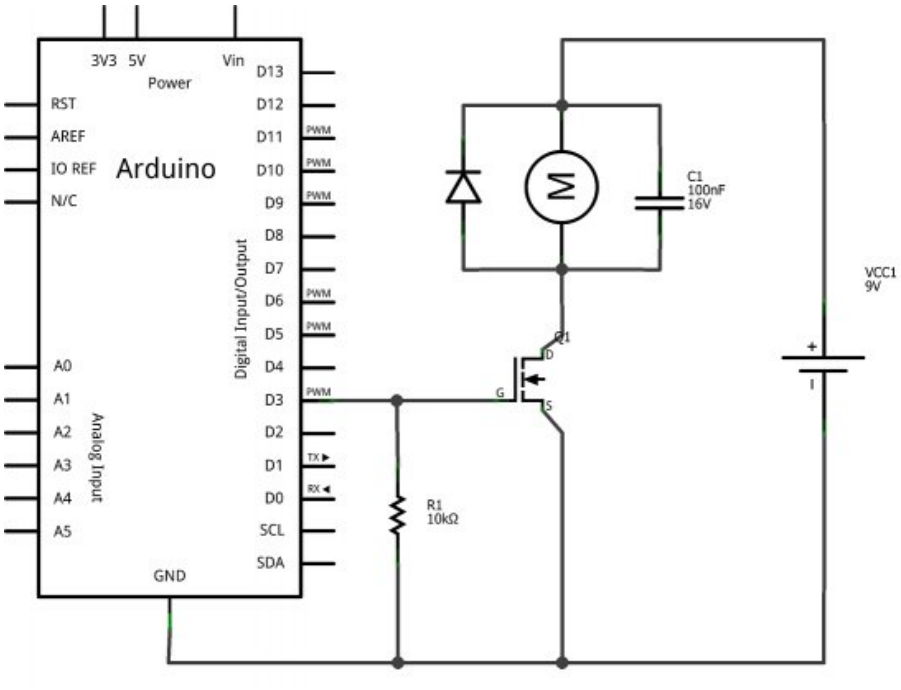


Figure 7.30 – Schéma du montage complet du moteur CC

Sa valeur devra être comprise entre 1nF et 100nF environ. Le but étant de supprimer les petits parasites (pics de tension). Bon, nous allons pouvoir attaquer les choses sérieuses ! :Pirate :

7.1.3 Piloter un moteur

[[information]] |Les montages de cette partie sont importants à connaître. Vous n'êtes pas obligé de les mettre en œuvre, mais si vous le voulez (et en avez les moyens), vous le pouvez. Je dis ça car la partie suivante vous montrera l'existence de shields dédiés aux moteurs à courant continu, vous évitant ainsi quelques maux de têtes pour la réalisation des schémas de cette page. ^^

7.1.3.1 Faire varier la vitesse : la PWM

Maintenant que nous avons les bases fondamentales pour faire tourner notre moteur sans tout faire griller (:roll :), nous allons pouvoir acquérir d'autres connaissances. À commencer par quelque chose de facile : le réglage de la vitesse de rotation du moteur. Comme nous l'expliquions dans le premier morceau de ce chapitre, un moteur à courant continu possède une relation directe entre sa tension d'alimentation et sa vitesse de rotation. En effet, plus la tension à ses bornes est élevée et plus son axe tournera rapidement (dans la limite de ses caractéristiques évidemment).

C Annexe 3 - Pilotage en vitesse d'un moteur à courant continu

C.1 Faire varier la vitesse

rajouter aux bornes de la diode un condensateur de déparasitage pour protéger le transistor et la diode contre les parasites. Au final, le schéma ressemble à ça :

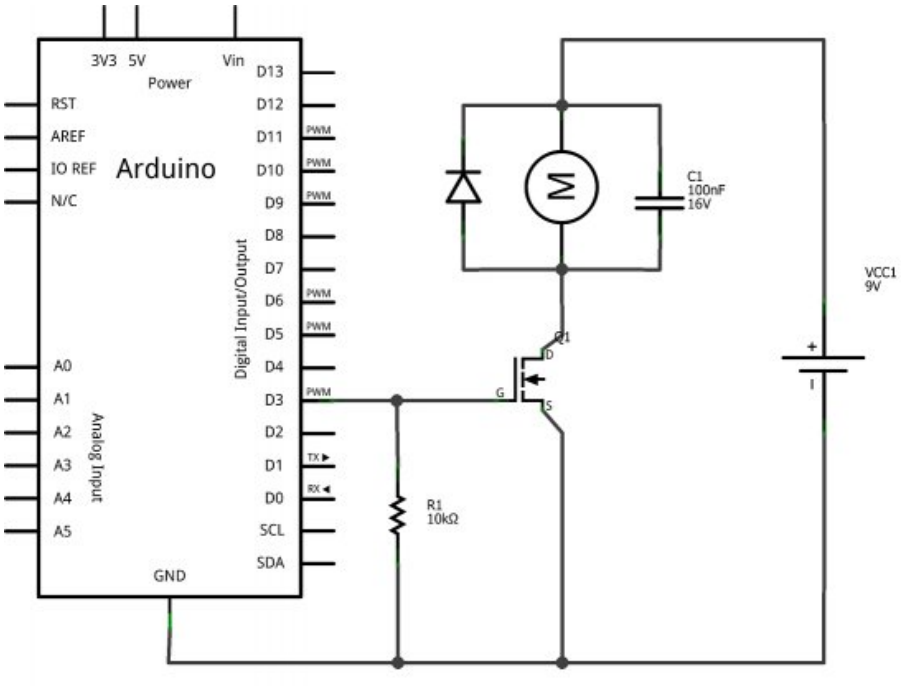


Figure 7.30 – Schéma du montage complet du moteur CC

Sa valeur devra être comprise entre 1nF et 100nF environ. Le but étant de supprimer les petits parasites (pics de tension). Bon, nous allons pouvoir attaquer les choses sérieuses ! :Pirate :

7.1.3 Piloter un moteur

[[information]] |Les montages de cette partie sont importants à connaître. Vous n'êtes pas obligé de les mettre en œuvre, mais si vous le voulez (et en avez les moyens), vous le pouvez. Je dis ça car la partie suivante vous montrera l'existence de shields dédiés aux moteurs à courant continu, vous évitant ainsi quelques maux de têtes pour la réalisation des schémas de cette page. ^^

7.1.3.1 Faire varier la vitesse : la PWM

Maintenant que nous avons les bases fondamentales pour faire tourner notre moteur sans tout faire griller (:roll :), nous allons pouvoir acquérir d'autres connaissances. À commencer par quelque chose de facile : le réglage de la vitesse de rotation du moteur. Comme nous l'expliquions dans le premier morceau de ce chapitre, un moteur à courant continu possède une relation directe entre sa tension d'alimentation et sa vitesse de rotation. En effet, plus la tension à ses bornes est élevée et plus son axe tournera rapidement (dans la limite de ses caractéristiques évidemment).

Cependant le microcontrôleur d'Arduino n'est capable de produire que des tensions de 0 ou 5V. En revanche, il peut "simuler" des tensions variables comprises entre 0 et 5V. Encore un petit rappel de cours nécessaire sur la PWM que nous avons déjà rencontrée ici pour vous rafraichir la mémoire. Nous sommes en mesure de produire à l'aide de notre microcontrôleur un signal carré dont le rapport cyclique est variable. Et grâce à cela, nous obtenons une tension moyenne (comprise entre 0 et 5V) en sortie de la carte Arduino. Il faut juste bien penser à utiliser les sorties adéquates, à savoir : 3, 5, 6, 9, 10 ou 11 (sur une duemilanove/UNO). Je résume : en utilisant la PWM, on va générer une tension par impulsions plus ou moins grandes. Ce signal va commander le transistor qui va à son tour commander le moteur. Le moteur va donc être alimenté par intermittences à cause des impulsions de la PWM. Ce qui aura pour effet de modifier la vitesse de rotation du moteur.

[[question]] |Mais, si le moteur est coupé par intermittences, il va être en rotation, puis va s'arrêter, puis va recommencer, etc. Ce sera pas beau et ça ne tournera pas moins vite. Je comprends pas trop ton histoire. o_O

Non, puisque le moteur garde une inertie de rotation et comme la PWM est un signal qui va trop vite pour que le moteur ait le temps de s'arrêter puis de redémarrer, on va ne voir qu'un moteur qui tourne à une vitesse réduite. Finalement, nous allons donc pouvoir modifier la vitesse de rotation de notre moteur en modifiant le rapport cyclique de la PWM. Plus il est faible (un état BAS plus long qu'un état HAUT), plus le moteur ira doucement. Inversement, plus le rapport cyclique sera élevé (état HAUT plus long que l'état BAS), plus le moteur ira vite. Tout cela couplé à un transistor pour faire passer de la puissance (et utiliser la tension d'utilisation adaptée au moteur) et nous pouvons faire tourner le moteur à la vitesse que nous voulons. Génial non ? Pour l'instant je ne vous ferai pas de démo (vous pouvez facilement imaginer le résultat), mais cela arrivera très prochainement lors de l'utilisation de l'Arduino dans la prochaine sous-partie. Le montage va être le même que tout à l'heure avec le "nouveau" transistor et sa résistance de base :

Maintenant que le moteur tourne à une vitesse réglable, il pourra être intéressant de le faire tourner aussi dans l'autre sens (si jamais on veut faire une marche arrière, par exemple, sur votre robot), voire même d'être capable de freiner le moteur. C'est ce que nous allons tout de suite étudier dans le morceau suivant en parlant d'un composant très fréquent dans le monde de la robotique : le **pont en H**.

7.1.3.2 Tourner dans les deux sens : le pont en H

Faire tourner un moteur c'est bien. Tourner à la bonne vitesse c'est mieux. Aller dans les deux sens c'est l'idéal. C'est donc ce que nous allons maintenant chercher à faire !

7.1.3.2.1 Découverte du pont en H Tout d'abord une question très simple : pourquoi le moteur tourne dans un seul sens ? Réponse évidente : parce que le courant ne va que dans un seul sens ! Pour pouvoir aller vers l'avant ET vers l'arrière il nous faut donc un dispositif qui serait capable de faire passer le courant dans le moteur dans un sens ou dans l'autre. Vous pouvez faire l'expérience en reprenant le premier montage de ce chapitre où il n'y avait que le moteur connecté sur une pile de 9V. Essayez d'inverser les deux bornes du moteur (ça ne risque rien ;)) pour observer ce qu'il se passe : le moteur change de sens de rotation. C'est dû au champ magnétique créé par les bobines internes du moteur qui est alors opposé. Reprenons notre dispositif de base avec un transistor (que nous symboliserons ici par un interrupteur). Si ce dernier est activé le moteur tourne, sinon le moteur est arrêté. Jusque là rien de nouveau. Rajoutons un deuxième transistor

C.2 La PWM (ou MLI) [2]

cyclique du signal est donc le pourcentage de temps de la période durant lequel le signal est au niveau logique 1. En somme, cette image extraite de la documentation officielle d'Arduino nous montre quelques exemples d'un signal avec des rapports cycliques différents :

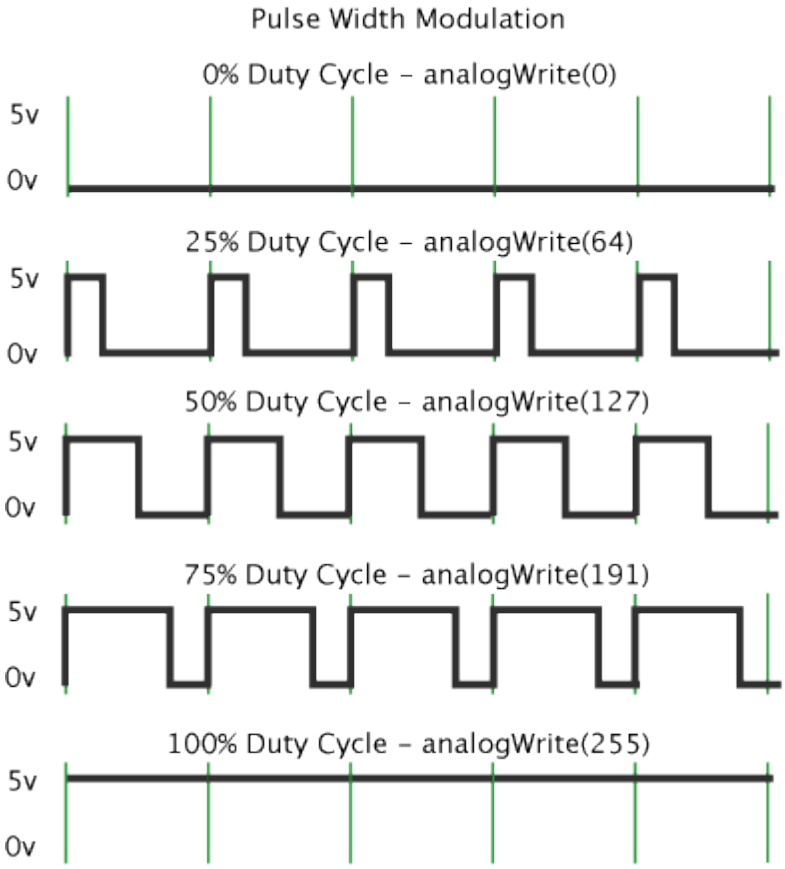


Figure : Des signaux aux rapports cycliques différents - (CC-BY-SA - Timothy Hirzel)

[[i]] | Astuce : Rapport cyclique ce dit **Duty Cycle** en anglais.

Ce n'est pas tout ! Après avoir généré ce signal, il va nous falloir le transformer en signal analogique. Et oui ! Pour l'instant ce signal est encore constitué d'états logiques, on va donc devoir le transformer en extrayant sa *valeur moyenne*... Je ne vous en dis pas plus, on verra plus bas ce que cela signifie.

5.3.2 La PWM de l'Arduino

5.3.2.1 Avant de commencer à programmer

5.3.2.1.1 Les broches de la PWM Sur votre carte Arduino, vous devriez disposer de 6 broches qui sont compatibles avec la génération d'une PWM. Elles sont repérées par le symbole

2. PWM : « Pulse With Modulation » et MLI : « Modulation de largeur d'impulsion »

tilde ~ . Voici les broches générant une PWM : 3, 5, 6, 9, 10 et 11.

5.3.2.1.2 La fréquence de la PWM Cette fréquence, je le disais, est fixe, elle ne varie pas au cours du temps. Pour votre carte Arduino elle est de environ 490Hz.

5.3.2.1.3 La fonction analogWrite() Je pense que vous ne serez pas étonné si je vous dis que Arduino intègre une fonction toute prête pour utiliser la PWM ? Plus haut, je vous disais ceci :

> la PWM est un signal de fréquence fixe qui a un rapport cyclique qui varie avec le temps suivant "les ordres qu'elle reçoit" Source : Moi :P

C'est sur ce point que j'aimerais revenir un instant. En fait, les ordres dont je parle sont les paramètres passés dans la fonction qui génère la PWM. Ni plus ni moins. Étudions maintenant la fonction permettant de réaliser ce signal : `analogWrite()`. Elle prend deux arguments :

- Le premier est le numéro de la broche où l'on veut générer la PWM
- Le second argument représente la valeur du rapport cyclique à appliquer. Malheureusement on n'exprime pas cette valeur en pourcentage, mais avec un nombre entier compris entre 0 et 255

Si le premier argument va de soi, le second mérite quelques précisions. Le rapport cyclique s'exprime de 0 à 100 % en temps normal. Cependant, dans cette fonction il s'exprimera de 0 à 255 (sur 8 bits). Ainsi, pour un rapport cyclique de 0% nous enverrons la valeur 0, pour un rapport de 50% on enverra 127 et pour 100% ce sera 255. Les autres valeurs sont bien entendu considérées de manière proportionnelle entre les deux. Il vous faudra faire un petit calcul pour savoir quel est le pourcentage du rapport cyclique plutôt que l'argument passé dans la fonction.

5.3.2.1.4 Utilisation Voilà un petit exemple de code illustrant tout ça :

```
// une sortie analogique sur la broche 6
const int sortieAnalogique = 6;

void setup()
{
    pinMode(sortieAnalogique, OUTPUT);
}

void loop()
{
    // on met un rapport cyclique de 107/255 = 42 %
    analogWrite(sortieAnalogique, 107);
}
```

Code : Exemple simple d'utilisation de la PWM

5.3.2.2 Quelques outils essentiels

Savez-vous que vous pouvez d'ores et déjà utiliser cette fonction pour allumer plus ou moins intensément une LED ? En effet, pour un rapport cyclique faible, la LED va se voir parcourir par

D Annexe 4 - Pilotage du sens de rotation d'un moteur à courant continu

D.1 Tourner dans les deux sens

7 Le mouvement grâce aux moteurs

Cependant le microcontrôleur d'Arduino n'est capable de produire que des tensions de 0 ou 5V. En revanche, il peut "simuler" des tensions variables comprises entre 0 et 5V. Encore un petit rappel de cours nécessaire sur la PWM que nous avons déjà rencontrée ici pour vous rafraichir la mémoire. Nous sommes en mesure de produire à l'aide de notre microcontrôleur un signal carré dont le rapport cyclique est variable. Et grâce à cela, nous obtenons une tension moyenne (comprise entre 0 et 5V) en sortie de la carte Arduino. Il faut juste bien penser à utiliser les sorties adéquates, à savoir : 3, 5, 6, 9, 10 ou 11 (sur une duemilanove/UNO). Je résume : en utilisant la PWM, on va générer une tension par impulsions plus ou moins grandes. Ce signal va commander le transistor qui va à son tour commander le moteur. Le moteur va donc être alimenté par intermittences à cause des impulsions de la PWM. Ce qui aura pour effet de modifier la vitesse de rotation du moteur.

[[question]] |Mais, si le moteur est coupé par intermittences, il va être en rotation, puis va s'arrêter, puis va recommencer, etc. Ce sera pas beau et ça ne tournera pas moins vite. Je comprends pas trop ton histoire. o_O

Non, puisque le moteur garde une inertie de rotation et comme la PWM est un signal qui va trop vite pour que le moteur ait le temps de s'arrêter puis de redémarrer, on va ne voir qu'un moteur qui tourne à une vitesse réduite. Finalement, nous allons donc pouvoir modifier la vitesse de rotation de notre moteur en modifiant le rapport cyclique de la PWM. Plus il est faible (un état BAS plus long qu'un état HAUT), plus le moteur ira doucement. Inversement, plus le rapport cyclique sera élevé (état HAUT plus long que l'état BAS), plus le moteur ira vite. Tout cela couplé à un transistor pour faire passer de la puissance (et utiliser la tension d'utilisation adaptée au moteur) et nous pouvons faire tourner le moteur à la vitesse que nous voulons. Génial non ? Pour l'instant je ne vous ferai pas de démo (vous pouvez facilement imaginer le résultat), mais cela arrivera très prochainement lors de l'utilisation de l'Arduino dans la prochaine sous-partie. Le montage va être le même que tout à l'heure avec le "nouveau" transistor et sa résistance de base :

Maintenant que le moteur tourne à une vitesse réglable, il pourra être intéressant de le faire tourner aussi dans l'autre sens (si jamais on veut faire une marche arrière, par exemple, sur votre robot), voire même d'être capable de freiner le moteur. C'est ce que nous allons tout de suite étudier dans le morceau suivant en parlant d'un composant très fréquent dans le monde de la robotique : le **pont en H**.

7.1.3.2 Tourner dans les deux sens : le pont en H

Faire tourner un moteur c'est bien. Tourner à la bonne vitesse c'est mieux. Aller dans les deux sens c'est l'idéal. C'est donc ce que nous allons maintenant chercher à faire !

7.1.3.2.1 Découverte du pont en H Tout d'abord une question très simple : pourquoi le moteur tourne dans un seul sens ? Réponse évidente : parce que le courant ne va que dans un seul sens ! Pour pouvoir aller vers l'avant ET vers l'arrière il nous faut donc un dispositif qui serait capable de faire passer le courant dans le moteur dans un sens ou dans l'autre. Vous pouvez faire l'expérience en reprenant le premier montage de ce chapitre où il n'y avait que le moteur connecté sur une pile de 9V. Essayez d'inverser les deux bornes du moteur (ça ne risque rien ;)) pour observer ce qu'il se passe : le moteur change de sens de rotation. C'est dû au champ magnétique créé par les bobines internes du moteur qui est alors opposé. Reprenons notre dispositif de base avec un transistor (que nous symboliserons ici par un interrupteur). Si ce dernier est activé le moteur tourne, sinon le moteur est arrêté. Jusque là rien de nouveau. Rajoutons un deuxième transistor

386

"de l'autre côté" du moteur. Rien ne va changer, mais il va falloir commander les deux transistors pour faire tourner le moteur. Ce n'est pas bon. Essayons avec quatre transistors, soyons fou !

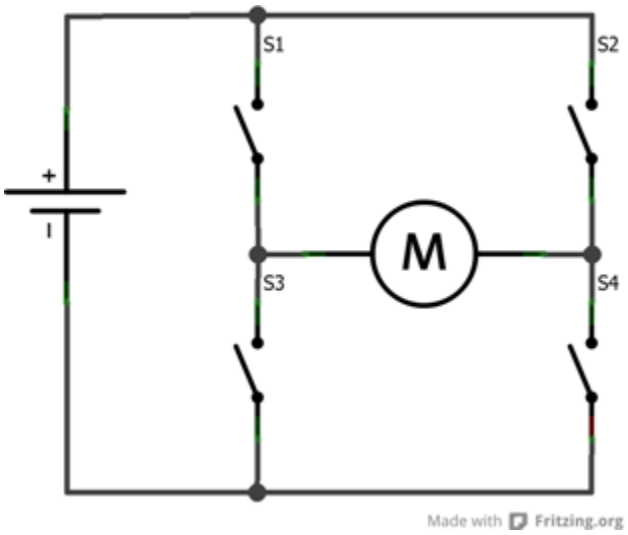


Figure 7.34 – Le pont en H

Eh bien, cela change tout ! Car à présent nous allons piloter le moteur dans les deux sens de rotation. Pour comprendre le fonctionnement de ce pont en H (appelé ainsi par sa forme), imaginons que je ferme les transistors 1 et 4 en laissant ouverts le 2 et le 3. Le courant passe de la gauche vers la droite.

Si en revanche je fais le contraire (2 et 3 fermés et 1 et 4 ouverts), le courant ira dans l'autre sens ! C'est génial non ?

Et ce n'est pas tout !

7.1.3.2.2 Allons plus loin avec le pont en H Comme vous l'aurez sûrement remarqué, les transistors fonctionnent deux par deux. En effet, si on en ferme juste un seul et laisse ouvert les trois autres le courant n'a nulle part où aller et rien ne se passe, le moteur est en roue libre. Maintenant, que se passe-t-il lorsqu'on décide de fermer 1 & 2 en laissant 3 et 4 ouverts ? Cette action va créer ce que l'on appelle un **frein magnétique**. Je vous ai expliqué plus tôt comment cela fonctionnait lorsque l'on mettait une diode de roue libre aux bornes du moteur. Le moteur se retrouve alors court-circuité. En tournant à cause de son inertie, le courant généré va revenir dans le moteur et va le freiner. Attention cependant, c'est différent d'un phénomène de roue libre où le moteur est libre de tourner.

[[attention]] | Ne fermez **jamais** 1 & 3 et/ou 2 & 4 ensembles, cela ferait un court-circuit de l'alimentation et vos transistors risqueraient de griller immédiatement si l'alimentation est capable de fournir un courant plus fort que ce qu'ils ne peuvent admettre.

7.1.3.3 Les protections nécessaires

7 Le mouvement grâce aux moteurs

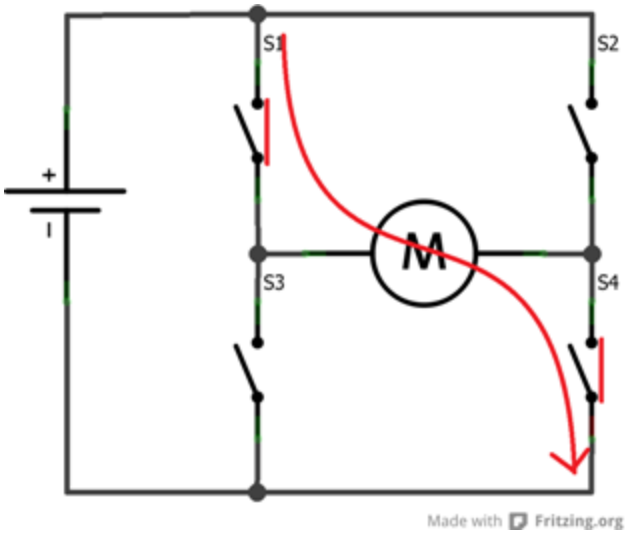


Figure 7.35 – Fonctionnement dans le sens horaire

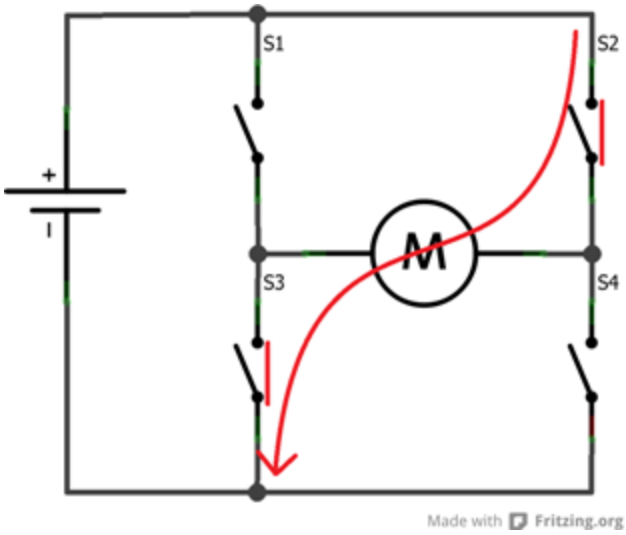
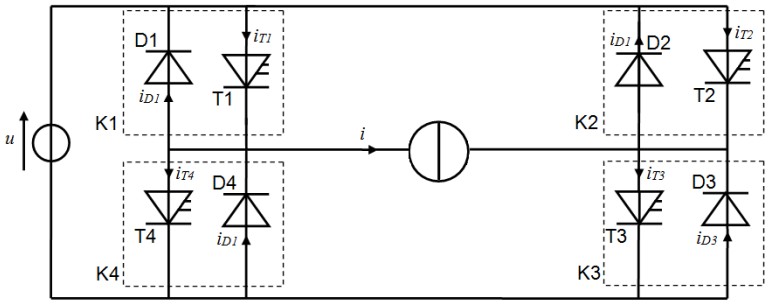


Figure 7.36 – Fonctionnement dans le sens anti-horaire

390

D.2 Montage

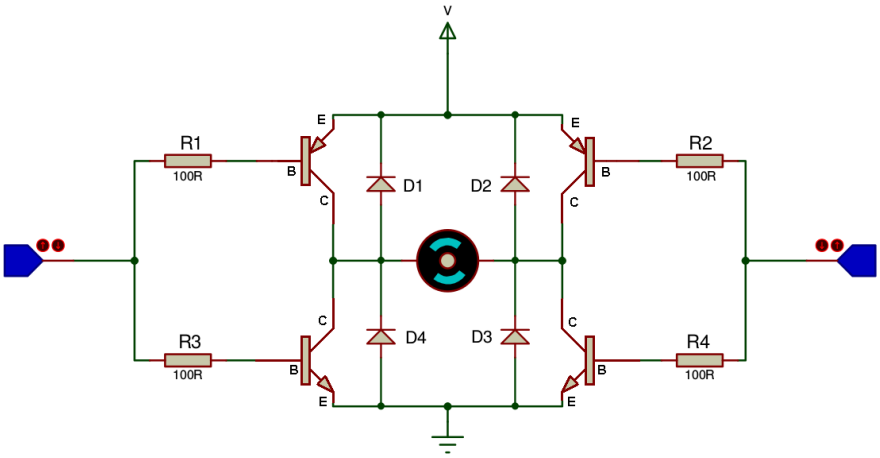
Le montage de base est le même que pour l'onduleur, ce montage est appelé hacheur « à 4 quadrants » :



Les transistors T1 et T3 d'une part, T2 et T4 d'autre part sont ouverts ou fermés en même temps (ils sont synchronisés). Ces deux groupes de transistors sont alternativement ouverts et fermés.

D.3 Mise en œuvre à l'aide d'un module Arduino

Le pilotage des 4 transistors peut s'effectuer à l'aide de signaux logiques issus d'un module Arduino. Pour une mise en œuvre simplifiée, on peut utiliser des transistors MOS ou bipolaires, de type PNP pour les interrupteurs du haut (1 et 2) et NPN pour les deux interrupteurs du bas (3 et 4). On considèrera que leurs comportements sont simplement inversés face à un même signal de commande.



La commande des différents interrupteurs à l'aide d'un module Arduino s'effectue à l'aide de deux sorties analogiques. Pour exemple, le programme suivant permet de changer le sens de rotation du moteur toutes les 3 s.

```
const int com_m1=3;
const int com_m2=5;

// the setup routine runs once when you press reset
void setup()
{
// initialize the digital pin as an output
pinMode(com_m1,OUTPUT);
pinMode(com_m2,OUTPUT);
}

// the loop routine runs over and over again forever
void loop()
{
digitalWrite(com_m2,LOW);
digitalWrite(com_m1,HIGH);
delay(3000);
digitalWrite(com_m1,LOW);
digitalWrite(com_m2,HIGH);
delay(3000);
}
```

moteur_ph/moteur_ph.ino