

[Arduino 6] Le mouvement grâce aux moteurs

S'il y a bien une chose sur laquelle on ne peut pas faire abstraction en robotique, c'est le mouvement. Le mouvement permet d'interagir avec l'environnement, il permet par exemple à un robot de rouler ou de prendre un objet voire même d'exprimer des émotions. Bref, ce mouvement est indispensable. Dans cette partie nous allons apprendre à donner du mouvement à nos applications. Pour cela, nous allons utiliser des moteurs. Ce sont eux qui, principalement, transforment l'énergie qu'ils reçoivent en un mouvement. Nous verrons également comment reconnaître et utiliser les différents types de moteurs.

Chaque chapitre présentera un moteur. Le premier sera à propos du **moteur à courant continu**. Il est **indispensable de le lire** car les deux autres chapitres reprendront des éléments qui seront donc considérés comme acquis (on ne va pas s'amuser à faire des copier/coller 3 fois des mêmes notions 😊)

[Arduino 601] Le moteur à courant continu

Nul doute que vous connaissez l'existence des moteurs car il en existe toute une panoplie ! Le premier qui vous viendra certainement à l'esprit sera le moteur de voiture, ou peut-être celui présent dans une perceuse électrique. Voilà deux exemples d'objets dans lesquels on peut trouver un moteur. Bien entendu, ces deux moteurs sont de type différent, il serait en effet peu probable de faire avancer votre voiture avec un moteur de perceuse électrique... et puis l'utilisation d'une perceuse intégrant un moteur de voiture de plusieurs centaines de kilos serait fastidieuse 😊. Voyons donc comment fonctionne le moteur électrique le plus répandu : le moteur à courant continu...

Un moteur, ça fait quoi au juste ?

Commençons en douceur par l'explication de ce à quoi sert un moteur et son fonctionnement.

Ce chapitre n'est pas un des plus simples car il va faire apparaître des notions de mécanique qui sont indispensables pour comprendre le mouvement. Il prend en général plusieurs heures de cours pour être bien expliqué. Nous allons donc vous faire ici uniquement une introduction à la mécanique du moteur. Cependant, cette introduction présente des notions très importantes pour bien comprendre la suite, ne la négligez donc pas !

Prenons un moteur électrique des plus basiques qui soient :



Vous avez devant vos yeux un moteur électrique tel que l'on peut en trouver dans les engins de modélisme ou dans les voitures téléguidées. Mais sachez qu'il en existe de toute sorte, allant du miniature au gigantesque, adaptés à d'autres types d'applications. Nous nous contenterons ici des moteurs électriques "basiques".

Transformation de l'énergie électrique en énergie mécanique

Un moteur ça fait quoi ? Ça tourne ! On dit qu'un **moteur est un composant de conversion d'énergie électrique en énergie mécanique**. Les moteurs à courant continu (ce terme deviendra plus clair par la suite) transforment l'énergie électrique en énergie mécanique de rotation, pour être précis. Mais ils peuvent également servir de générateur d'électricité en convertissant une énergie mécanique de rotation en énergie électrique. C'est le cas par exemple de la dynamo sur votre vélo !

Ce dernier point n'est pas à négliger, car même si dans la plupart des applications votre moteur servira à générer un mouvement, il sera possible qu'il soit actionné "à l'envers" et génère alors du courant. Il faudra donc protéger votre circuit pour ne pas l'abîmer à cause de cette "injection" d'énergie non désirée. On va revenir dessus plus loin. 😊

Principe de fonctionnement du moteur à courant continu

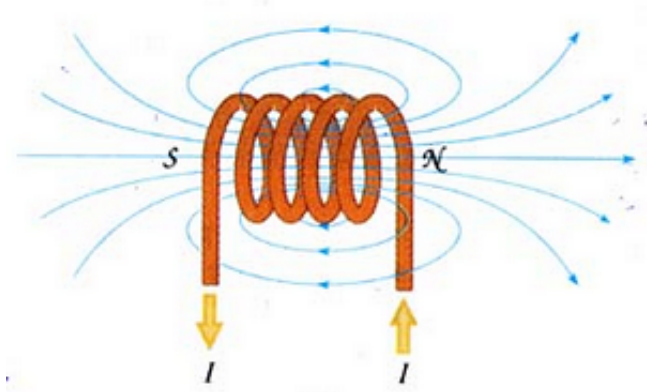
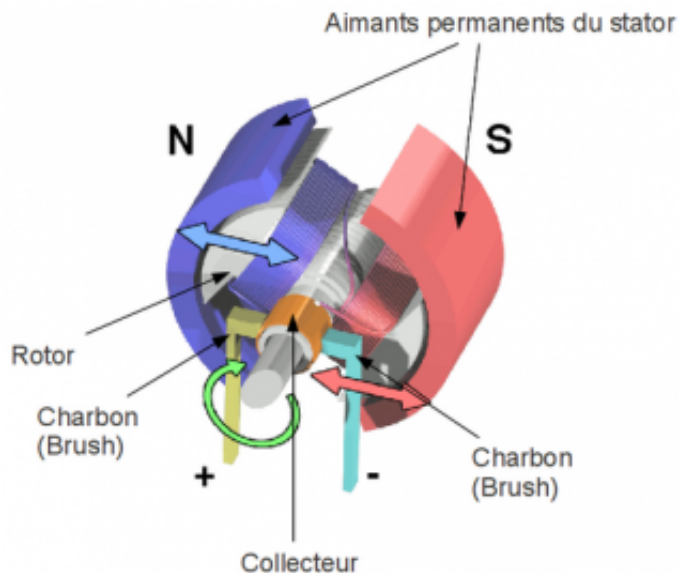
Du vocabulaire

Tout d'abord, nous allons prendre une bonne habitude. Le moteur à courant continu s'appelle aussi "Machine à Courant Continu", que j'abrégerais en MCC. Le moteur à courant continu est composé de deux parties principales : le **rotor** (partie qui tourne) et le **stator** (partie qui ne tourne pas, statique). En électrotechnique le stator s'appelle aussi **inducteur** et le rotor s'appelle l'**induit**. Sur l'image à droite, vous pouvez observer au milieu – entouré par les aimants bleu et rouge qui constituent le stator – le rotor composé de fils de cuivre enroulés sur un support lui même monté sur un axe. Cet axe, c'est l'**arbre** de sortie du moteur. C'est lui qui va transmettre le mouvement à l'ensemble mécanique (pignons, chaîne, actionneur...) qui lui est associé en aval. Dans le cas d'un robot sur roues par exemple, on va mettre la roue sur cet axe, bien souvent par l'intermédiaire d'un réducteur qui diminue la vitesse de rotation tout en augmentant le couple. On verra tout à l'heure pour éclaircir ces termes qui doivent, pour l'instant, ne

pas vous dire grand chose.

De nouvelles bases sur l'électricité

Vous le savez peut-être, lorsque un courant circule dans un fil il génère un champ magnétique. Plus le courant qui circulera dans le fil sera grand, plus l'intensité du champ magnétique sera élevée. Lorsqu'on enroule du fil électrique sur lui même, on forme une **bobine**. Un des avantages de la bobine est que l'on "cumule" ce champ magnétique. Donc plus on a de tours de fil (des **spires**) et plus le champ magnétique sera élevé pour un courant donné.



En somme, on retiendra que lorsque l'on crée une bobine de fil électrique, en général du cuivre, on additionne les champs magnétiques créés par chaque spire de la bobine. Ainsi, vous comprendrez aisément que plus la bobine contient de spires et plus le champ magnétique qu'elle induit est important. Je ne vous ai pas trop perdu, ça va pour le moment ? 😊 Bon, continuons.

Le magnétisme

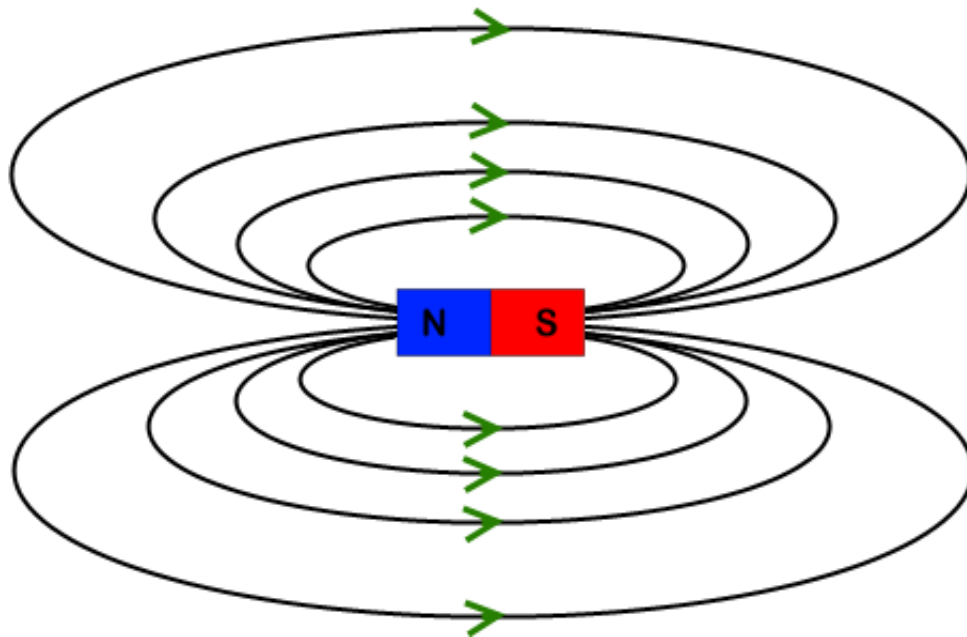
Oui, parlons-en. Ce sera bref, rassurez-vous. Je vais faire appel à votre expérience... avec les aimants. Vous avez tous déjà eu l'occasion d'avoir deux aimants dans la main et d'observer la résistance qu'ils émettent lorsque l'on veut les rapprocher l'un de l'autre, ou au contraire lorsqu'ils s'attirent soudainement dès qu'on les met un peu trop près. Ce phénomène est dû au champ magnétique que génèrent les aimants. Voilà un aimant permanent le plus simple soit-il :



Aimant permanent

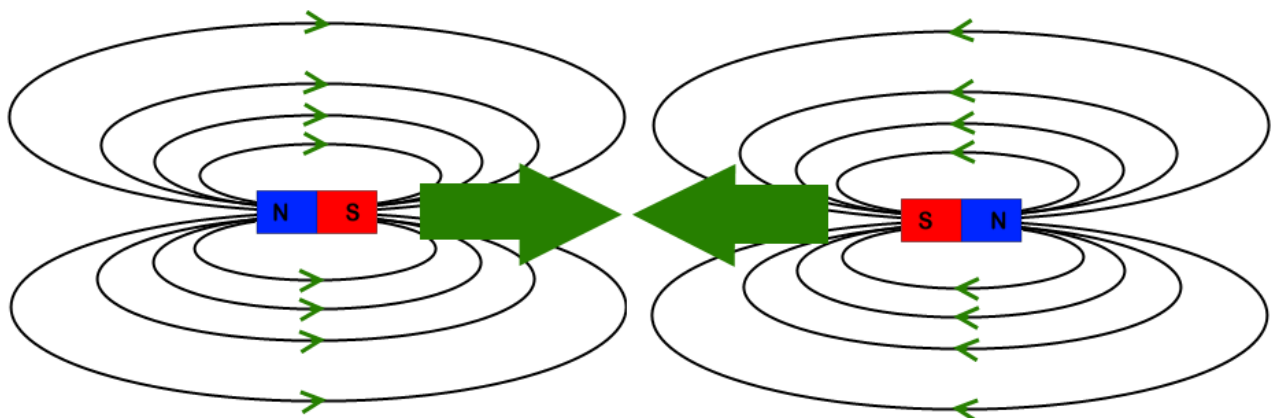
Il possède un pôle **N**ord et un pôle **S**ud. Cet aimant génère un **champ magnétique permanent**, c'est à dire que le champ magnétique est toujours présent. C'est quelque

chose de totalement invisible mais qui permet de faire des choses intéressantes.



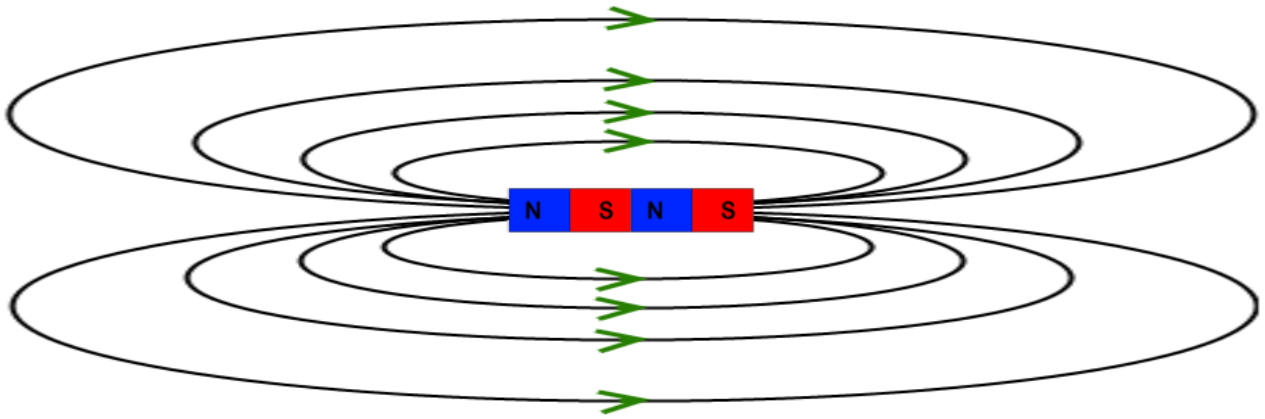
Champ magnétique généré par un aimant permanent

Notez bien que j'ai ajouté des flèches représentatives du sens de parcours du champ magnétique, c'est important pour la suite. Bon, pour terminer mon explication sur le champ magnétique, je vous propose d'imaginer qu'il s'agisse d'un flux invisible, un peu comme le courant. Pour se rapprocher de l'analogie avec l'eau, on peut imaginer aussi que l'aimant est une fontaine qui propulse de l'eau (champ magnétique) et qui la récupère à l'opposé de là où il l'a éjectée. Tout ça, pour en arriver à vous dire qu'approcher deux aimants avec le même pôle, ils se repoussent mutuellement (les deux fontaines éjectent de l'eau l'une contre l'autre, ce qui a pour effet de les repousser). Et on le comprend bien lorsque l'on regarde le sens du champ magnétique :



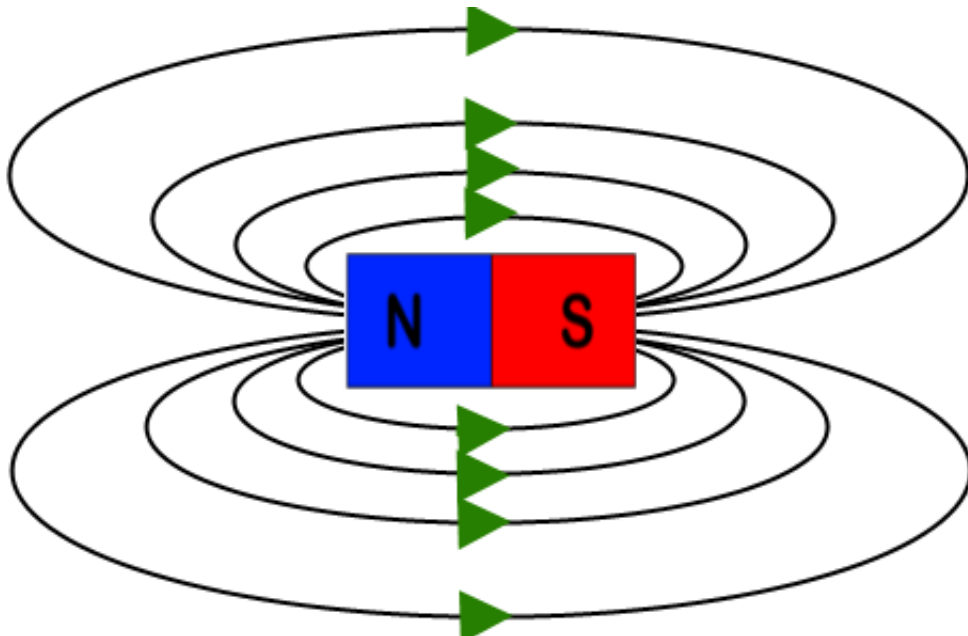
Deux aimants permanents qui se repoussent mutuellement car leur champ magnétique est opposé

En revanche, deux aimants orientés dans le même sens se rapprocheront car leur champ magnétique ira dans le sens opposé. La première "fontaine" va aspirer ce que l'autre éjecte, et l'autre va aspirer ce que la première éjecte.



Résultat de la mise en “série” de deux aimants permanents identiques

Par conséquent, le champ magnétique global sera plus intense. On peut alors schématiser le résultat sous la forme d'un seul aimant plus puissant.



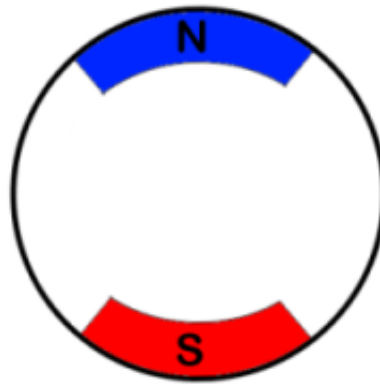
Schématisation du résultat précédent

Ça nous amène où tout ça ? Je comprends mieux comment fonctionne les aimants, mais pour un moteur électrique, c'est pareil ? 🤔

Eh oui, sans quoi mes explications n'auraient eu aucun sens si je vous avais dit qu'un moteur fonctionnait complètement différemment. 😊 Décomposons notre explication en deux parties.

Le stator

Le stator, je l'ai dit au début, est une partie immobile du moteur. Sur l'image, il se trouve sur les côtés contre le châssis. Il forme un aimant avec ses pôles Nord et Sud. Cet ensemble aimant+châssis constitue donc le stator :



Stator d'une MCC

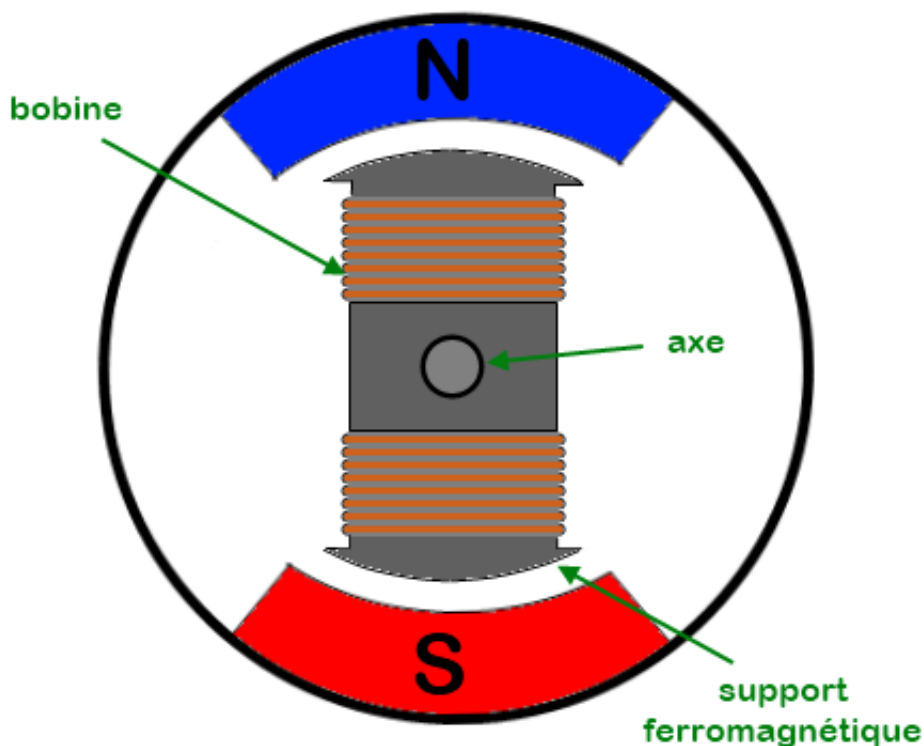
Il n'y a pas plus de choses à dire, l'essentiel du phénomène de rotation créé par un moteur électrique va se jouer dans le rotor.

Le rotor et la mise en mouvement

Le rotor, je le rappelle, est situé au centre du stator. Pour faire très simple, je vous donnerai les explications ensuite, le rotor est la pièce maîtresse qui va recevoir un courant continu et va induire un champ magnétique variable pour mettre en rotation l'arbre du rotor. Si l'on veut, oui, il s'auto-met en rotation. 🤖

Waaho ! Avec du courant continu il arrive à créer un champ magnétique variable ?
o_O

Surprenant n'est-ce pas ? Eh bien, pour comprendre ce qu'il se passe, je vous propose de regarder comment est constitué un rotor de MCC (j'abrège) :

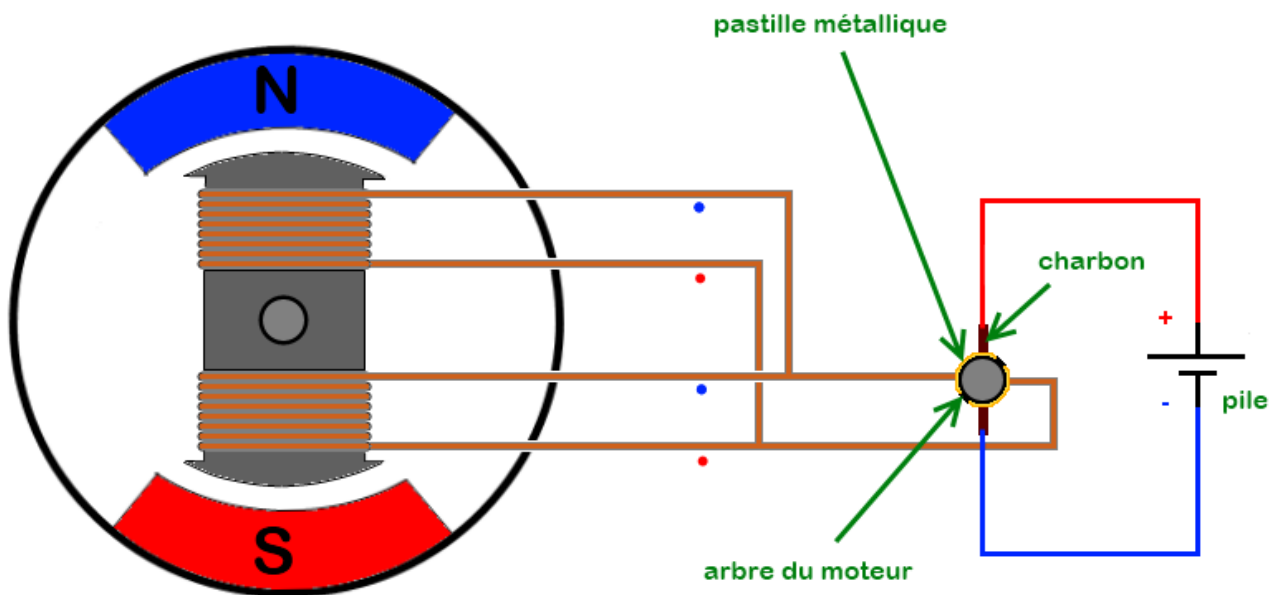


Il s'agit bien d'un schéma de principe, normalement un moteur à courant continu est constitué de trois bobines sur son rotor. Autrement on pourrait obtenir un équilibre qui empêcherait la rotation de l'arbre du moteur, mais surtout le moteur tournerait dans un sens aléatoire. Ce qui n'est pas très adapté quand on veut faire avancer son robot. 🤖

Voilà donc le rotor de notre moteur. Bien, passons à la prati...

Eh oh, attends !! 😬 C'est quoi ces deux bobines, comment on les alimente ? o_O

Ha, j'oubliais presque ! Merci de me l'avoir rappelé. Il y a en effet un élément dont nous n'avons pas encore évoqué l'existence, il s'agit du **collecteur**. Comme son nom le suggère, c'est un élément du moteur qui se situe sur l'arbre de rotation (ou l'axe du moteur si vous préférez) et qui a pour objectif de récupérer le courant afin de l'amener jusqu'aux bobines. On peut faire le schéma complet du moteur avec les bobines et le collecteur :



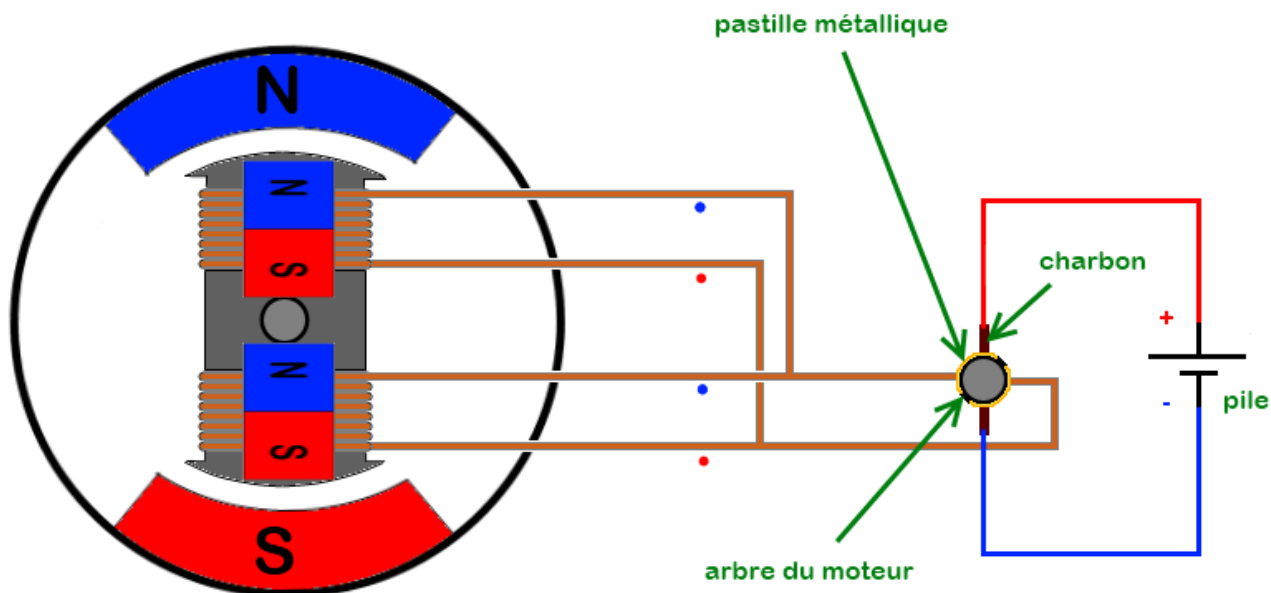
Dites-vous bien qu'il ne s'agit là que d'un schéma de principe simplifié, car je le disais, les moteurs n'ayant que deux bobines n'existent pas.

Le collecteur est représenté ici sur la partie droite de l'image. Il est situé sur l'arbre du moteur (son axe). Ce collecteur est constitué de deux pastilles métalliques auxquelles sont reliées les extrémités des bobines. Le contact électrique entre la pile qui alimente le moteur et les bobines se fait par le collecteur et par des éléments "spéciaux" que l'on appelle les **charbons**. Ces deux éléments servent à amener le courant dans les bobines en faisant un simple contact électrique de toucher. C'est à dire que les charbons frottent sur les pastilles métalliques lorsque le moteur tourne.

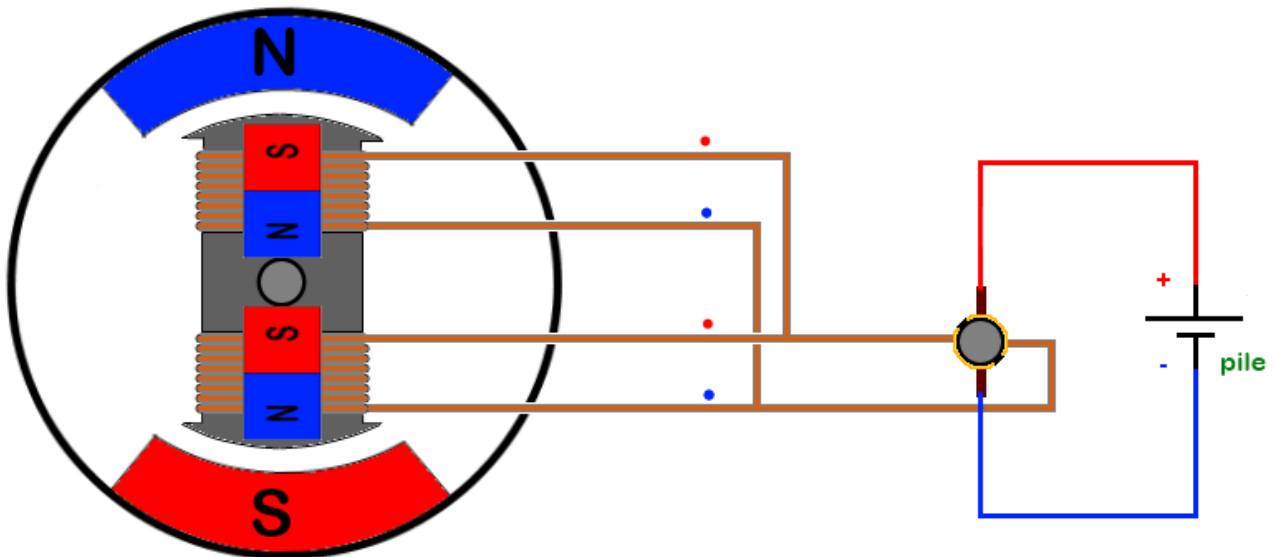
Et y tourne comment ce moteur, on le saura un jour ? 😬

Ça vient, patience. 🤖 Prenons la configuration du moteur tel qu'il est sur l'image précédente. Faites bien attention au sens des bobines, car si elles sont bobinées dans

un sens opposé ou bien si le courant circule dans un sens opposé, le moteur ne tournera pas. J'ai donc pris le soin de mettre un point bleu et rouge, pour indiquer le sens des bobines (vous allez comprendre). Nous y voilà. 😊 Sur le schéma précédent, le pôle positif de la pile est relié, via le collecteur, à l'entrée bleue des deux bobines. Leur sortie, en rouge, est donc reliée, toujours via le collecteur, à la borne négative de la pile. Vous admettrez donc, avec ce que l'on a vu plus haut, qu'il y a un courant qui parcourt chaque bobine et que cela génère un champ magnétique. Ce champ est orienté selon le sens du courant qui circule dans la bobine. Dans un premier temps, on va se retrouver avec un champ magnétique tel que celui-ci :



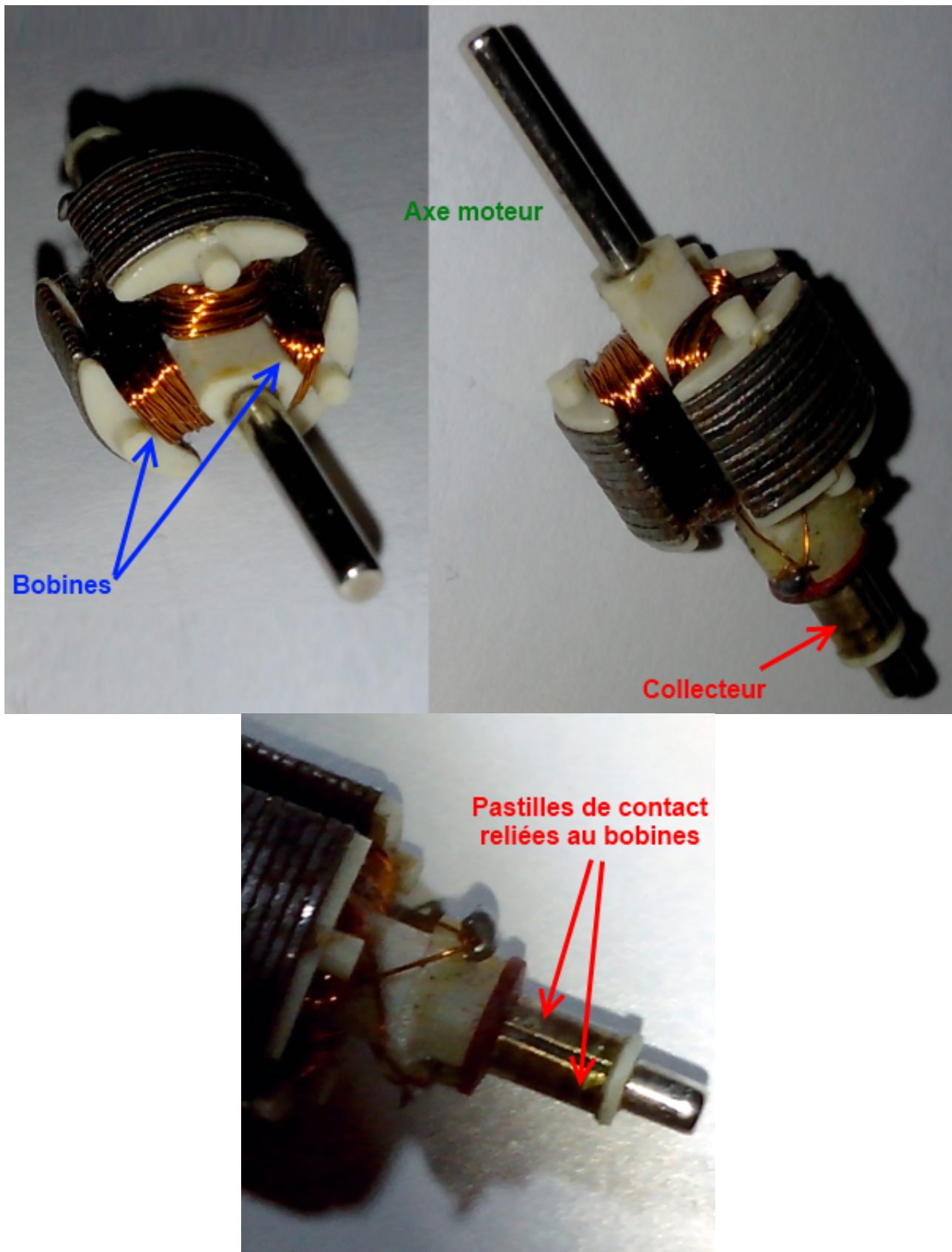
Ce champ va être opposé aux deux aimants permanents du stator du moteur, cela va donc mettre en mouvement l'axe du rotor. Et ce mouvement est défini par le fait que deux aimants orientés par leurs pôles opposés (face nord de l'un face au nord du deuxième, idem pour le sud) se repoussent. Par conséquent, l'axe du moteur, je le disais, va se mettre à tourner jusqu'à ce que les aimants permanents du stator se retrouvent face à chacun de leur complément créé par le champ magnétique des bobines :



ATTENDEEEEEZ ! Ce n'est pas fini ! Non, car dans cette configuration, si rien ne se passe, eh bien... rien ne se passera. 🤖 Et oui, puisque le moteur est arrivé dans une phase de stabilité. En effet, chaque aimant est face au champ magnétique opposé, donc ils s'attirent mutuellement ce qui a pour effet de régir cette situation d'équilibre. L'élément qui va s'opposer à cet équilibre est le branchement des bobines du rotor. Vous ne l'avez peut-être pas remarqué, mais les bobines ne sont plus connectées comme à la situation précédente. Le point rouge des bobines est maintenant relié au pôle positif de la pile et le point bleu au pôle négatif. Le champ magnétique généré par les bobines change alors d'orientation et l'on se retrouve avec des champs opposés. Le moteur est à nouveau en situation de déséquilibre (car les champs magnétiques se repoussent) et cela entraîne un mouvement de rotation de l'axe du moteur. Vous l'aurez compris, ces situations se répètent indéfiniment car **le moteur n'est jamais dans une configuration équilibrée**. C'est cette situation de déséquilibre qui fait que le moteur tourne.

Alors attention, je le répète une dernière fois, un moteur n'ayant que deux bobines comme sur mes schémas ne peut pas fonctionner, car c'est un modèle simplifié qui engendrerait immédiatement une situation équilibrée à la mise sous tension.

Pour vous prouver que ce que je dis est vrai, voilà des photos du rotor d'un moteur à courant continu que j'avais démonté il y a bien, bien, bieeeeen longtemps : 📷



Vous voyez ? Trois bobines et trois pastilles reliées à chacune, sur le collecteur. Bon, je ne vous refais pas les explications, vous êtes capables de comprendre comment cela fonctionne. 😊

La mécanique liée au moteur

A présent, nous allons détailler quelques notions de mécanique liées aux moteurs.

Le couple

Le couple est une notion un peu dure à comprendre, mais on va y arriver ! Partons de son unité. L'unité du couple est le Newton-Mètre (Nm), attention j'ai bien dit Newton-Mètre et non pas Newton ~~par~~ mètre ! Cette unité nous informe de deux choses : le couple est à la fois lié à une distance (le mètre) mais aussi à une force (le Newton). Maintenant je rajoute une information : le couple s'exprime par rapport à un axe. On peut en conclure que le couple est **la capacité du moteur à faire tourner quelque chose sur son axe**. Plus le couple est élevé et plus le moteur sera capable de mettre en mouvement quelque chose de lourd. Exemple : Vous avez peut-être déjà essayé de dévisser un écrou sur une roue de voiture. Vous avez probablement remarqué que plus vous avez une clef avec un bras long (un [effet de levier](#) important) et plus il était facile de faire bouger l'écrou (pour le premier tour, quand il est bien vissé/coincé). Ce phénomène s'explique simplement par le fait que vous avez plus de couple avec un levier long qu'avec un levier court. Et c'est logique ! Si l'on considère que le couple s'exprime en Newton-mètre, le Newton se sera la force de vos muscles (considérée fixe dans notre cas d'étude, sauf si vous vous appelez Hulk) et le mètre sera la longueur du levier. Plus votre levier est grand, plus la distance est élevée, et plus le couple augmente. Ce qui nous permet d'introduire la formule suivante :

$$C = F \times r$$

Avec :

- C : le couple, en Newton-mètre
- F : la force exercée, en Newton
- r : le rayon de l'action (la longueur du levier si vous préférez), en mètre

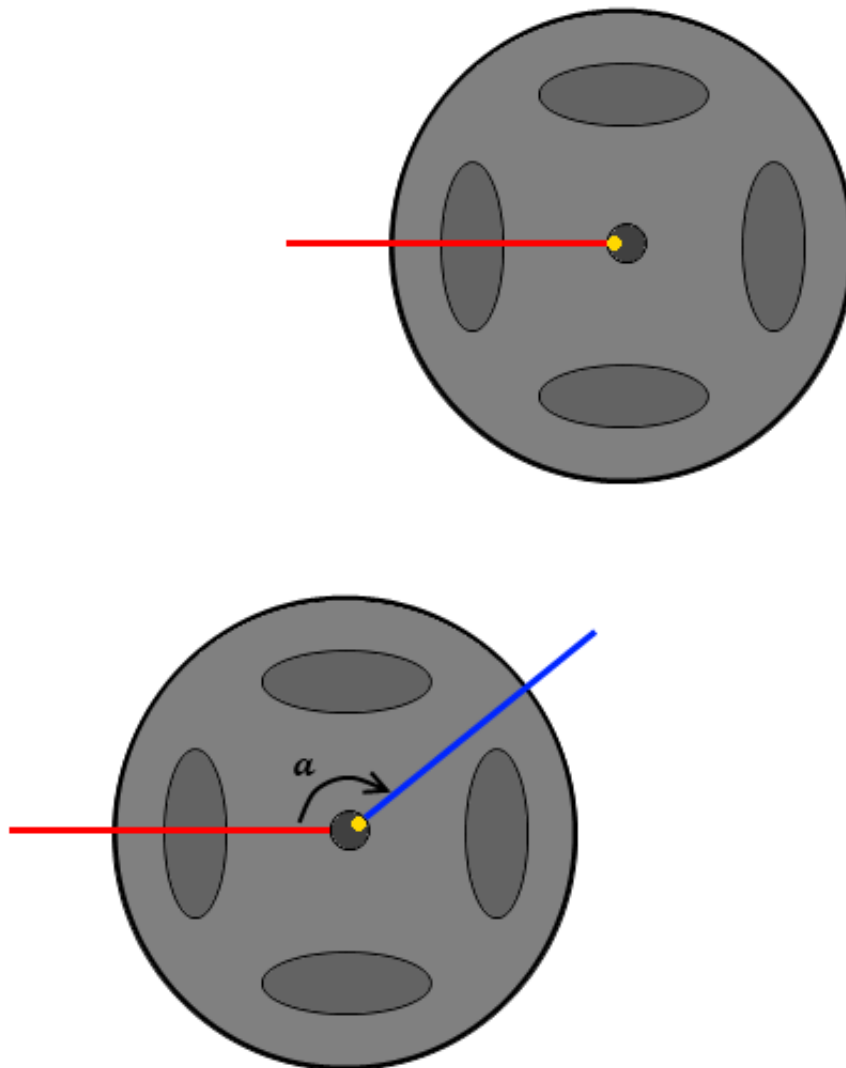
On pourra également se souvenir que plus la force exercée sur l'axe de rotation d'un moteur est grande, plus il faudra un couple élevé. Et plus le couple du moteur sera élevé, moins votre futur robot aura de difficultés à supporter de lourdes charges. Cela dit, tout n'est pas parfait car plus la charge est lourde, plus la consommation électrique du moteur va augmenter. On va voir la relation qui recoupe ces deux informations.

Dans le système international, l'expression du couple se fait en N.m (Newton mètre), mais le commun des mortels arrive mieux à interpréter des kilos plutôt que des Newtons, donc les constructeurs prennent des raccourcis. Pour passer des Newtons en kilos, il suffit simplement de les multiplier par la constante gravitationnelle 'g' (qui vaut environ 9.81). Soit $9.81N \approx 1kg$. Il en équivaut alors la même formule introduisant les mètres : $9.81N.m = 1kg.m$.

La vitesse de rotation

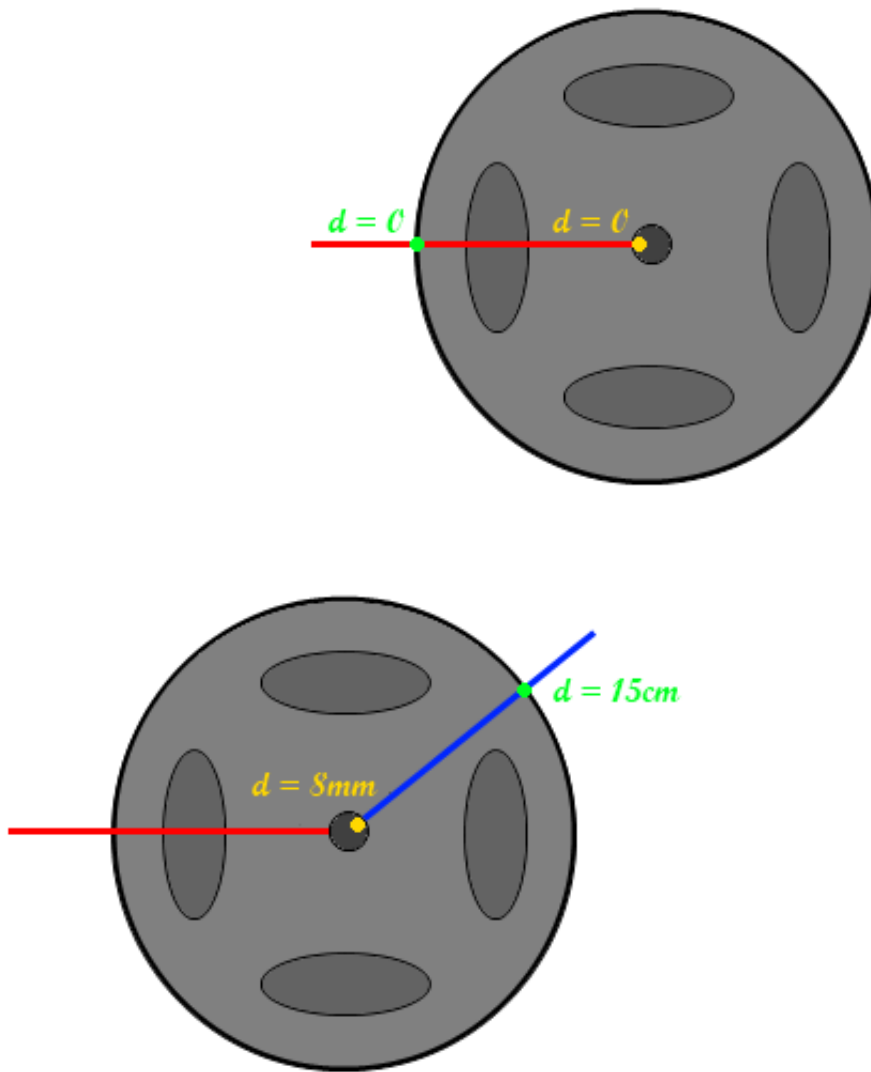
La vitesse de rotation est mesurée par rapport à l'axe de rotation du moteur. Imaginons que le moteur entraîne son axe, lorsqu'il est alimenté par un courant, ce dernier va avoir une vitesse de rotation. Il peut tourner lentement ou rapidement. On mesure une vitesse de rotation en mesurant l'angle en radians parcourus par cet axe pendant une seconde. C'est à dire que le moteur est en fonctionnement, que son axe tourne et que l'on mesure jusqu'où va l'axe de rotation, à partir d'un point de départ fixe, en une seconde.

Regardez plutôt l'image suivante pour mieux visualiser ce que je veux vous dire (Comprenez que le truc gris et rond c'est le moteur que j'ai dessiné. 😊 On le voit de face et le cercle au milieu c'est son axe) :



Marquage de l'axe du moteur par un point jaune (gauche). Au bout d'une seconde (droite), mesure de l'angle α entre la position de départ et d'arrivée du point jaune. On obtient alors la vitesse de rotation de l'axe du moteur. Cette mesure est exprimée en angle par seconde.

Savez-vous pourquoi l'on mesure ainsi la vitesse de rotation de l'axe du moteur ? Eh bien car cette mesure est indépendante du diamètre de cet axe. Et oui, car un point éloigné du centre de l'axe du moteur a une distance beaucoup plus grande à parcourir que son homologue proche du centre de l'axe. Du coup, pour aller parcourir une distance plus grande en un temps donné il est obligé d'aller plus vite :



En prenant la mesure à partir d'un point de départ fixe, la distance parcourue par le point jaune et vert est nulle (gauche). En faisant tourner l'axe du moteur pendant une seconde, on s'aperçoit que la distance parcourue par chaque point est différente (droite). La distance parcourue par le point vert est quasiment 20 fois plus grande que celle parcourue par le point jaune ! Et c'est pourquoi le point vert aura été plus rapide que le point jaune car la distance qu'il parcourt en un même temps est beaucoup plus grande.

En mécanique, comme on aime les choses marrantes on exprime la vitesse de rotation en radians par seconde *rad/s* et son symbole est le caractère grec ω , prononcez 'oméga'. Pour rappel, 360 est aux degrés ce que 2 pi est aux radians (autrement dit, une vitesse de 2π /secondes équivaut à dire "l'axe fait un tour par seconde"). Cela se traduit par $360^\circ = 2\pi$ radian. Malheureusement, la vitesse de rotation angulaire n'est pas donnée avec les caractéristiques du moteur. En revanche, on trouve une vitesse en tour/minutes (*tr/mn*). Vous allez voir que pour passer de cette unité aux rad/s, c'est assez facile. En effet, on sait qu'un tour correspond à une rotation de l'axe sur 360° . Soit $1tr = 360^\circ$. Et dans une minute il y a 60 secondes. Donc l'axe tourne $\frac{1}{60}$ de tour par seconde, s'il fait un tour par minute. On peut alors établir la relation suivante :

$$1tr/mn = 360 \times \frac{1}{60} = 6^\circ/s$$

Hors, on sait que $360^\circ = 2\pi \text{ rad}$, ce qui donne une nouvelle relation :

$$1 \text{ tr/mn} = 2\pi \times \frac{1}{60} = \frac{\pi}{30} \text{ rad/s}$$

On peut finalement donner la formule qui convertit un radian par seconde en tours par minutes :

$$1 \text{ rad/s} = \frac{1}{\frac{\pi}{30}} = \frac{30}{\pi} \approx 9,55 \text{ tr/mn}$$

Et je fais comment si je veux savoir à quelle vitesse ira mon robot ?

Eh bien comme je vous l'expliquais précédemment, pour répondre à cette question il faut connaître le diamètre de la roue. Prenons l'exemple d'une roue ayant 5cm de diamètre (soit 0.05 mètres) et un moteur qui tourne à 20 rad/s. Le périmètre de la roue vaut donc 15.7 cm (0.157 m) d'après la formule du périmètre d'un cercle qui est $P = 2 \times \pi \times r$, avec r le rayon du cercle. Cela signifie qu'en faisant tourner la roue sur une surface plane et en lui faisant faire un tour sur elle-même, la roue aura parcouru 0,157m sur cette surface. On admet que le moteur tourne à 20 rad/s ce qui représente donc 3.18 tours de l'axe du moteur par seconde (d'après la dernière formule que je vous ai donnée). On peut donc calculer la distance parcourue en une seconde grâce à la formule :

$$V = \frac{d}{t}$$

Avec :

- V : la vitesse en mètre par seconde (m/s)
- d : la distance en mètre (m)
- t : le temps en secondes (s)

On va donc adapter cette formule avec la distance qu'a parcouru la roue en faisant un tour sur elle-même (d_{roue}) et le nombre de tours par seconde de l'axe du moteur (t_{tour}) :

$$V = \frac{d_{\text{roue}}}{t_{\text{tour}}} \quad \text{On sait que } d_{\text{roue}} = 0.157\text{m} \quad \text{et que } t_{\text{tour}} = 3,18 \text{ tr/s} = \frac{1}{3,18} \text{ tr.s}$$

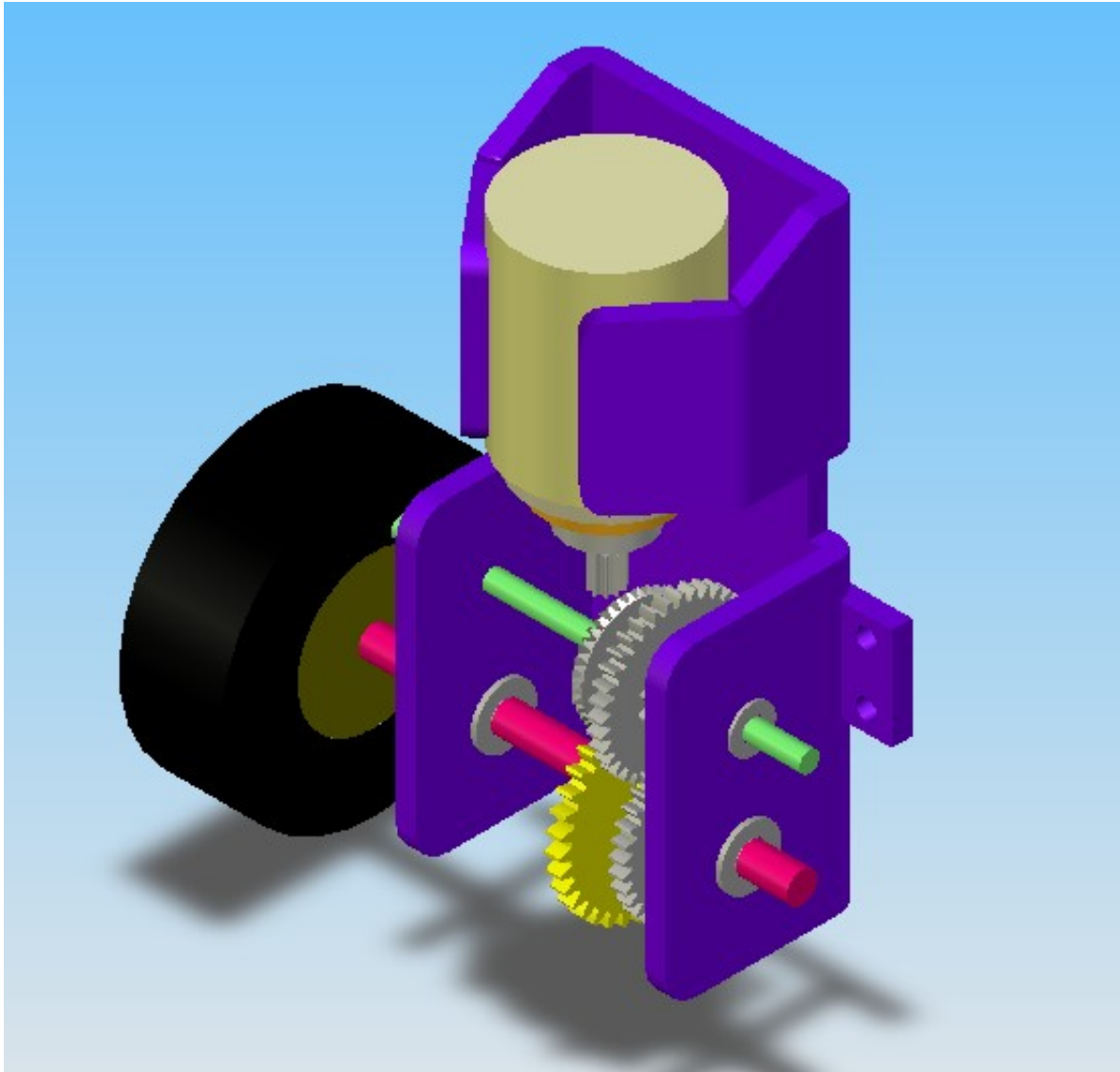
$$V = \frac{0,157}{\frac{1}{3,18}} = 0,157 \times 3,18 \quad V = 0,5 \text{ m/s} \quad \text{Le robot parcourt donc une distance de}$$

50 centimètres en une seconde (ce qui équivaut à 180 mètres par heure). Vous avez maintenant toutes les cartes en main pour pouvoir faire avancer votre robot à la vitesse que vous voulez !

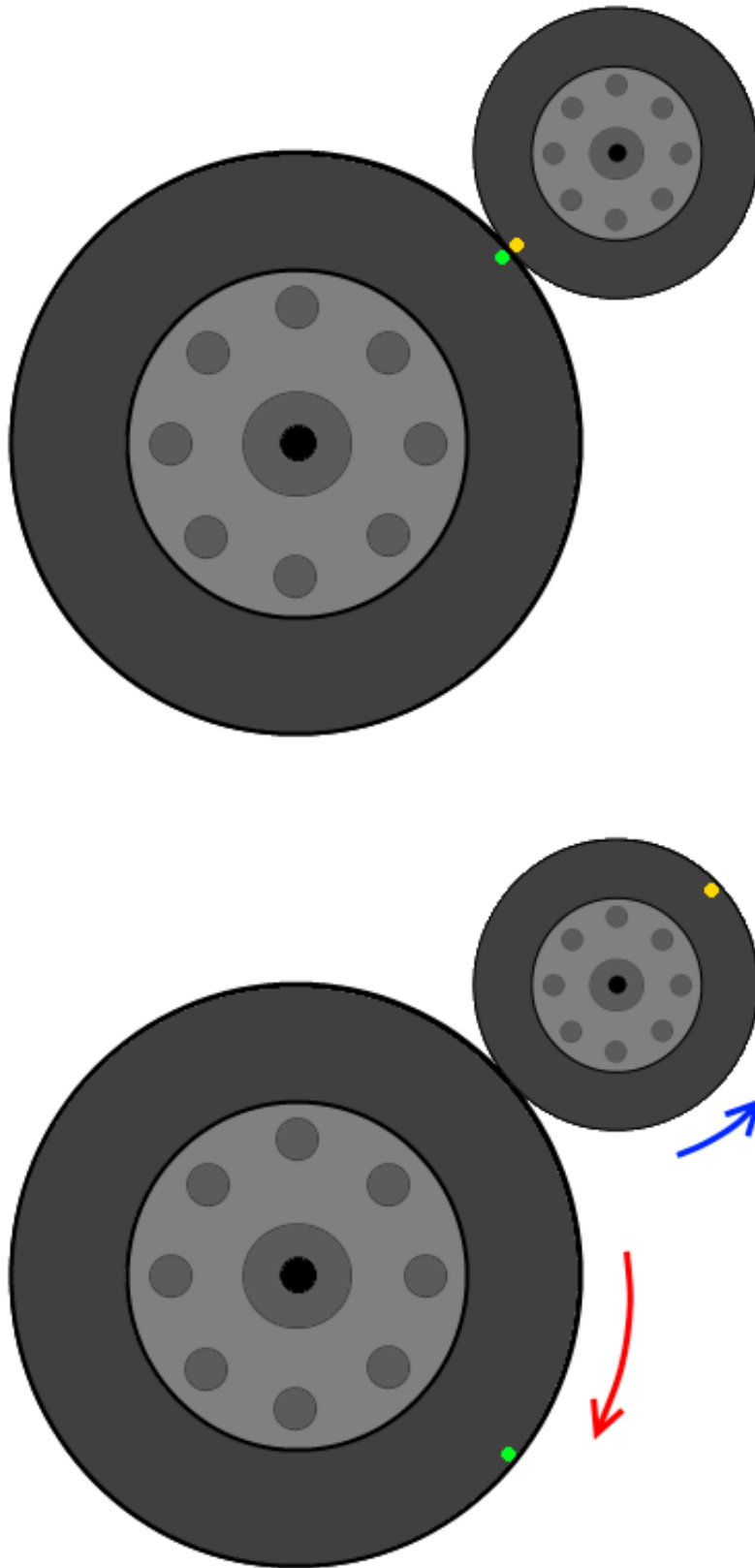
Les réducteurs

Un moteur électrique est bien souvent très rapide en rotation. Hors si vous avez besoin de faire un robot qui ne va pas trop vite, il va falloir faire en sorte de réduire sa vitesse de rotation. On peut très bien mettre un "frein" qui va empêcher le moteur de tourner vite, ou bien le piloter (on va voir ça toute à l'heure). Cela dit, même si on réduit sa vitesse de rotation, le moteur ne va pas pouvoir supporter des charges lourdes. Autrement dit, votre robot ne pourra même pas se supporter lui-même ! Nous avons donc besoin de couple. Et pour avoir du couple, tout en réduisant la vitesse de rotation, on va utiliser ce

que l'on appelle un **réducteur**. Un réducteur est un ensemble composé d'**engrenages** qui permet de réduire la vitesse de rotation de l'axe du moteur tout en augmentant le couple de sortie. Sur l'image suivante, extraite du site de l'[Académie d'Aix Marseille](#), on peut observer un ensemble moteur + réducteur + roue :



La règle qui régit son fonctionnement indique qu'entre deux engrenages la puissance est conservée (aux pertes près qui sont dues au frottement des engrenages entre eux). Et comme la puissance mécanique est dépendante du couple et de la vitesse (partie suivante), on peut facilement passer de l'un à l'autre. Reprenons notre roue faisant 5cm de diamètre. Mettez en contact contre elle une grande roue de 10cm de diamètre (deux fois plus grande). Lorsque la petite roue fait un tour, elle va entraîner la deuxième roue plus grande qui va faire... un demi-tour. Oui car le périmètre de la grande roue est deux fois plus grand que celui de la petite. Lorsque la petite parcourt 0,157m en faisant un tour sur elle-même, la grande parcourt elle aussi cette distance mais en ne faisant qu'un demi-tour sur elle-même.



Deux roues en contact, la petite entraîne la grande dont le diamètre est deux fois plus grand que la petite (gauche). Le point vert et jaune sert à repérer la rotation de chaque roue. Lorsque la petite roue fait un demi tour, la grande roue fait un quart de tour (droite). Si elle fait un tour complet, la grande roue ne fera qu'un demi-tour.

Ce que l'on ne voit pas sur mon dessin, c'est le couple. Hors, ce que vous ne savez peut-être pas, c'est que l'axe de la grande roue bénéficie en fait de deux fois plus de

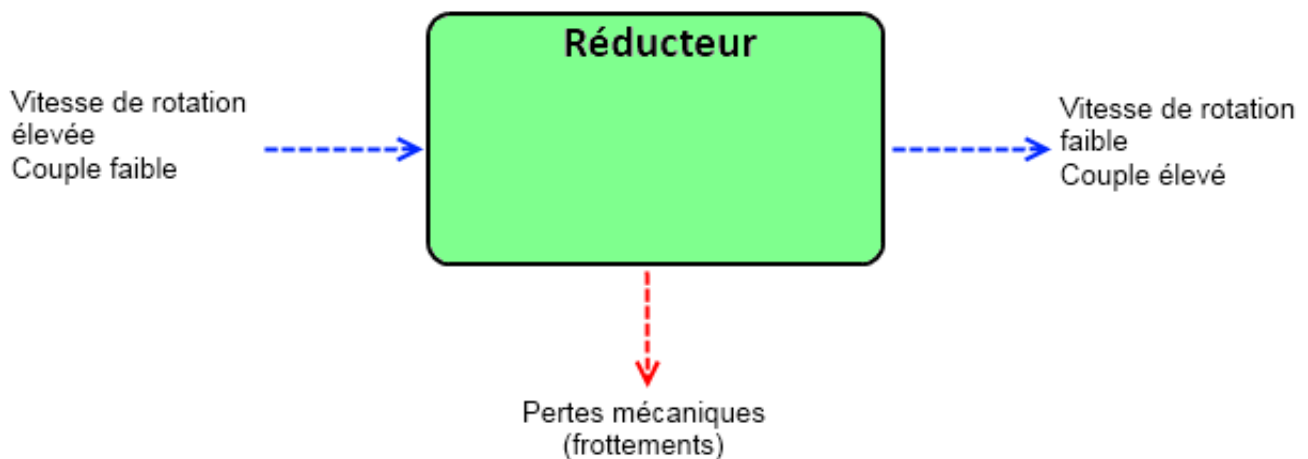
couple que celui de la petite. Car les réducteurs ont pour propriété, je le disais, de modifier le couple de sortie et la vitesse. Et ce selon la relation suivante qui donne le **rapport de réduction** :

$$R = \frac{\omega_{entree}}{\omega_{sortie}} = \frac{C_{sortie}}{C_{entree}}$$

Avec :

- R : le rapport de réduction du réducteur
- ω_{entree} : la vitesse de rotation de l'axe du moteur en entrée du réducteur
- ω_{sortie} : la vitesse de rotation de l'axe du moteur en sortie du réducteur
- C_{sortie} : couple exercé par l'axe de sortie du réducteur
- C_{entree} : couple exercé par l'axe du moteur, en entrée du réducteur

Un réducteur s'apparente donc à un système qui modifie deux grandeurs qui sont liées : le couple et la vitesse. On peut schématiser le fonctionnement d'un réducteur de la manière suivante :



C'est quoi ça, les pertes mécaniques ? 🤔

Justement, venons-en à un autre point que je voudrais aborder.

La puissance et le rendement

Dans un moteur, on trouve deux puissances distinctes :

- La première est la **puissance électrique**. Elle représente la quantité d'énergie électrique dépensée pour faire tourner l'axe du moteur. Elle représente aussi la quantité d'énergie électrique induite lorsque le moteur tourne en générateur, c'est à dire que le moteur transforme une énergie mécanique de rotation en une énergie électrique. Elle se calcule simplement à partir de la formule suivante :

Puissance = Tension x Courant

$$P_{elec} = U \times I$$

Selon les conventions, la tension est exprimée en Volt et le courant en Ampère. Quant à la puissance, elle est exprimée en **Watt (W)**.

- La seconde est la **puissance mécanique**. Elle correspond au couple du moteur multiplié par sa vitesse angulaire :

$$P_{meca} = C \times \omega$$

Le couple doit être exprimé en Newton-Mètre (Nm) et la vitesse en radians par seconde (rad/s). Pour la puissance mécanique, il s'agit encore de Watt.

Une puissance (mécanique ou électrique) s'exprime habituellement en **Watts** (symbole **W**). On retrouve cependant d'autres unités telle que le Cheval Vapeur (CV), avec 1 CV qui vaut (arrondi) 735,5 W.

Mais comme dans tout système, la perfection n'existe pas, on va voir la différence qu'il y a entre la puissance mécanique et électrique, alors que *à priori* elles devraient être équivalentes. Lorsque le moteur est en fonctionnement, il génère des **pertes**. Ces pertes sont dues à différents **phénomènes électriques** ou **thermiques** (échauffement) ou tels que les **frottements mécaniques** (air, pièces en contact, magnétique). **Il y a donc une différence entre la puissance (électrique) en entrée du moteur et la puissance (mécanique) en sa sortie.** Cette différence s'exprime avec la notion de **rendement**. Le rendement est une caractéristique intrinsèque à chaque moteur et permet de définir l'écart entre la puissance d'entrée du moteur et sa puissance de sortie. Il s'exprime sans unité. Il permet également de savoir quel est le pourcentage de pertes provoquées par le moteur. Le rendement se note avec la lettre grecque *eta* (η) et se calcule grâce à la formule suivante :

$$\eta = \frac{P_{sortie}}{P_{entree}}$$

Dans le cas du moteur, on aurait alors les puissances électrique et mécanique telles quelles :

$$\eta = \frac{P_{meca}}{P_{elec}}$$

Et dans le cas où le moteur est utilisé en générateur électrique (on fait tourner l'axe à la main par exemple), la formule reste la même mais la place des puissances électrique et mécanique est inversée :

$$\eta = \frac{P_{elec}}{P_{meca}}$$

Attention, le rendement est une valeur sans unité, on peut en revanche l'exprimer sous forme de pourcentage.

Si l'on prend un exemple : un moteur de puissance électrique 100W, ayant une puissance mécanique de 84W aura un rendement de : $\eta = \frac{P_{meca}}{P_{elec}} \quad \eta = \frac{P_{84}}{P_{100}} \quad \eta = 0,84$
Ce qui correspond à 84%. Sachez toutefois que le rendement ne pourra dépasser les 100% (ou 1), car **il n'existe pas de systèmes capables de fournir plus d'énergie qu'ils n'en reçoivent**. Cela dit, si un jour vous parvenez à en trouver un, vous pourrez devenir le Roi du Monde !! :Pirate:

Les moteurs électriques ont habituellement un bon rendement, entre 80% (0.8) et 95% (0.95). Cela signifie que pour 100W électriques injectés en entrée, on obtiendra en sortie 80 à 95W de puissance mécanique. Tandis qu'un moteur à explosion de voiture dépasse à peine les 30% de rendement !

Quelques relations

Une toute dernière chose avant de commencer la suite, il y a deux relations à connaître vis-à-vis des moteurs.

Lien entre vitesse et tension

Dans un moteur CC, quelque soit sa taille et sa puissance, il faut savoir que la tension à ses bornes et la vitesse de sortie sont liées. Plus la tension sera élevée et plus la vitesse sera grande. Nous verrons cet aspect dans la prochaine partie. Faites attention à bien rester dans les plages de tension d'alimentation de votre moteur et ne pas les dépasser. Il pourrait griller ! En effet, vous pouvez dépasser **de manière temporaire** la tension 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).

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 dû 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.

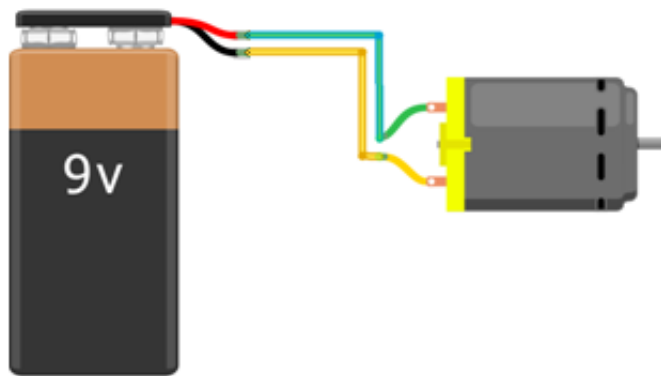
Si vous faites circuler trop de courant dans un moteur pour trop longtemps, il va chauffer. Les moteurs sont des composants sans protection. Même s'ils chauffent ils ne feront rien pour s'arrêter, bien au contraire. Cela peut mener à une surchauffe et une destruction du moteur (les bobines à l'intérieur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

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...

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 :



C'est beau, ça tourne.

C'est tout ? o_O

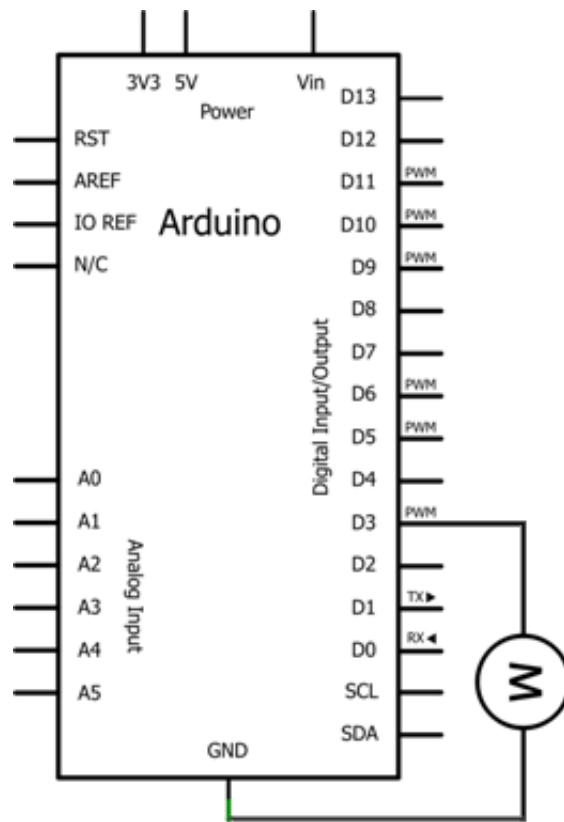
Ben oui, quoi de plus ? Le moteur est connecté, son axe tourne, la pile débite du courant... Ha ! Voilà ce qui nous intéresse dans l'immédiat : la pile débite du courant. Oui et pas des moindres car les moteurs électriques sont bien généralement de véritables gloutons énergétiques. Si vous avez la chance 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. 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.

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.

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 :



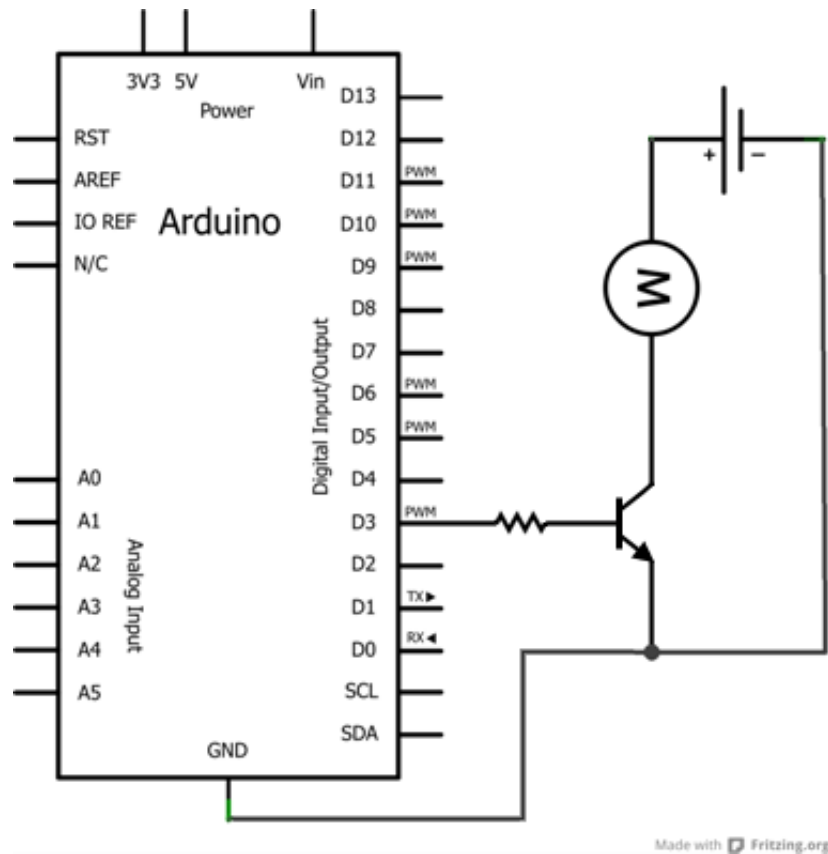
Avec le programme adéquat, le moteur va tourner à la vitesse que l'on souhaite, si l'on veut, réglable par potentiomètre et s'arrêter ou démarrer quand on le lui demande. C'est mieux. C'est la carte Arduino qui pilote le moteur. Malheureux ! Vous ne croyez tout de même pas que l'on va se contenter de 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.

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

Je n'ai pas dit 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.

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 avons alors commandé chaque transistor pour allumer ou éteindre les afficheurs correspondants. Essayons de faire de même avec notre moteur :



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

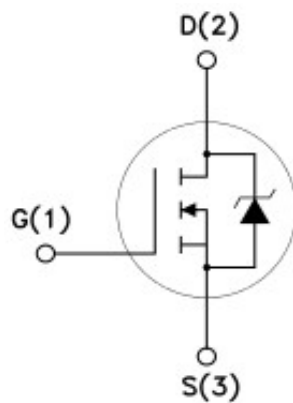
Pour répondre à ces trois contraintes, trois solutions. La première consisterait à mettre un transistor qui accepte un courant assez élevé par rapport à la consommation réelle du moteur, ou bien d'adjoindre un *dissipateur* sur le transistor pour qu'il refroidisse. La deuxième solution concernant les 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.

Le “bon” transistor

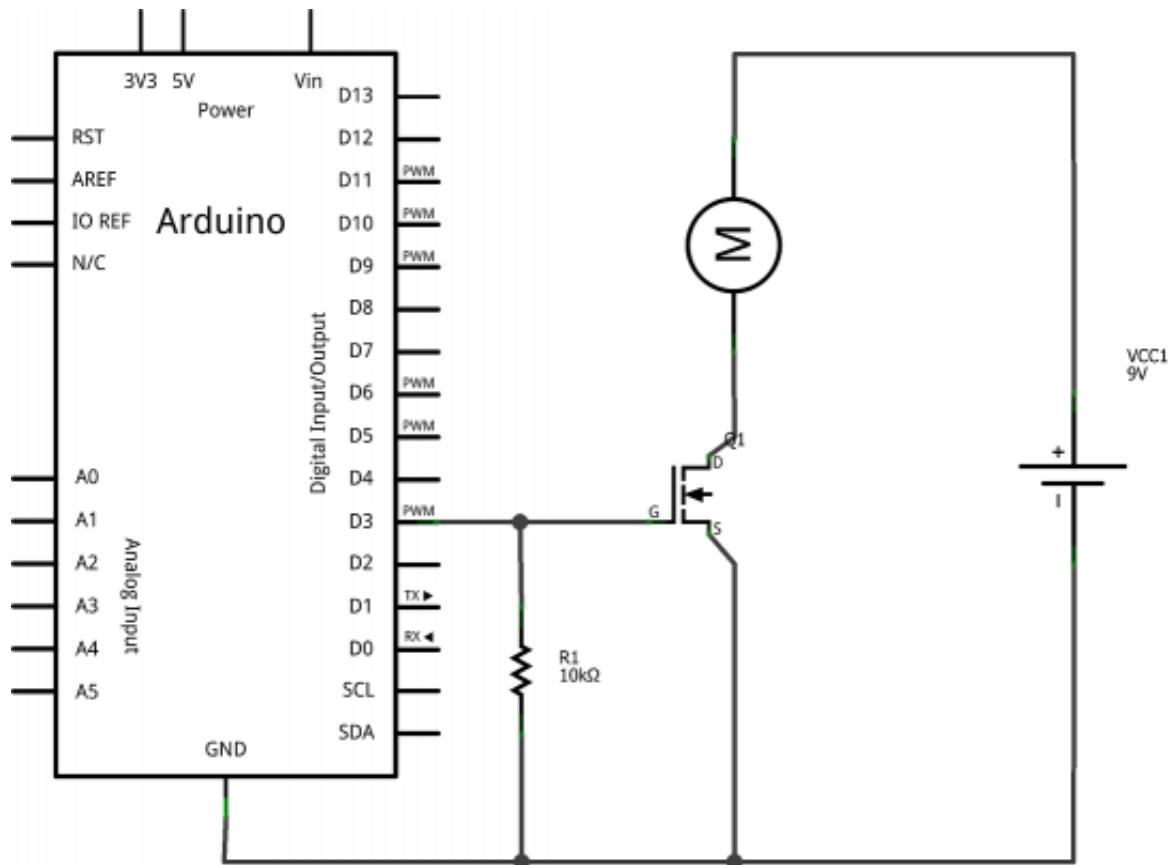
Comme je viens de vous l'expliquer, il nous faut un transistor comme “interface” de puissance. C'est lui qui nous sert d'interrupteur pour laisser passer ou non le courant. Pour l'instant, nous avons beaucoup parlé des transistors “bipolaires”. Ils sont sympas, pas chers, mais il y a un problème : ils 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.

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 :



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) :



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 :

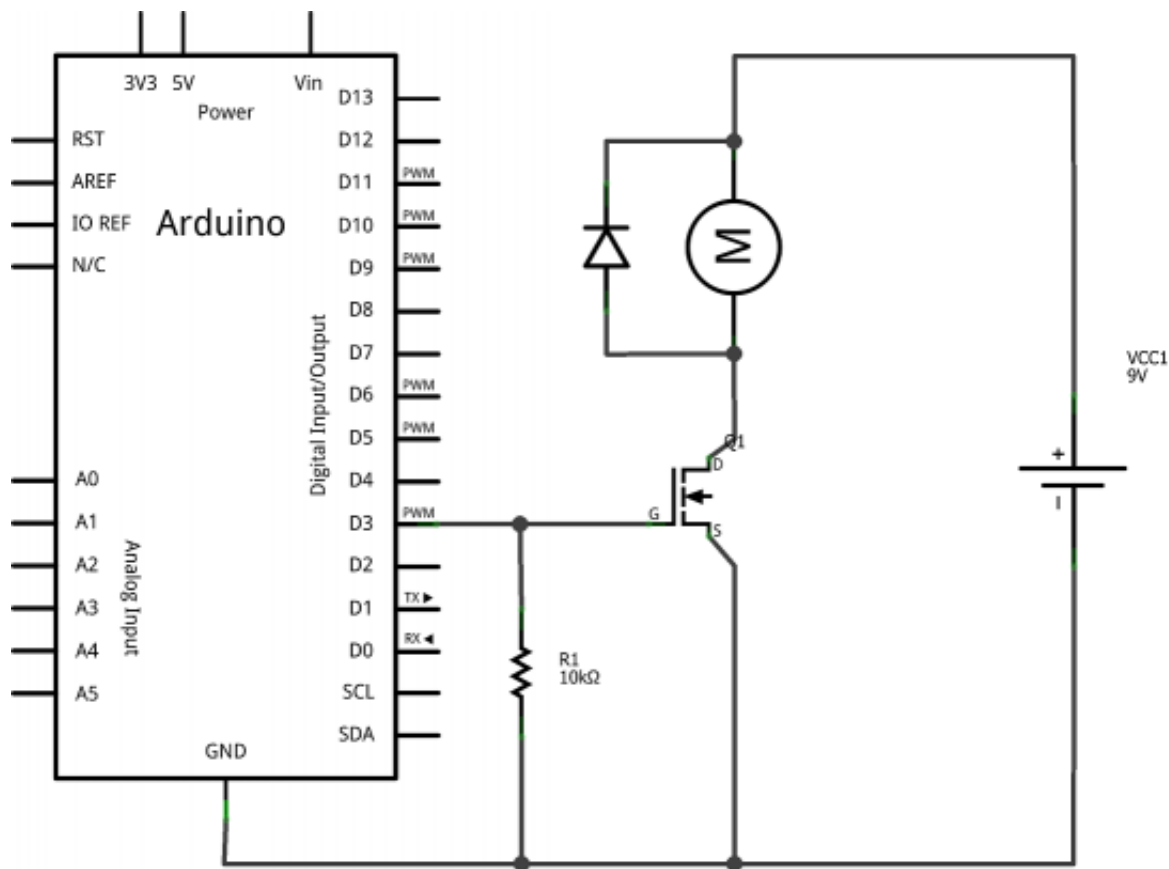


...est un composant électronique qui ne laisse passer le courant que dans un sens (cf. [ce chapitre](#)). Vos souvenirs sont-ils à nouveau en place ? Alors, on continue ! Reprenons le schéma précédent avec le transistor piloté par l'Arduino et qui commande à son tour le moteur. Saturons le transistor 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 ?

Le moteur va continuer de tourner à cause de son inertie !

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 :

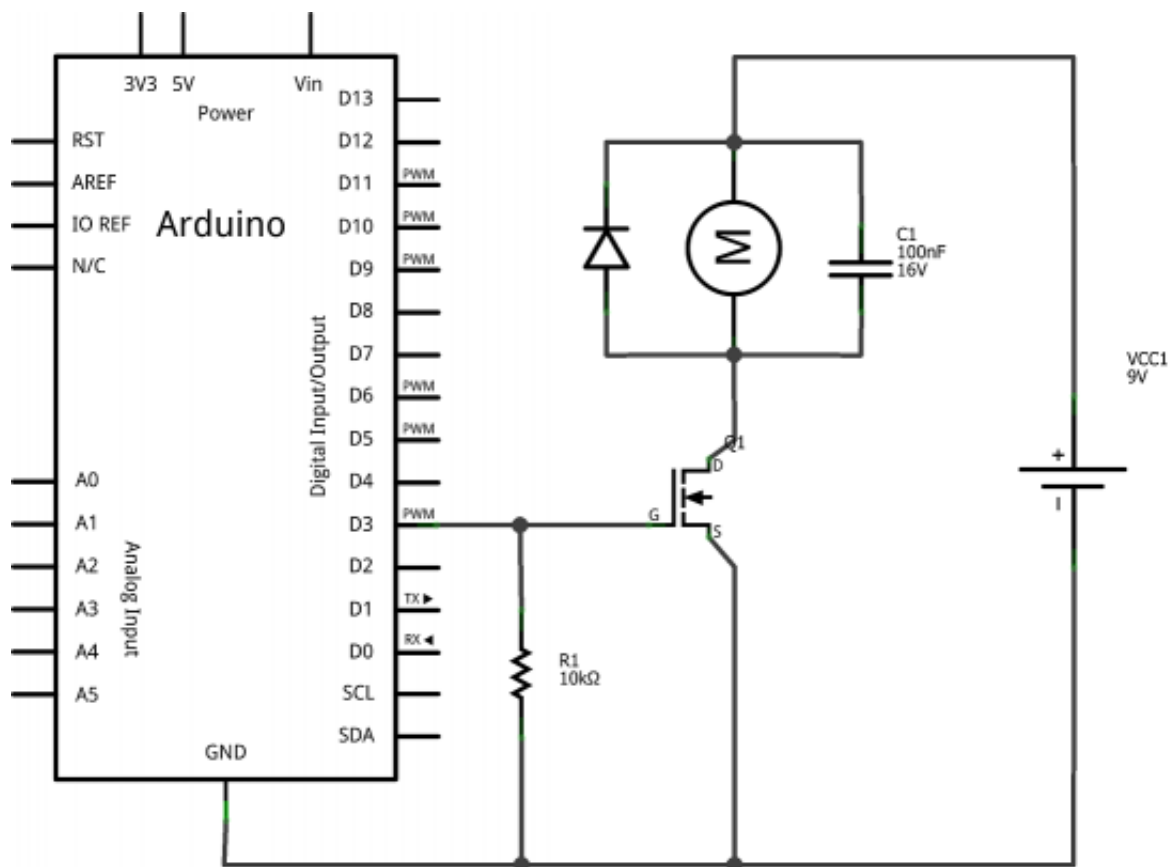


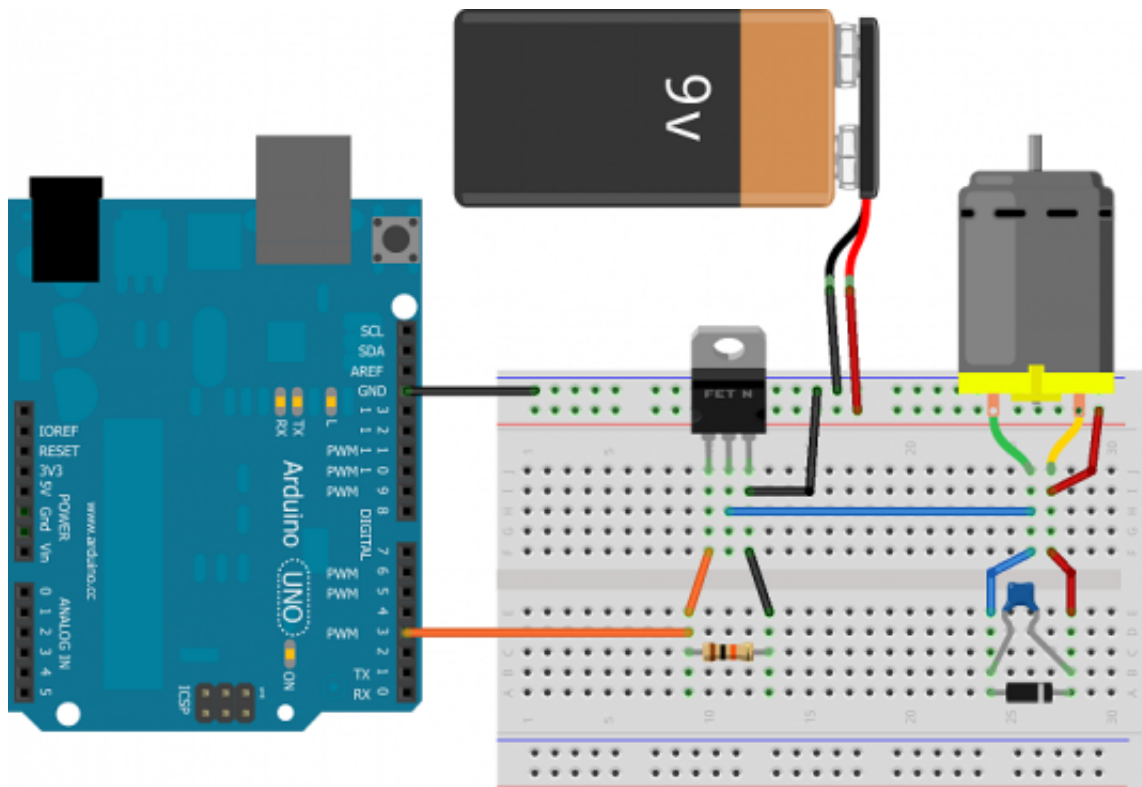
Reprenons au moment où le moteur tourne. Plus de courant ne circule dans le transistor et la seule raison pour laquelle le moteur continue de tourner est qu'il possède une inertie mécanique. Il génère donc cette fameuse tension qui est orientée vers l'entrée du transistor. Comme le transistor est bloqué, 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'auto-freiner".

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 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 :





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:

Piloter un moteur

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. 🤖

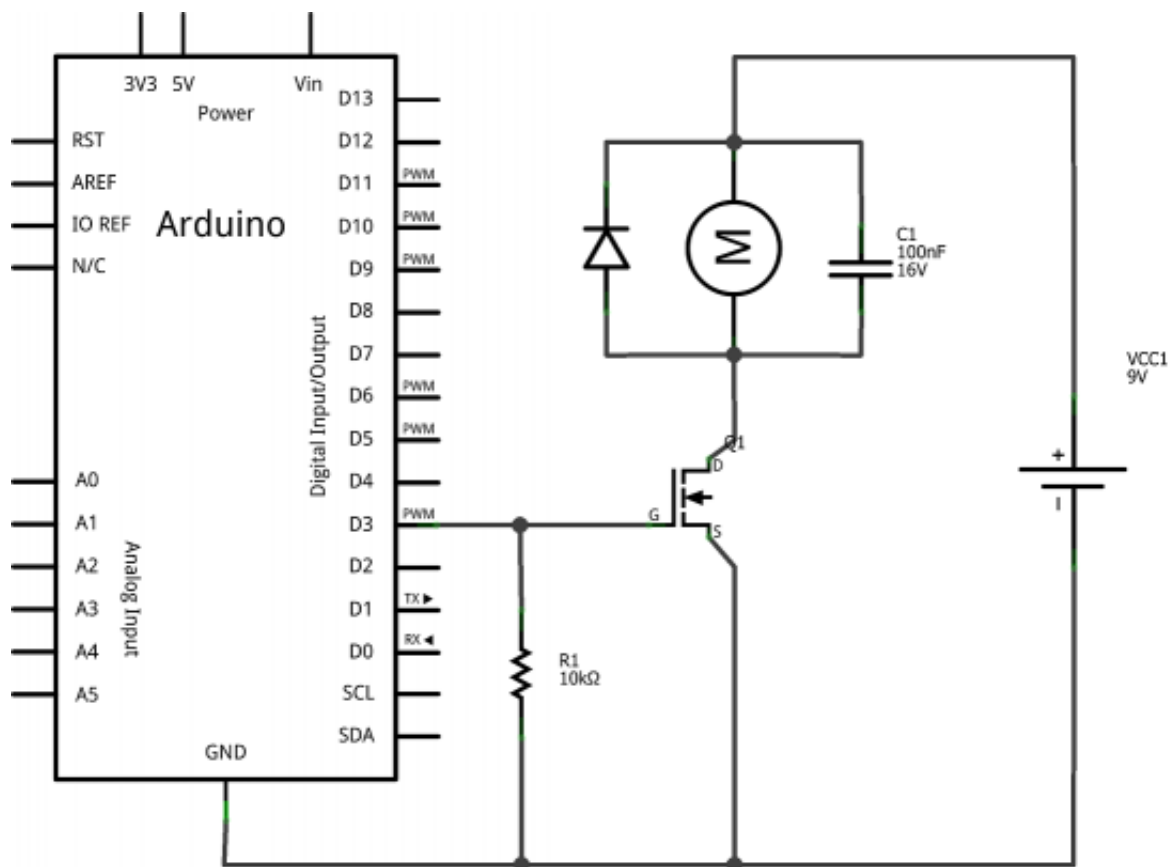
Faire varier la vitesse : la PWM

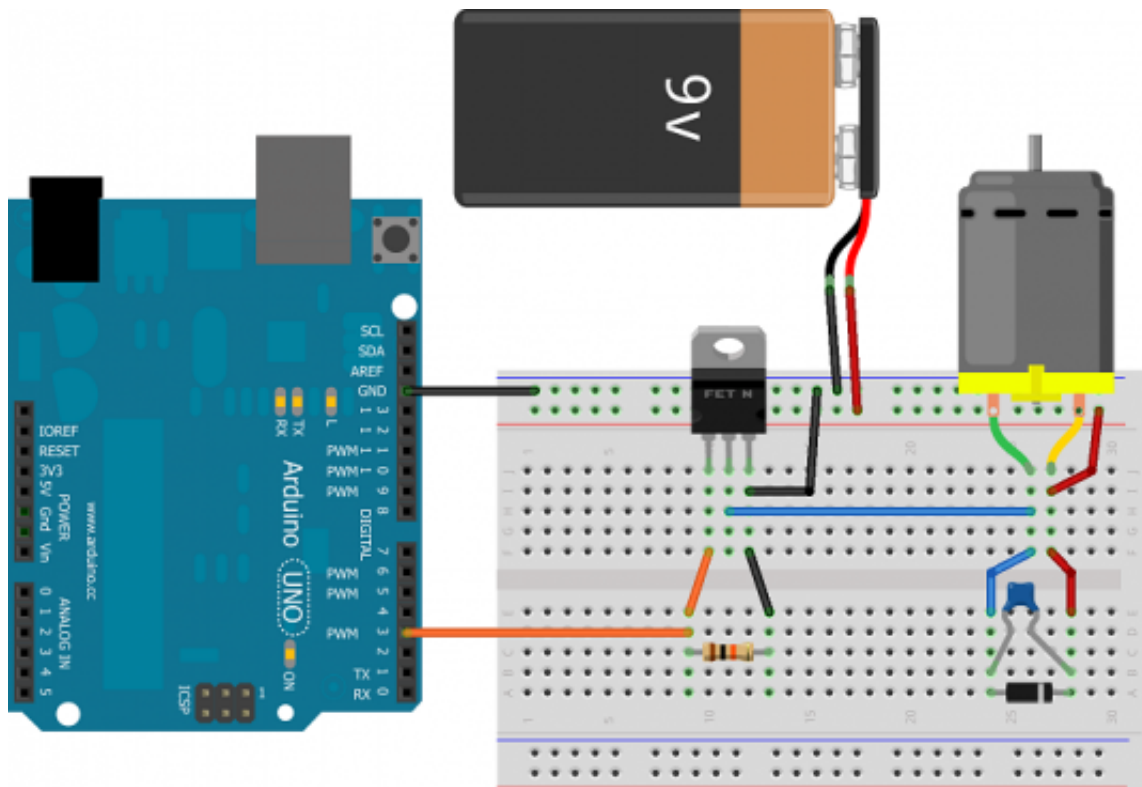
Maintenant que nous avons les bases fondamentales pour faire tourner notre moteur sans tout faire griller (😓), nous allons pouvoir acquérir d'autres connaissances. À commencer par quelque chose de facile : le réglage de la vitesse de rotation du moteur. Comme nous l'expliquions dans le premier morceau 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 rafraîchir 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.

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 :





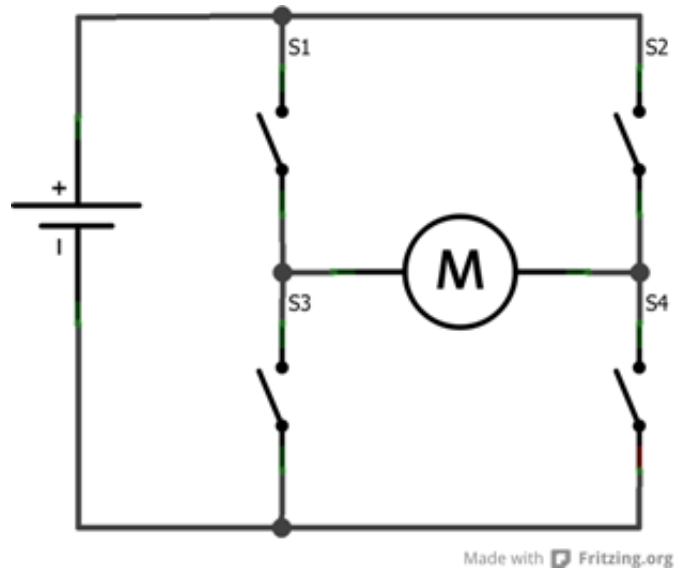
Maintenant que le moteur tourne à une vitesse réglable, il pourra être intéressant de le faire tourner aussi dans l'autre sens (si jamais on veut faire une marche arrière, par exemple, sur votre robot), voire même d'être capable de freiner le moteur. C'est ce que nous allons tout de suite étudier dans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

Tourner dans les deux sens : le pont en H

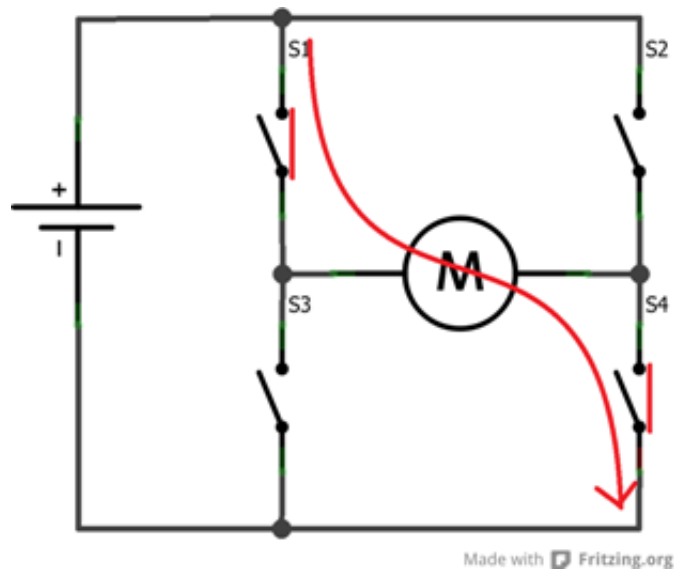
Faire tourner un moteur c'est bien. Tourner à la bonne vitesse c'est mieux. Aller dans les deux sens c'est l'idéal. C'est donc ce que nous allons maintenant chercher à faire !

Découverte du pont en H

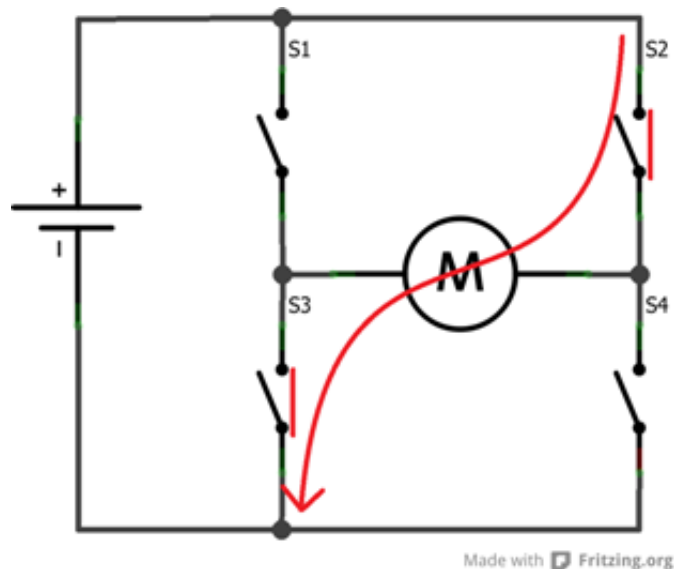
Tout d'abord une question très simple : pourquoi le moteur tourne dans un seul sens ? Réponse évidente : parce que le courant ne va que dans un seul sens ! Pour pouvoir aller vers l'avant ET vers l'arrière il nous faut donc un dispositif qui serait capable de faire passer le courant dans le moteur dans 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 "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 !



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.



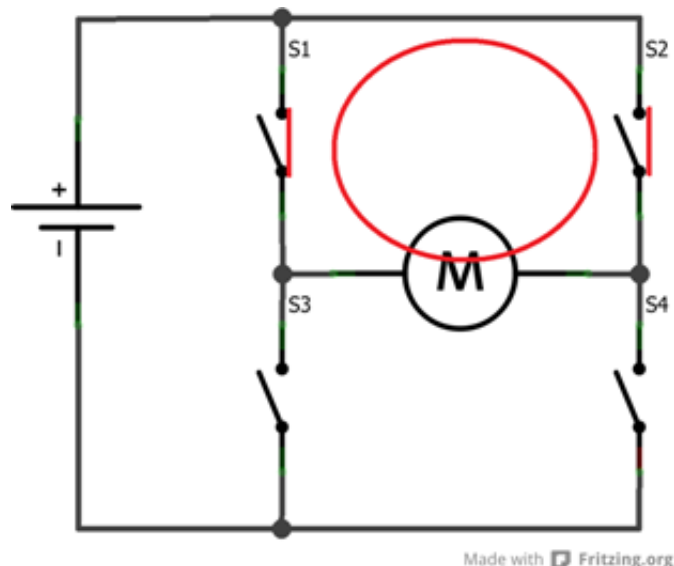
Si en revanche je fais le contraire (2 et 3 fermés et 1 et 4 ouverts), le courant ira dans l'autre sens ! C'est génial non ?

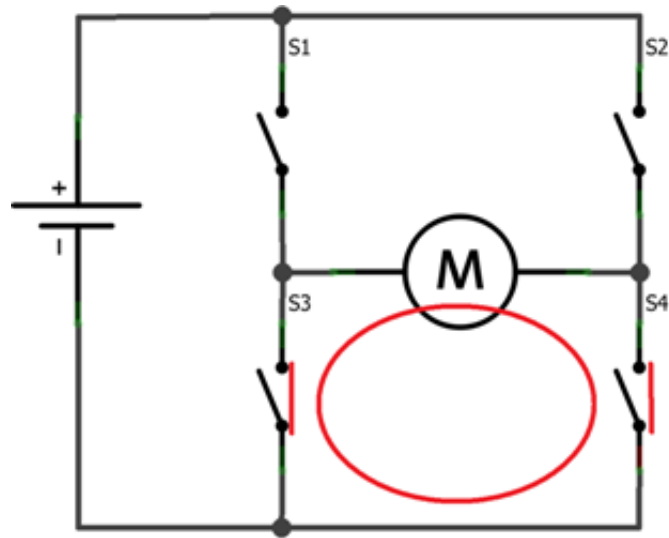


Et ce n'est pas tout !

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.





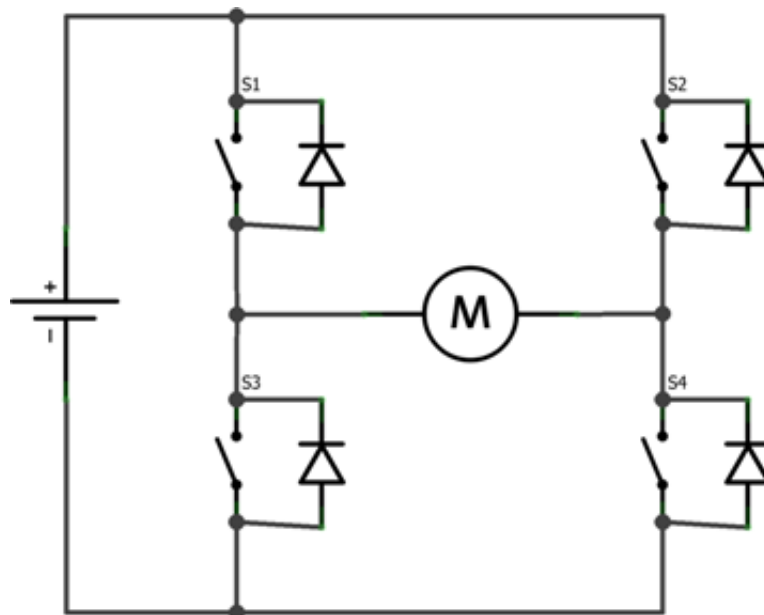
Made with Fritzing.org

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.

Les protections nécessaires

Les diodes de roue libre

Comme nous l'avons vu plus haut, pour protéger un transistor des parasites ou lors du freinage électronique du moteur, nous plaçons une diode. Dans le cas présent, cette diode devra être en parallèle aux bornes du transistor (regardez le schéma qui suit). Ici nous avons quatre transistors, nous utiliserons donc quatre diodes que nous placerons sur chaque transistor. Ainsi, le courant trouvera toujours un moyen de passer sans risquer de forcer le passage dans les transistors en les grillant. Comme vu précédemment, des diodes de type Schottky sont recommandées pour leurs caractéristiques de tension de seuil faible et commutation rapide.

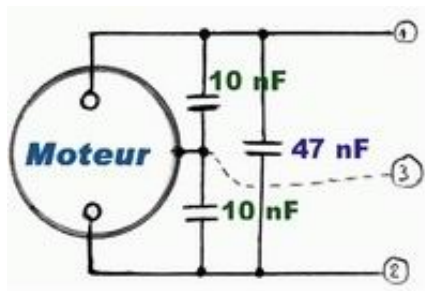


Made with Fritzing.org

Un peu de découplage

Lorsque nous utilisons le moteur avec une PWM, nous générons une fréquence

parasite. De plus, le moteur qui tourne génère lui même des parasites. Pour ces deux raisons, il est souvent utile d'ajouter des **condensateurs de filtrage** aux bornes du moteur. Comme sur le montage suivant, on peut en placer un en parallèle des deux broches du moteur, et deux autres plus petits entre une broche et la carcasse du moteur.



Ensuite, lorsque le moteur démarre il fera un appel de courant. Pour éviter d'avoir à faire transiter ce courant depuis la source de tension principale (une batterie par exemple), il est de bon usage de mettre un gros condensateur polarisé aux bornes de l'alimentation de puissance du pont en H. Ainsi, au moment du départ l'énergie sera en partie fournie par ce condensateur plutôt qu'en totalité par la batterie (ce qui évitera un échauffement abusif des conducteurs mais aussi une éventuelle baisse de la tension due à l'appel de courant).

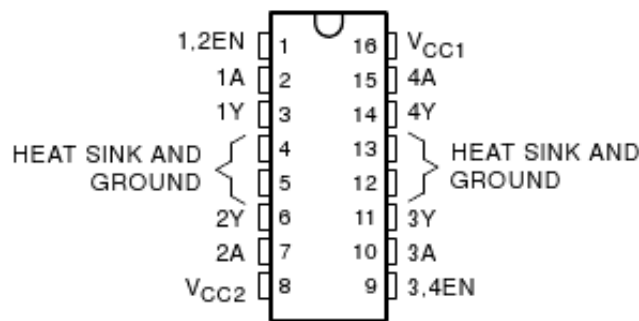
Des solutions intégrées : L293, L298...

Afin d'éviter de vous torturer avec les branchements des transistors et leur logique de contrôle, des composants "clés en main" ont été développés et produits. Nous allons maintenant étudier deux d'entre eux que nous retrouvons dans quasiment tous les shields moteurs Arduino : le L293(D) et son grand frère, plus costaud, le L298.

Le L293(D)

Tout d'abord, voici un lien vers [la datasheet du composant](#). Les premières données nous apprennent que ce composant est un "quadruple demi-pont en H". Autrement formulé, c'est un double pont en H (car oui, 4 fois un demi ça fait 2 !). Ce composant est fait pour fonctionner avec des tensions de 4.5V à 36V et sera capable de délivrer 600 mA par canaux (dans notre cas cela fera 1,2A par moteur puisque nous utiliserons les demi-ponts par paire pour tourner dans les deux sens). Un courant de pic peut être toléré allant jusqu'à 1,2A par canaux (donc 2,4A dans notre cas). Enfin, ce composant existe en deux versions, le L293 et le L293D. La seule différence (non négligeable) entre les deux est que le L293D intègre déjà les diodes en parallèle des transistors. Un souci de moins à se préoccuper ! En revanche, cela implique donc des concessions sur les caractéristiques (le courant max passe à 1A par canaux et 2A pic pour la version sans les diodes). Le branchement de ce composant est assez simple (page 2 de la datasheet), mais nous allons le voir ensemble maintenant. Ce composant a 16 broches

et fonctionne selon un système de symétrie assez simple.



De chaque côté les broches du milieu (4, 5, 12 et 13) servent à relier la masse mais aussi à dissiper la chaleur. On trouve les entrées d'activation des ponts (*enable*) sur les broches 1 et 9. Un état HAUT sur ces broches et les ponts seront activés, les transistors pourront s'ouvrir ou se fermer, alors qu'un état BAS désactive les ponts, les transistors restent ouverts. Ensuite, on trouve les broches pour piloter les transistors. Comme un bon tableau vaut mieux qu'un long discours, voici les cas possibles et leurs actions :

Input 1 (broche 2 et 10)	Input 2 (broche 7 et 15)	Effet
0	1	Tourne dans le sens horaire
1	0	Tourne dans le sens anti-horaire
0	0	Frein
1	1	Frein

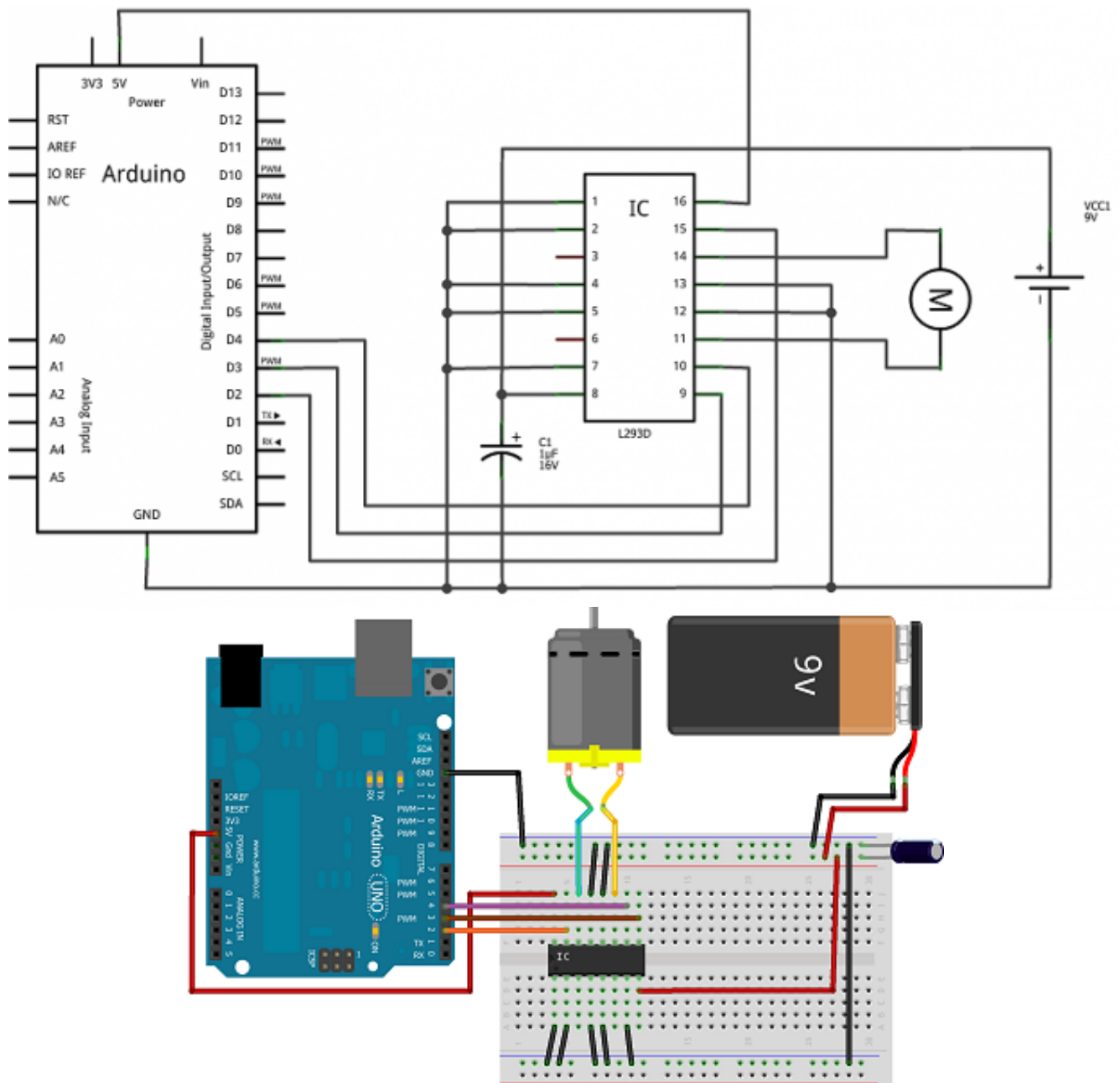
Ainsi, en utilisant une PWM sur la broche d'activation des ponts on sera en mesure de faire varier la vitesse. Il ne nous reste plus qu'à brancher le moteur sur les sorties respectives (2 et 7 ou 11 et 14 selon le pont utilisé) pour le voir tourner. 😊 Et voilà ! Vous savez à peu près tout ce qu'il faut savoir (pour l'instant 😊) sur ce composant.

Attends attends attends, pourquoi il y a deux broches Vcc qui ont des noms différents, c'est louche ça !

Ah oui, c'est vrai et c'est important ! Le composant possède deux sources d'alimentation. Une pour la partie "logique" (contrôle correct des transistors), VCC1 ; et l'autre pour la partie puissance (utile pour alimenter les moteurs à la bonne tension), VCC2. Bien que ces deux entrées respectent les mêmes tensions (4.5V à 36V), nous ne sommes pas obligés de mettre des tensions identiques. Par exemple, la tension pour la logique pourrait venir du +5V de la carte Arduino tandis que la partie puissance pourrait être fournie par une pile 9V par exemple (n'oubliez pas de bien relier les masses entre elles pour avoir un référentiel commun).

N'utilisez **JAMAIS** le +5V de la carte Arduino comme alimentation de puissance (pour la logique c'est OK). Son régulateur ne peut fournir que 250mA ce qui est faible. Si vous l'utilisez pour alimenter des moteurs vous risquez de le griller !

Comme je suis sympa (👍) je vous donne un exemple de branchement du composant avec un moteur et une carte Arduino (j'ai pris le modèle L293D pour ne pas m'embêter à devoir mettre les diodes de protection sur le schéma 😊) :

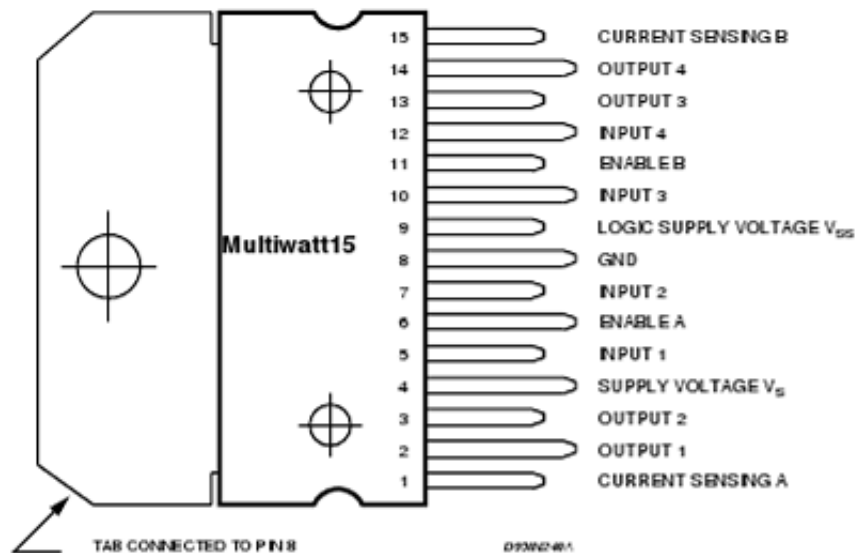


Vous noterez la présence du gros condensateur polarisé (100 µF / 25V ou plus selon l'alimentation) pour découpler l'alimentation de puissance du L293D. Comme je n'utilise qu'un seul pont, j'ai relié à la masse les entrées de celui qui est inutilisé afin de ne pas avoir des entrées qui "grésillent" et fassent consommer le montage pour rien. Enfin, vous remarquez que j'utilise trois broches de l'Arduino, deux pour le sens (2 et 4) et une PWM pour la vitesse (3).

Le L298

Étudions maintenant le grand frère du L293 : [le L298](#). Si je parle de grand frère ce n'est pas innocent. En effet, son fonctionnement est très similaire à celui du L293, mais il est capable de débiter des courants jusqu'à 2A nominal par pont et jusqu'à 3A pendant un bref instant. Il propose aussi une fonction pouvant être intéressante qui est la mesure du courant passant au travers du pont (pour vérifier si votre moteur est rendu en butée par exemple). Que dire de plus ? On retrouve deux broches d'alimentation, une pour la logique et l'autre pour la puissance. Celle pour la logique peut aller de 4.5 à 7V (là

encore on pourra utiliser celle de l'Arduino). L'entré puissance, en revanche, admet une tension comprise entre 5 et 46V. Pour un fonctionnement optimal, la documentation nous recommande de placer des condensateurs de 100nF sur chaque ligne d'alimentation. Et comme pour le L293, on pourra aussi placer un gros condensateur polarisé de 100µF (tension à choisir selon l'alimentation) sur la ligne d'alimentation de puissance. Comme le fonctionnement est le même que celui du L293, je vais juste vous proposer une liste des broches utiles (oui je suis fainéant !).



Pour le premier pont :

- Les sorties sont situées sur les broches 2 et 3.
- Les entrées pour le sens de rotation sont la 5 et 7 et la PWM (*enable*) ira sur la broche 6.

Pour le second pont :

- Les sorties sont situées sur les broches 13 et 14.
- Les entrées pour le sens de rotation sont la 5 et 7 et la PWM (*enable*) ira sur la broche 6.

Pour les deux ponts :

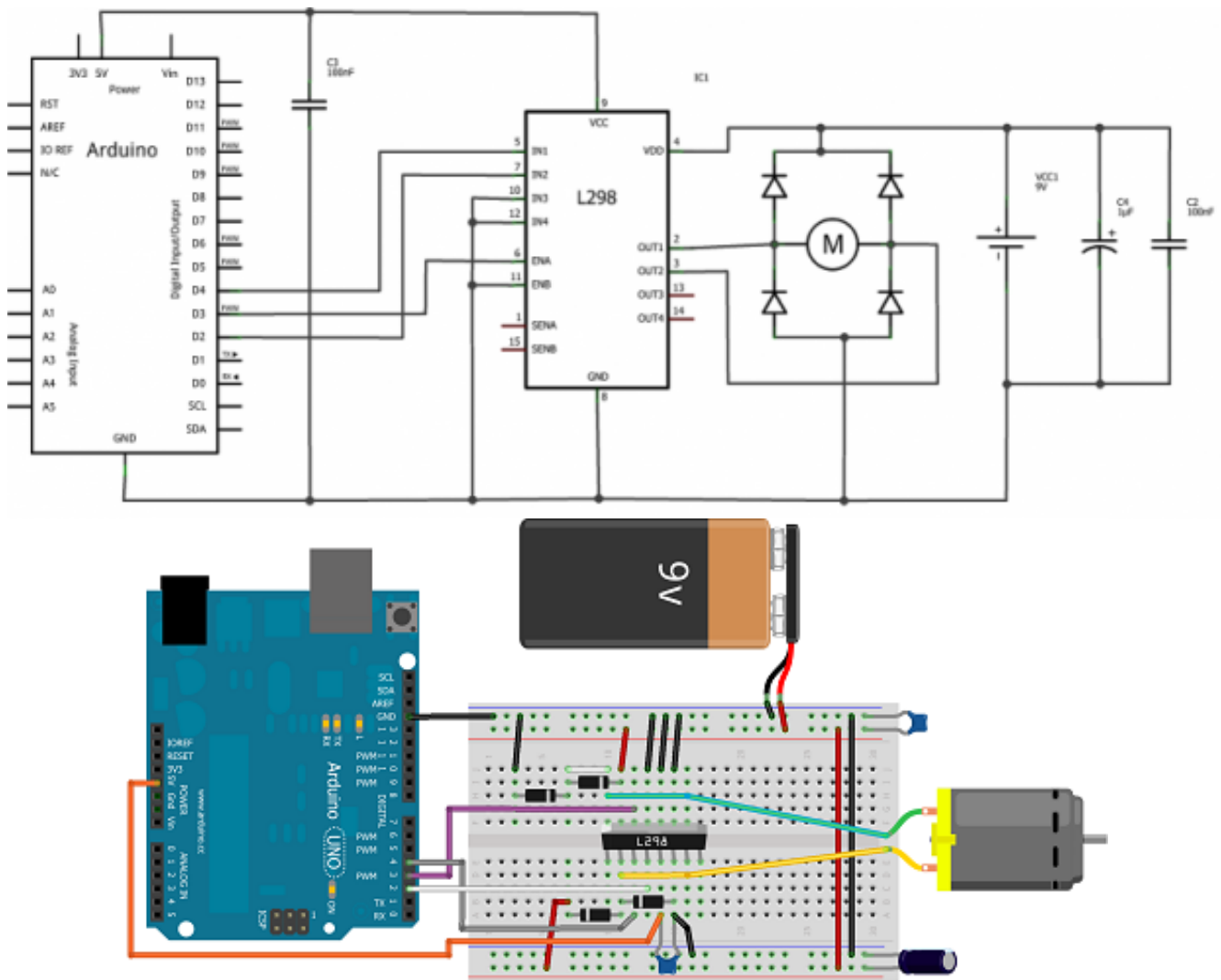
- La masse, qui est au milieu sur la broche 8.
- L'alimentation de la logique de commande (le 5V) sur la broche suivante, la 9.
- Et l'alimentation de la partie puissance sur la broche 4.

Je ne mentionne pas les broches 1 et 15 qui sont celles servant à mesurer le courant traversant les ponts. Je doute que vous vous en serviez dans un premier temps et si vous arrivez jusque là je n'ai aucun doute que vous arriverez à les mettre en oeuvre (indice : il faudra utiliser une résistance 😊)

Le L298 n'existe pas avec les diodes de roue libre intégrées. Prenez donc garde à bien les rajouter dans votre montage sous peine de voir votre composant griller.

Comme précédemment, voici un schéma d'illustration (l'image représentant le L298 n'est pas exacte, mais le boîtier multiwatt n'existe pas encore dans Fritzing donc j'ai dû

feinter) :



Et Arduino dans tout ça ?

Bref rappel sur les PWM

Si vous avez bien lu la partie précédente, vous avez dû apprendre que pour pouvoir modifier la vitesse de rotation du moteur il faut utiliser un signal PWM. Mais vous souvenez-vous comment on s'en sert avec Arduino ? Allez, zou, petite piqûre de rappel ! Commençons par redire où sont situées les broches utilisables avec la PWM. Elles sont au nombre de 6 et ont les numéros 3, 5, 6, 9, 10 et 11. Pour les utiliser, vous devrez les configurer en sortie dans le setup() de votre programme :

```
1 const int brochePWM = 3;
2
3 void setup()
4 {
5     //configuration en sortie de la broche 3
6     pinMode(brochePWM, OUTPUT);
7 }
```

Ensuite, vous pourrez agir sur le *rapport cyclique* du signal PWM (le ratio entre temps à l'état HAUT et temps à l'état BAS) en utilisant la fonction `analogWrite(broche, ratio)`. L'argument *broche* désigne... la broche à utiliser et l'argument *ratio* indique la portion de temps à l'état haut du signal.

```
1 /* le signal PWM est généré sur la broche 3 de la carte Arduino
2 avec un rapport cyclique de 50% (état HAUT égal en temps à celui de l'état BAS */
3 analogWrite(brochePWM, 127);
```

Le rapport cyclique est défini par un nombre allant de 0 à 255. Cela signifie qu'à 0, le signal de sortie sera nul et à 255, le signal de sortie sera à l'état HAUT. Toutes les valeurs comprises entre ces deux extrêmes donneront un rapport cyclique plus ou moins grand. Dans notre cas, le moteur tourne plus ou moins vite selon si le rapport cyclique est grand ou petit. Pour savoir quel rapport cyclique correspond avec quelle valeur, il faut faire une règle de trois :

Valeur argument Rapport cyclique (%)

0	0
127	50
255	100

Le calcul donnant la valeur pour chaque portion est défini par cette relation :

$$\text{argument} = \frac{x \times 100}{255}$$

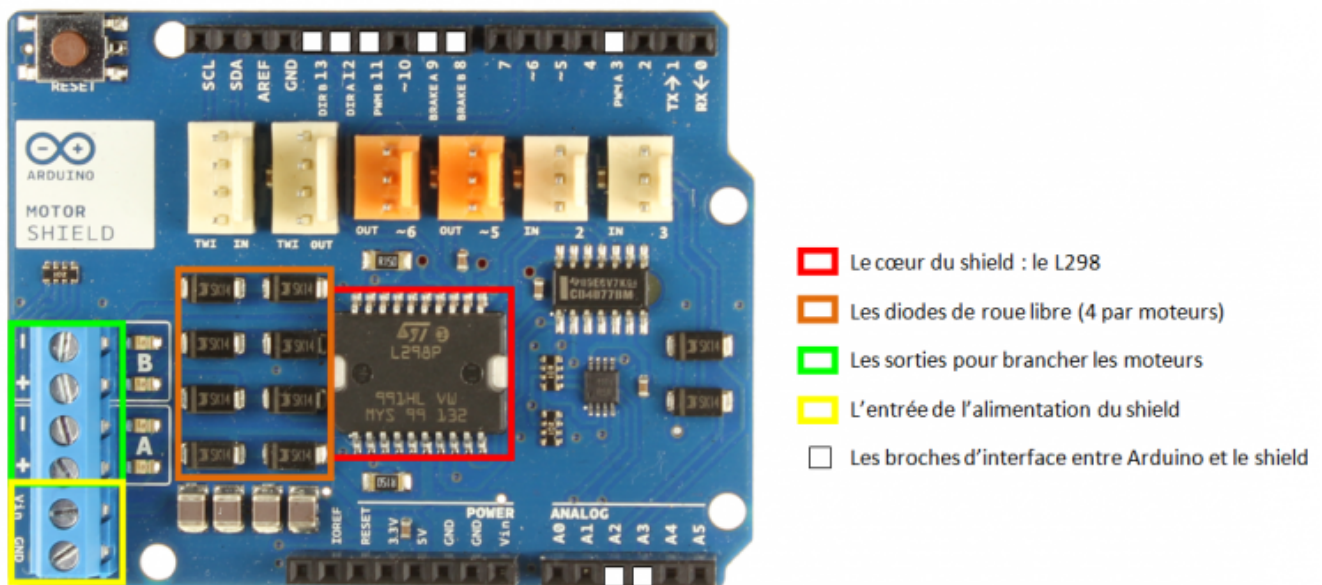
Le résultat de ce calcul donne la valeur de l'argument pour le rapport cyclique recherché. *x* est la valeur du rapport cyclique que vous souhaitez donner au signal.

Utiliser un shield moteur

Comme nous l'avons vu précédemment, réaliser un pont en H demande quelques efforts (surtout si vous désirez tout faire vous mêmes 😊). Afin de rendre ces derniers plus accessibles aux personnes ayant moins de moyens techniques (tout le monde ne dispose pas du matériel pour réaliser ses propres cartes électroniques !), l'équipe Arduino a développé et mis en productions un shield (une carte d'extension) pour pouvoir utiliser facilement des moteurs. Cette extension possède ainsi tout ce qu'il faut pour mettre en œuvre rapidement un ou des moteurs. La seule contrepartie est que les broches à utiliser sont imposées. Cependant, il existe une multitude de shields moteurs *non officiels* pouvant faire les mêmes choses ou presque. L'avantage de ces derniers est indéniablement leur prix souvent moins cher. En revanche, il n'est pas toujours facile de trouver leur documentation et le format de la carte ne se soucie pas forcément du "standard" Arduino (et n'est donc pas forcément adaptable en "s'ajoutant par dessus" comme un shield officiel le ferait). Je vais donc maintenant vous présenter le shield officiel, son fonctionnement et son utilisation, puis ensuite un shield non-officiel (acheté pas cher sur le net) que je possède et avec lequel je ferai mes photos/vidéos. Vous devriez alors avoir assez de connaissances pour utiliser n'importe quel shield non-officiel que vous pourrez trouver. Les deux shields présentés ont un point commun : ils utilisent tous les deux le L298 comme composant pour les ponts en H.

Le shield officiel d'Arduino

Tout d'abord, voici l'adresse de description de ce shield : [le shield moteur](#). Comme vous avez bien lu la partie précédente à propos du L298, vous connaissez déjà la majeure partie des choses à savoir. Parmi elles, vous savez que le L298 nécessite trois broches de "pilotage" (par pont intégré) et envoie la puissance sur deux broches (par moteur). Éventuellement nous disposons aussi des deux "sondes de courant" mais nous y reviendrons plus tard. Voici un petit synoptique de résumé que je vous ai concocté pour l'occasion : 😊



Voici comment il fonctionne et les quelques précautions d'utilisation.

- L'alimentation de puissance sur les borniers à visser à gauche est reliée à l'Arduino et peut donc lui servir de source d'alimentation. Si vous voulez dédier cette alimentation à la carte moteur, il faut donner un coup de cutter sur le strap marqué Vin en dessous de la carte
- Les entrées/sorties du shield sont reliées à l'Arduino de la manière suivante :

Fonction	Broches mot. A	Broches mot. B
Direction	12	13
PWM	3	11
Frein	9	8
Mesure de courant A0		A1

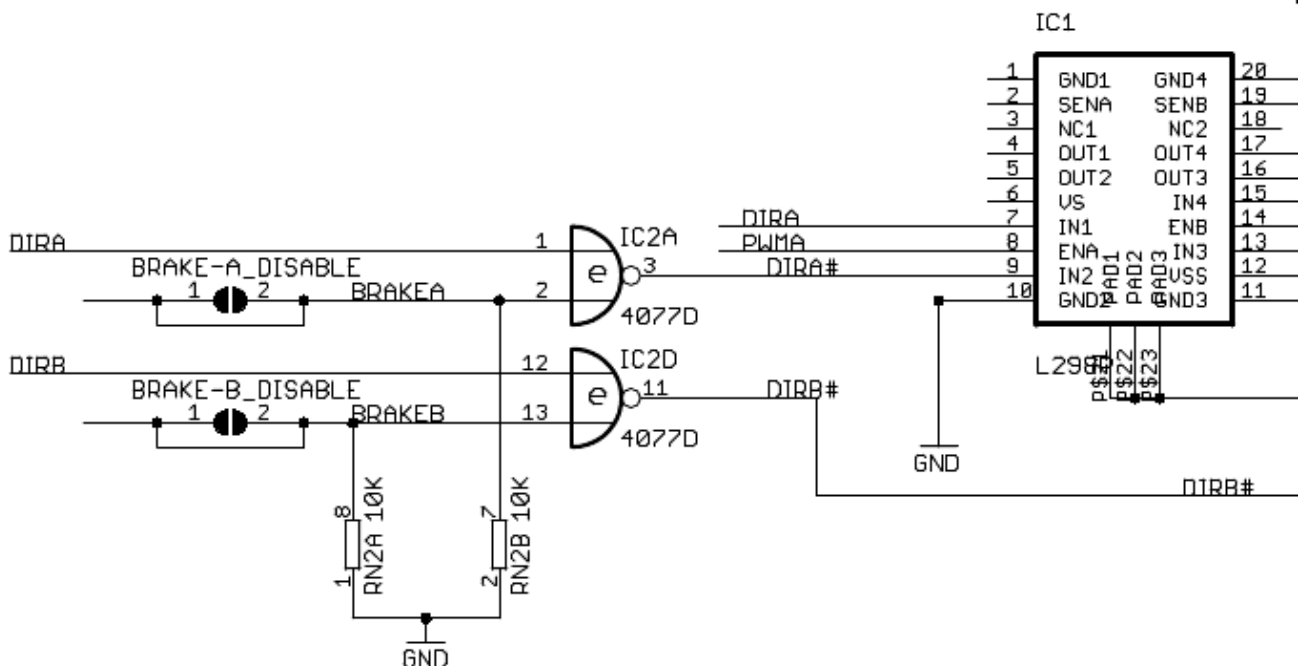
La mesure de courant se fait sur les broches A0 et A1. Si vous avez besoin de ces broches pour d'autre applications, vous pouvez là encore désactiver la fonction en coupant le strap en dessous de la carte. Sinon, la mesure se fera simplement avec la fonction `analogRead(canal)`. Le circuit a été calibré de façon à ce qu'il y ait 3.3V lorsque le pont délivre 2A (le maximum). Cette relation est proportionnelle, on a donc un rapport de 1.65V par Ampère. Comme l'équipe d'Arduino sait pertinemment que le nombre de broches est limité, ils ont utilisé un montage électronique supplémentaire, pour gagner une broche par pont. En effet, grâce à un petit montage avec une [porte logique OU Exclusif](#), on peut désactiver la fonction de "frein" tout en gardant celle du sens. Grâce à cela, on peut se limiter à seulement deux broches pour commander chaque moteur : celle du sens et celle de la vitesse. Voici comment ils ont fait : Tout d'abord, regardons la table de vérité du OU EXCLUSIF. Cette dernière s'interprète comme suit : "La sortie est à 1 si une des deux entrées **uniquement** est à 1". Sous forme de tableau on obtient

ça:

Entrée A Entrée B Sortie

0	0	0
1	0	1
0	1	1
1	1	0

Maintenant rappelez-vous, les conditions de freinage étaient justement représentées lorsque les deux entrées du pont étaient au même niveau. En couplant intelligemment le résultat de cette porte logique et les entrées de pilotage, on peut décider oui ou non d'avoir la fonction de frein. Afin de mieux comprendre, je vous invite à consulter cet extrait du schéma technique du shield :



Grâce à ce montage, vous pouvez choisir ou non d'avoir un mode de frein sur vos moteurs. Si vous préférez avoir deux broches disponibles et ne pas avoir de frein (juste une roue libre lorsque la PWM est à 0), alors il vous suffira une fois de plus de couper les straps en dessous de la carte.

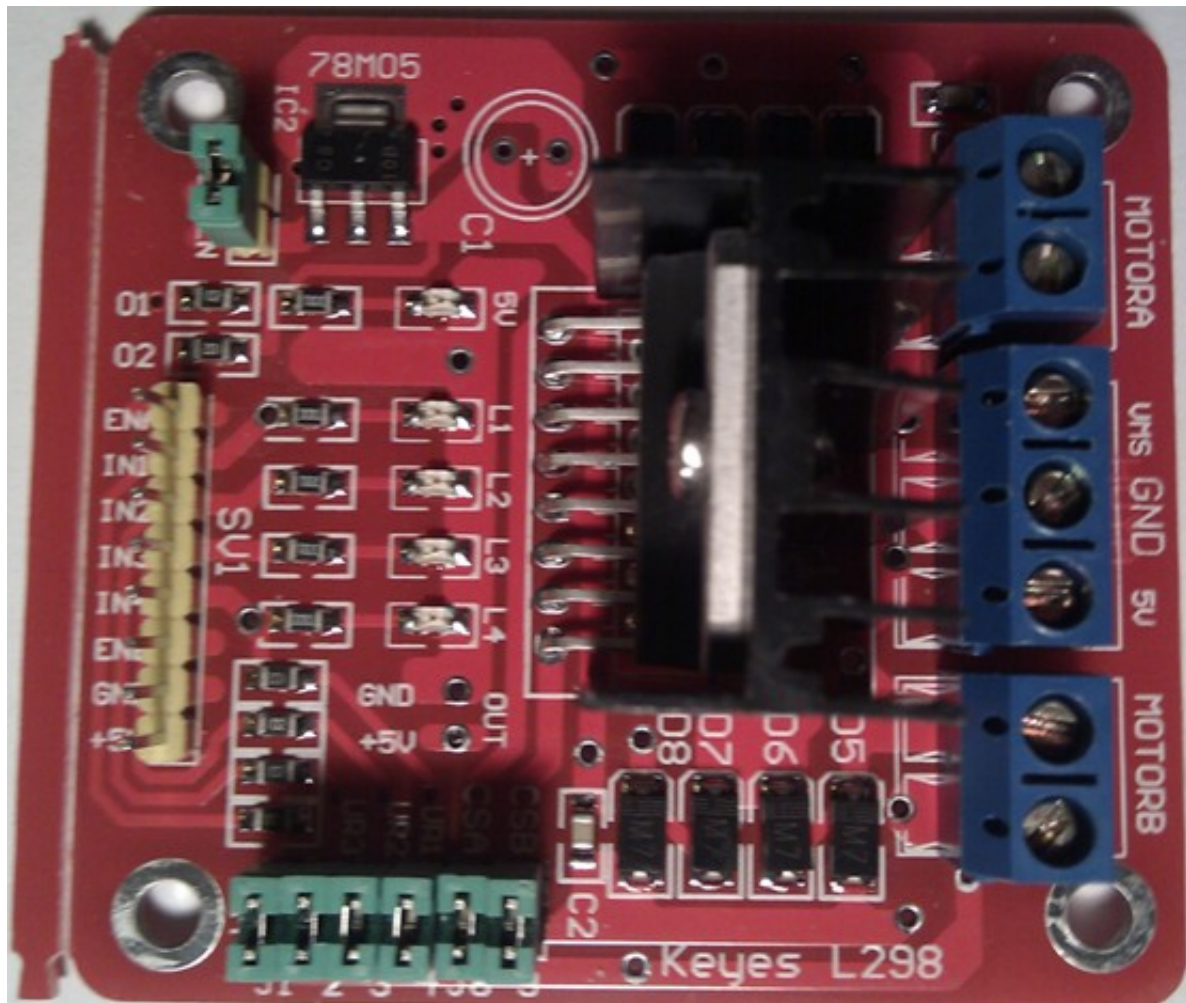
N'ayez pas peur d'avoir des regrets ! Si vous coupez un strap, vous pourrez toujours le remettre en ajoutant un petit point de soudure pour relier les deux pastilles prévues à cet effet. 😊 Le mieux aurait été d'avoir la possibilité de mettre des cavaliers que l'on enlève à la main, mais bon, c'est comme ça.

Vous savez maintenant tout à propos de ce shield. Je vais maintenant vous en présenter un non-officiel et ensuite nous passerons à un petit montage/code d'exemple pour finir ce chapitre.

Mon shield non-officiel

Maintenant que vous connaissez le fonctionnement global du shield officiel, vous allez

pouvoir utiliser sans problème la plupart des shields moteurs. Afin de ne pas faire de publicité pour un site ou un autre, je vais vous présenter mon shield qui vaut aussi bien qu'un autre (mais pas forcément mieux). Il n'y a aucun parti pris, j'ai acheté ce dernier afin de profiter de tarif intéressant lors d'une commande avec d'autres composants. Si j'avais été uniquement à la recherche d'un shield moteur, j'en aurais peut-être pris un autre qui sait ! Bref, assez de ma vie, passons à l'étude du module ! Afin de bien commencer les choses, je vais d'abord vous montrer une photo d'identité de ce dernier. Ensuite je vous expliquerai où sont les broches qui nous intéressent et ferai un parallèle avec le shield officiel. Les deux étant basés sur un L298 l'explication sera assez rapide car je n'ai pas envie de me répéter. Je ferai néanmoins un petit aparté sur les différences (avantages et inconvénients) entre les deux.



Voici une petite liste des points importants :

- À gauche en **jaune** : les entrées de commande. **EnA**, **In1**, **In2** pour le moteur A ; **EnB**, **In3**, **In4** pour le moteur B. On trouve aussi une broche de masse et une sortie 5V sur laquelle je reviendrai.
- En bas en **vert** différents *jumpers* (des cavaliers si vous préférez 😊) pour activer des résistances de pull-down (force une entrée/sortie à l'état bas) et câbler la mesure de courant de sortie des ponts
- À droite en **bleu**, les bornes pour brancher les moteurs A et B (respectivement en haut et en bas) et au milieu le bornier pour amener l'alimentation de puissance (et une entrée ou sortie) de 5V

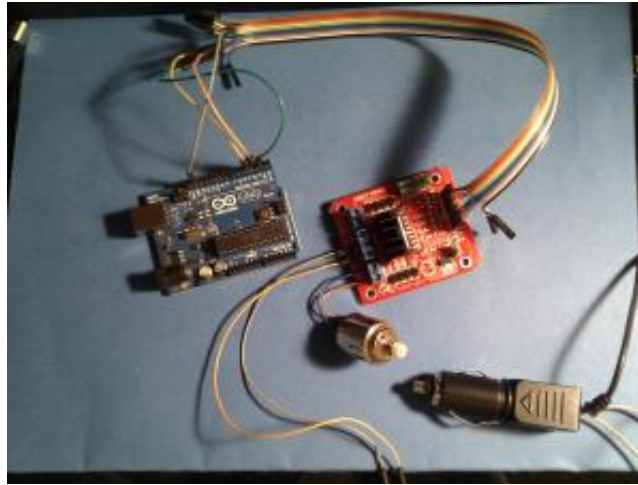
Au milieu on retrouve le L298 avec de chaque côté (en haut et en bas) les diodes de roue libre pour chaque moteur. Une petite précision s'impose par rapport à ce shield. La carte embarque un régulateur 5V (le petit bloc noir en haut à gauche marqué 78M05). Ce dernier peut être utilisé ou non (Activez-le avec le jumper vert juste à côté). Si vous le laissez activé, c'est lui qui fournira l'alimentation pour la logique du L298. Si vous le désactivez, vous devrez fournir vous-même le 5V pour la logique. Dans tous les cas, il vous faut relier les masses puissances et logiques entre Arduino et le shield afin d'avoir un référentiel commun. Si vous l'activez, alors vous obtiendrez une sortie de 5V sur le bornier bleu à droite (utile pour alimenter l'Arduino par exemple). Si vous le désactivez, alors vous devrez fournir le 5V (et donc le bornier bleu devra être utilisé comme une entrée). Ce shield n'est en fait qu'une simple carte électronique disposant du L298 et facilitant l'accès à ses broches. Le fonctionnement se fait exactement comme nous l'avons vu dans le chapitre précédent, lorsque je vous présentais le L293 et L298 pour la première fois. Pas de facétie avec des portes logiques pour gagner des broches. Ici, tout est brut de décoffrage, on commande directement le pont en H. Il vous faudra donc trois broches par moteur, deux pour gérer la direction et le frein et une (PWM) pour la vitesse.

Petit programme de test

Nous allons maintenant pouvoir passer aux choses sérieuses : l'utilisation du moteur avec l'Arduino !

L'électronique

Pour cela, nous allons commencer par câbler le shield. En ayant la partie précédente concernant le vôtre sous les yeux, vous devriez pouvoir vous en sortir sans trop de difficulté. (Désolé, pas de schéma ce coup-ci car le logiciel que j'utilise ne possède pas encore le shield moteur dans sa base de données, faites donc preuve d'imagination 😊). Personnellement, je n'utiliserai qu'un seul moteur (car dans l'immédiat j'en ai qu'un sous la main 😊). Je vais donc le brancher sur les bornes bleues "Moteur A". Ensuite, je vais relier les différentes broches de commande à mon Arduino. La broche EnA sera reliée à une sortie de PWM (dans mon cas la broche 3) et les broches In1 et In2 seront reliées à n'importe quelles broches numériques (2 et 4 pour moi). Il ne nous reste plus qu'à nous occuper de l'alimentation. Tout d'abord, je mets un fil entre la masse du shield et celle de l'Arduino (pour avoir un référentiel commun). Comme ma carte possède son propre régulateur de tension 5V, pas besoin de l'amener depuis Arduino. Enfin, je relie les deux fils pour la puissance. Dans mon cas ce sera une alimentation 12V (400 mA max, wouhou) qui vient d'un adaptateur allume-cigare (censé fournir du 5V) que j'ai démonté pour obtenir une source de 12V. Je vous propose aussi de rajouter un potentiomètre sur une entrée analogique. De cette façon nous allons pouvoir faire varier la vitesse sans recharger le programme 😊. Et voilà, point de vue électronique tout est prêt. Voilà ce que ça donne chez moi (un beau bazar 😊, mais j'ai oublié le potentiomètre) :



L'informatique

Maintenant, nous allons devoir nous occuper du code et comme toujours, nous commençons par lister les variables concernant les broches utilisées :

```
1 //la PWM pour la vitesse
2 const int enable = 3;
3 //les broches de signal pour le sens de rotation
4 const int in1 = 2;
5 const int in2 = 4;
6
7 //une entrée analogique (A0) pour régler la vitesse manuellement
8 const int potar = 0;
```

Ces différentes broches seront bien entendu des broches de sortie (sauf l'analogique), donc nous les déclarons comme telles dans le `setup()` :

```
1 void setup()
2 {
3     pinMode(enable, OUTPUT);
4     pinMode(in1, OUTPUT);
5     pinMode(in2, OUTPUT);
6     //j'utilise la liaison série pour voir la vitesse définie par le potentiomètre
7     Serial.begin(115200);
8
9     //on démarre moteur en avant et en roue libre
10    analogWrite(enable, 0);
11    digitalWrite(in1, LOW);
12    digitalWrite(in2, HIGH);
13 }
```

Et voilà, si vous exécutez le code maintenant votre moteur sera... arrêté ! Eh oui, j'ai volontairement mis une vitesse nulle à la fin du `setup()` pour éviter que le moteur ne s'emballé au démarrage du programme. Mais si vous changez cette dernière (mettez 50 pour voir) vous verrez votre moteur se mettre à tourner. Nous allons donc rajouter un peu d'interactivité, pour que vous puissiez vous-même augmenter/diminuer la vitesse en fonction de la valeur lue sur le potentiomètre :

```
1 void loop()
2 {
3     //on lit la valeur du potentiomètre
4     int vitesse = analogRead(potar);
5 }
```



```

6 //division de la valeur lue par 4
7 vitesse /= 4;
8
9 //envoie la nouvelle vitesse sur le moteur
10 analogWrite(enable, vitesse);
11
12 //on affiche la vitesse sur le moniteur série
13 Serial.println(vitesse);
14
15 delay(50);
16 }

```

Mais pourquoi tu divises la vitesse par 4 à la ligne 5 ? Je veux aller à fond moi !

C'est très simple. La lecture analogique nous renvoie une valeur entre 0 et 1023 (soit 1024 valeurs possibles). Or la fonction analogWrite ne peut aller qu'entre 0 et 255 (total de 256 valeurs). Je divise donc par 4 pour rester dans le bon intervalle ! Car : $4 \times 256 = 1024$.

Programme plus élaboré

Maintenant, je vous fais cadeau d'un code vous permettant d'aller dans les deux sens et à vitesse variable. Mais, je vous conseille d'essayer de le faire par vous-même avant de regarder ce qu'il y a dans la balise secret. Le potentiomètre est utilisé comme régulateur de vitesse, mais on va virtuellement décaler l'origine. Autrement dit, entre 0 et 511 nous irons dans un sens, et entre 512 et 1023 nous irons dans l'autre sens. Nous ferons aussi en sorte que la vitesse soit de plus en plus élevée lorsque l'on "s'éloigne" du 0 virtuel (de la valeur 512 donc). Je vous donne le code tel quel (avec des commentaires bien sûr). Libre à vous de le traiter comme un exercice. À sa suite, une petite vidéo du résultat.

```

1  const int enable = 3; //la PWM
2  const int in1 = 2;    //les broches de signal
3  const int in2 = 4;
4  const int potar = 0; //la broche pour régler la vitesse
5
6  void setup()
7  {
8      pinMode(enable, OUTPUT);
9      pinMode(in1, OUTPUT);
10     pinMode(in2, OUTPUT);
11     Serial.begin(115200);
12
13     //on démarre moteur en avant et en roue libre
14     analogWrite(enable, 0);
15     digitalWrite(in1, LOW);
16     digitalWrite(in2, HIGH);
17 }
18
19 void loop()
20 {
21     int vitesse = analogRead(potar);
22
23     //dans le sens positif
24     if(vitesse > 512)
25     {
26         //on décale l'origine de 512

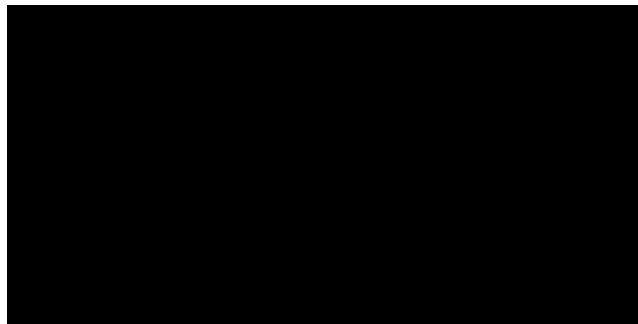
```

```

27     vitesse -= 512;
28     //le moteur va dans un sens
29     digitalWrite(in1, LOW);
30     digitalWrite(in2, HIGH);
31     Serial.print("+");
32 }
33 else //dans l'autre sens
34 {
35     //de même on décale pour que la vitesse augmente en s'éloignant de 512
36     vitesse = 512-vitesse;
37     //le moteur va dans l'autre sens
38     digitalWrite(in1, HIGH);
39     digitalWrite(in2, LOW);
40     Serial.print("-");
41 }
42
43 //pour rester dans l'intervalle [0;255] (sinon on est dans [0;512])
44 vitesse /= 2;
45 //envoie la vitesse
46 analogWrite(enable, vitesse);
47
48 //et l'affiche
49 Serial.println(vitesse);
50 delay(50);
51 }

```

Bravo à ceux qui ont essayé de faire ce programme, même s'ils n'y sont pas arrivés ! Dans ce dernier cas, vous pouvez aller voir sur les forums et poser vos éventuelles questions après avoir vérifié que vous avez bien tout essayé de comprendre. 😊 Voilà la vidéo qui montre le fonctionnement du programme :



Désolé pour la qualité de la vidéo, il faut vraiment que je change d'appareil...

Vous savez désormais comment fonctionne un moteur à courant continu et quels sont les moyens de le piloter. Il va dorénavant être possible de vous montrer l'existence de moteurs un peu particuliers qui se basent sur le moteur à courant continu pour fonctionner. Et vous allez voir que l'on va pouvoir faire plein de choses avec ! 😊

[Arduino 602] Un moteur qui a de la tête : le Servo-Moteur

Dans ce chapitre, nous allons parler d'un moteur que nos amis modélistes connaissent bien : le **Servomoteur** (abrégé : "servo"). C'est un moteur un peu particulier, puisqu'il confond un ensemble de mécanique et d'électronique, mais son principe de fonctionnement reste assez simple. Les parties seront donc assez courtes dans

l'ensemble car les servomoteurs contiennent dans leur "ventre" des moteurs à courant continu que vous connaissez à présent. Cela m'évitera des explications supplémentaires. 😊

Principe du servo-moteur

Un servomoteur... Étrange comme nom, n'est-ce pas ? Cela dit, il semblerait qu'il le porte bien puisque ces moteurs, un peu particuliers je le disais, emportent avec eux une électronique de commande (faisant office de "cerveau"). Le nom vient en fait du latin *servus* qui signifie esclave. Mais avant de s'atteler à l'exploration interne de ce cher ami, façon de parler, nous allons avant tout voir à quoi il sert.

Vue générale

Le servo, un drôle de moteur

Commençons en image, avec la photographie d'un servomoteur :



C'est, en règle générale, à quoi ils ressemblent, variant selon leur taille.

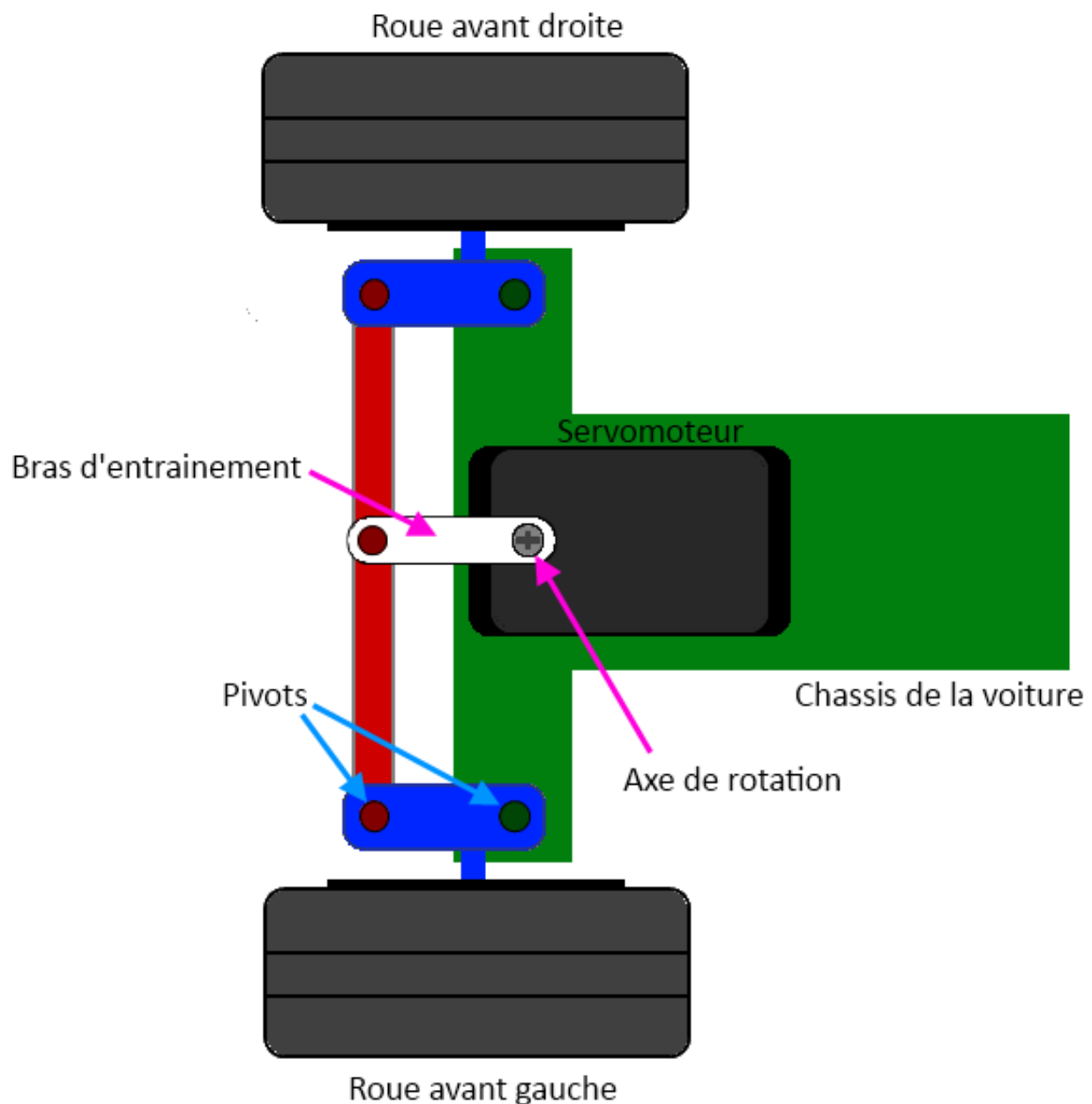
Pfiouuu, c'est quoi ce moteur, ça n'y ressemble même pas ! 😊

J'veus l'avais dit que c'était des moteurs particuliers ! En détail, voyons à quoi ils servent. De manière semblable aux moteurs à courant continu, les servomoteurs disposent d'un axe de rotation. Sur la photo, il se trouve au centre de la roue blanche. Cet axe de rotation est en revanche entravé par un système de bridage. Cela ne veut pas dire qu'il ne tourne pas, mais cela signifie qu'il ne peut pas tourner au delà d'une certaine limite. Par exemple, certains servomoteurs ne peuvent même pas faire tourner leur axe de rotation en leur faisant faire un tour complet ! D'autres en sont capables, mais pas plus d'un tour. Enfin, un cas à part que nous ne ferons qu'évoquer, ceux qui tournent sans avoir de limite (autant de tours qu'ils le veulent). Et là, c'est le moment où je vous dis : *"détrompez-vous !"* en répondant à la question critique que vous avez en tête : *"Un moteur qui ne peut même pas faire un tour avec son axe de rotation, ça sert à rien ? o_O"* En effet, s'il ne peut pas faire avancer votre robot, il peut cependant le

guider. Prenons l'exemple d'une petite voiture de modélisme à quatre roues. Les roues arrière servent à faire avancer la voiture, elles sont mises en rotation par un moteur à courant continu, tandis que les roues avant, qui servent à la direction de la voiture pour ne pas qu'elle se prenne les murs, sont pilotées par un servomoteur. Comment ? Eh bien nous allons vous l'expliquer.

L'exemple de la voiture radiocommandée

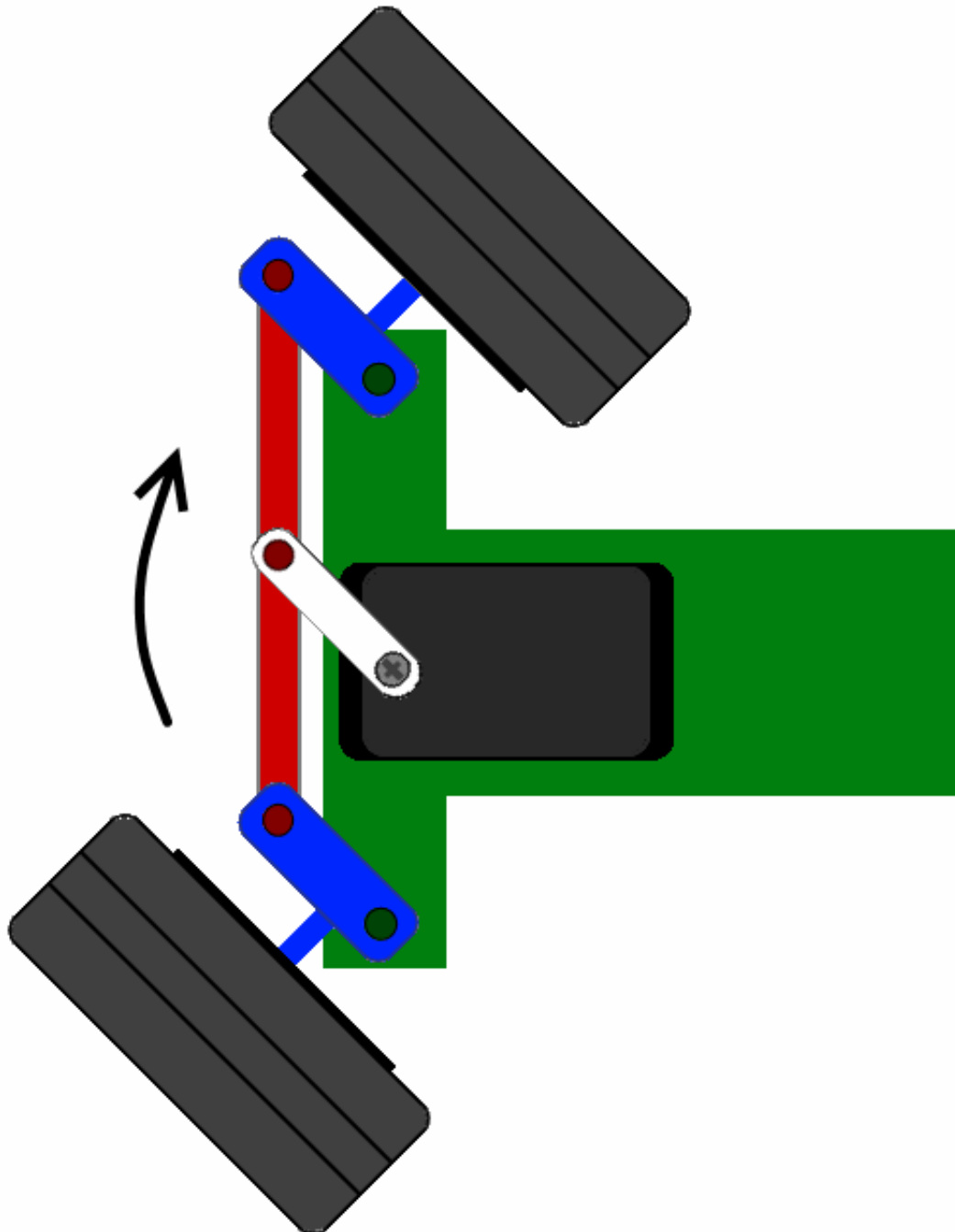
Regardons l'image que je vous ai préparée pour comprendre à quoi sert un servomoteur :



Vue de dessus Représentation schématique du système de guidage des roues d'une voiture radiocommandée

Chaque roue est positionnée sur un axe de rotation (partie bleue) lui même monté sur un pivot sur le châssis de la voiture (en vert). La baguette (rouge) permet de garder la parallélisme entre les roues. Si l'une pivote vers la gauche, l'autre en fait de même (ben ouais, sinon la voiture devrait se couper en deux pour aller dans les deux directions

opposées 😊). Cette baguette est fixée, par un pivot encore, au bras de sortie du servomoteur. Ce bras est à son tour fixé à l'axe de rotation du servomoteur. Ainsi, lorsque le servomoteur fait tourner son axe, il entraîne le bras qui entraîne la baguette et fait pivoter les roues pour permettre à la voiture de prendre une direction dans son élan (tourner à gauche, à droite, ou aller tout droit). Il n'y a rien de bien compliqué. Ce qu'il faut retenir est que le servomoteur va entraîner la baguette pour orienter les roues dans un sens ou dans l'autre. Elles auront donc un angle d'orientation par rapport au châssis de la voiture. Voyez plutôt :



Alors, vous allez me dire : *“mais pourquoi on met pas un moteur à courant continu avec un bras sur son axe, ce serait plus simple, non ?”* Eh bien non car cela ne conviendrait

pas. Je vous explique pourquoi. Nous l'avons vu, un moteur à courant continu tourne sans s'arrêter, sauf si on lui coupe l'alimentation. Le problème c'est que, dans notre cas, si on laisse le moteur tourner, il pourrait faire pivoter les roues plus loin que leur angle maximal et casser le système de guidage car il ne saura pas quand il faut s'arrêter (à savoir, quand les roues sont arrivées à leur angle maximal). Bon, on pourrait très bien faire un système qui coupe l'alimentation quand les roues arrivent sur leur butée. En plus, les moteurs à courant continu sont de bien piètres athlètes, il leur faudrait nécessairement un réducteur pour arriver à avoir une vitesse faible et un couple plus élevé. Mais pourquoi s'embêter avec ça plutôt que d'utiliser quelque chose de déjà tout prêt ? C'est le servomoteur qui va faire tout ça ! Pour être précis, le servomoteur est commandé de telle sorte qu'au lieu de donner une vitesse de rotation de son axe, il donne une position angulaire de l'arbre relié à son axe. Donc, on lui demande de faire tourner son axe de 10° vers la gauche et il s'exécute !

Composition d'un servomoteur

Les servomoteurs ont donc l'avantage d'être *asservis* en **position angulaire**. Cela signifie, je vous l'expliquais, que l'axe de sortie du servomoteur respectera une consigne d'orientation que vous lui envoyez en son entrée. En plus, tenez-vous bien, si par malheur les roues venaient à changer d'orientation en passant sur un caillou par exemple, l'électronique interne du servomoteur essaiera tant bien que mal de conserver cette position ! Et quelle que soit la force que l'on exerce sur le bras du servomoteur, il essaiera de toujours garder le même angle (dans les limites du raisonnable évidemment). En quelque sorte vous ne pilotez pas directement le moteur, mais plutôt **vous imposez le résultat que vous voulez avoir en sortie**.

Apparence

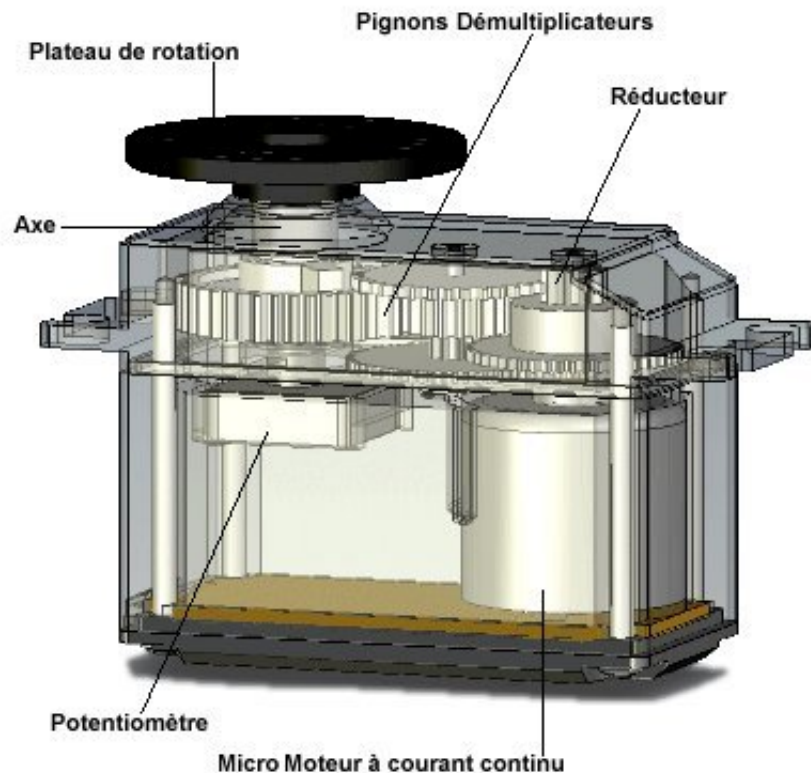
On en trouve de toutes les tailles et de toutes les puissances. La plupart du temps la sortie peut se positionner entre 0 et 180° . Cela dit, il en existe également dont la sortie peut se débattre sur seulement 90° et d'autres, ayant un plus grand débattement, sur 360° . Ceux qui ont la possibilité de faire plusieurs tours sont souvent appelés **servotreuils**. Enfin, les derniers, qui peuvent faire tourner leur axe sans jamais se buter, sont appelés **servomoteurs à rotation continue**. Les servomoteurs sont très fréquemment employés dans les applications de modélisme pour piloter le safran d'un bateau, le gouvernail d'un avion ou bien même les roues d'une voiture téléguidée dont on a parlé jusqu'à présent. Maintenant que les présentations sont faites, mettons-le à nu ! Il est composé de plusieurs éléments visibles ... :

- Les fils, qui sont au nombre de trois (nous y reviendrons)
- L'axe de rotation sur lequel est monté un accessoire en plastique ou en métal
- Le boîtier qui le protège

... mais aussi de plusieurs éléments que l'on ne voit pas :

- un moteur à courant continu
- des engrenages pour former un réducteur (en plastique ou en métal)
- un capteur de position de l'angle d'orientation de l'axe (un potentiomètre bien souvent)
- une carte électronique pour le contrôle de la position de l'axe et le pilotage du moteur à courant continu

Voilà une image 3D (extraite du [site internet suivant](#)) de vue de l'extérieur et de l'intérieur d'un servomoteur :



Site G

Connectique

Le servomoteur a besoin de trois fils de connexion pour fonctionner. Deux fils servent à son alimentation, le dernier étant celui qui reçoit le signal de commande :

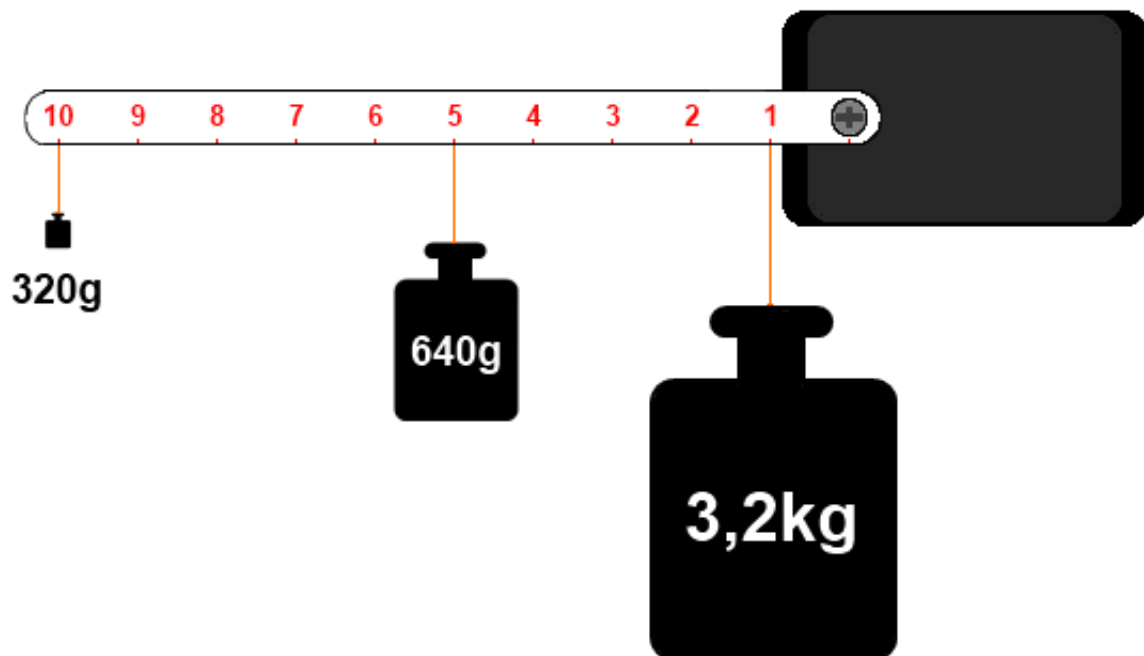
- **rouge** : pour l'alimentation positive (4.5V à 6V en général)
- **noir** ou **marron** : pour la masse (0V)
- **orange**, **jaune**, **blanc**, ... : entrée du signal de commande

Nous verrons tout à l'heure ce que nous devons entrer sur le dernier fil.

La mécanique

Comme on le voit dans l'image précédente, le servomoteur possède plusieurs pignons (engrenages) en sortie du petit moteur CC. Cet ensemble est ce qui constitue le **réducteur**. Ce réducteur fait deux choses : d'une part il réduit la vitesse de rotation en sortie de l'axe du servomoteur (et non du moteur CC), d'autre part il permet d'augmenter le couple en sortie du servomoteur (là encore non en sortie du moteur CC). Alors, à quoi ça sert de réduire la vitesse et d'augmenter le couple ? Eh bien les moteurs CC se débrouillent très bien pour tourner très vite mais lorsqu'ils font une si petite taille ils sont bien moins bons pour fournir du couple. On va donc utiliser ce réducteur qui va réduire la vitesse, car nous n'avons pas besoin d'avoir une vitesse trop élevée, et augmenter le couple pour ainsi pouvoir déplacer une charge plus lourde. Ceci est prouvé par la formule que je vous ai donnée dans le chapitre précédent : $R = \frac{\omega_{entree}}{\omega_{sortie}} = \frac{C_{sortie}}{C_{entree}}$. Le rapport de réduction (R) du réducteur définit le couple et la vitesse de sortie (en sortie

du réducteur) selon la vitesse et le couple d'entrée (en sortie du moteur CC). Ces données sont souvent transparentes lorsque l'on achète un servomoteur. Dans la quasi totalité des cas, nous n'avons que la vitesse angulaire (en degré par seconde °/s), le couple de sortie du servomoteur et le débattement maximal (s'il s'agit d'un servomoteur ayant un débattement de 0 à 90°, 180, 360 ou autre). Et c'est largement suffisant étant donné que c'est ce qui nous intéresse dans le choix d'un servomoteur. Il y a cependant une unité qui pourra peut-être vous donner quelques doutes ou une certaine incompréhension. Cette caractéristique est celle du couple du servomoteur et a pour unité le *kg.cm* (kilogramme-centimètre). Nous allons tout de suite rappeler ce que cela signifie. Avant tout, rappelons la formule suivante : $C = F \times r$ qui donne la relation entre le couple C du servomoteur (en kilogramme mètre), F la force exercée sur le bras du servomoteur (en kilos) et r la distance (en m) à laquelle s'exerce cette force par rapport à l'axe de rotation du servomoteur. Disséquons dans notre langage la signification de cette formule : le couple (C) exercé sur un axe est égal à la force (F) appliquée au bout du levier accroché à ce même axe. À force identique, plus le levier est long et plus le couple exercé sur cet axe est important. En d'autres termes, si votre servomoteur dispose d'un bras d'un mètre de long (oui c'est très long) eh bien il aura beaucoup plus de difficultés à soulever une charge de, disons 10g, que son homologue qui supporte la même charge avec un bras nettement raccourci à 10 centimètres. Prenons l'exemple d'un servomoteur assez commun, le [Futaba_s3003](#). Sa documentation nous indique que lorsqu'il est alimenté sous 4.8V (on reviendra dessus plus tard), il peut fournir un couple (*torque* en anglais) de $3,2\text{kg.cm}$. C'est à dire, qu'au bout de son bras, s'il fait 1 centimètre, il pourra soulever une charge de 3,2kg. Simple, n'est-ce pas ? 😊 Si le bras fait 10 centimètres, vous aurez compris que l'on perd 10 fois la capacité à soulever une masse, on se retrouve alors avec un poids de 320g au maximum (sans compter le poids du bras lui-même, certes négligeable ici, mais parfois non).



Voilà une image qui permet d'illustrer un peu ce que je vous raconte depuis tout à l'heure (ça commençait à être ennuyeux, non ?). Bref. Ici, chaque poids représenté est celui maximum que peut soulever le servomoteur selon la distance à laquelle il est situé. Et ne vous avisez pas de les mettre tous car votre pauvre servo serait bien dans

l'incapacité de les soulever en même temps. Et oui, malgré le fait qu'il n'y ait que 320g au bout du bras, le servo voit comme s'il y avait un poids de 3,2kg ! Dans cette situation on aurait trois fois 3,2kg, ce qui ferait un poids total de 9,6kg ! Impossible pour le servo de ne bouger ne serait-ce que d'un millimètre (vous risqueriez fort de le détruire d'ailleurs).

Bon, d'accord, je comprends, mais et le zéro il y est pas sur ton dessin. Comment je sais quel poids je peux mettre sur l'axe du moteur ? o_O

Eh bien tout dépend du diamètre de cet axe. Voilà une question pertinente ! Alors, oui, répondons à la question. Mais avant, vous devriez avoir une idée de la réponse que je vais vous donner. Non ? Ben si, voyons ! Plus on éloigne le poids de l'axe et plus celui-ci diminue, et cela fonctionne dans l'autre sens : plus on le rapproche, plus sa valeur maximale augmente. En théorie, si on se met à 0cm, on pourrait mettre un poids infini. Admettons, plus rigoureusement, que l'on mette le poids à 1mm de l'axe (soit un axe de diamètre 2mm). Le poids que le servo pourrait soulever serait de... 10 fois plus ! Soit 32kg !! En conclusion, on peut admettre la formule suivante qui définit le poids maximal à mettre à la distance voulue :

$$P_{max} = \frac{C}{d}$$

Avec :

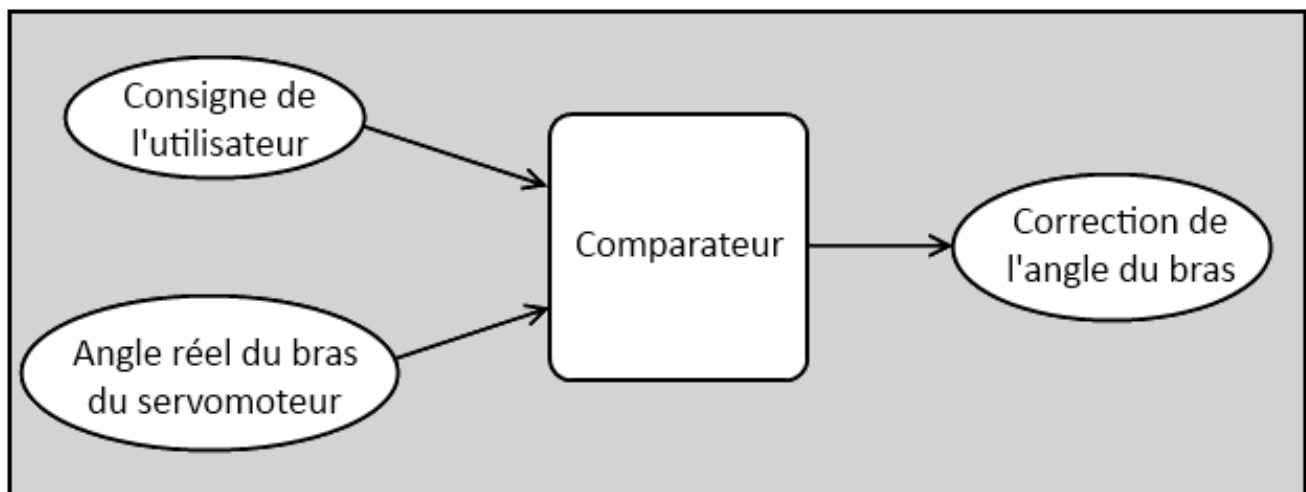
- P_{max} : poids maximal de charge en kilogramme (kg)
- C : couple du servomoteur, en kilogramme centimètre (kg.cm)
- d : distance à laquelle le poids est placé en centimètre (cm)

Et si on se concentrait sur le pourquoi du servomoteur, car son objectif principal est avant tout de donner une position angulaire à son bras. Allez, voyons ça tout de suite !

L'électronique d'asservissement

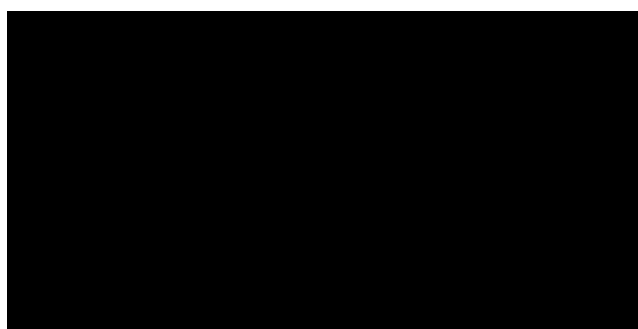
"Qu'est-ce que l'asservissement ?", vous demandez-vous sans doute en ce moment. Malgré la signification peu intuitive que ce terme porte, il se cache derrière quelque chose de simple à comprendre, mais parfois très compliqué à mettre en œuvre. Heureusement, ce n'est pas le cas pour le servomoteur. Toutefois, nous n'entrerons pas dans le détail et nous nous contenterons de présenter le fonctionnement. L'asservissement n'est ni plus ni moins qu'un moyen de gérer une consigne de régulation selon une commande d'entrée. Euuuh, vous me suivez ? 🤔 Prenons l'exemple du servomoteur : on l'alimente et on lui envoie un signal de commande qui permet de définir à quel angle va se positionner le bras du servomoteur. Ce dernier va s'exécuter. Essayez de forcer sur le bras du servomoteur... vous avez vu ? Quelle que soit la force que vous exercez (dans les limites du raisonnable), le servo va faire en sorte de toujours garder la position de son bras à l'angle voulu. Même si le poids est largement supérieur à ce qu'il peut supporter, il va essayer de remettre le bras dans la position à laquelle il se trouvait (à éviter cependant). Ainsi, si vous changez l'angle du bras en forçant dessus, lorsque vous relâcherez le bras, il va immédiatement reprendre sa position initiale (celle définie grâce au signal de commande). Pour pouvoir réaliser le maintien de la position du bras de manière correcte, le servo utilise une **électronique de commande**. On peut la nommer **électronique d'asservissement**, car c'est elle qui

va gérer la position du bras du servomoteur. Cette électronique est constituée d'une zone de comparaison qui compare (étonnamment 🤖) la position du bras du servo au signal de commande. Le deuxième élément qui constitue cette électronique, c'est le capteur de position du bras. Ce capteur n'est autre qu'un potentiomètre couplé à l'axe du moteur. La mesure de la tension au point milieu de ce potentiomètre permet d'obtenir une tension image de l'angle d'orientation du bras. Cette position est ensuite comparée, je le disais, à la consigne (le signal de commande) qui est transmise au servomoteur. Après une rapide comparaison entre la consigne et valeur réelle de position du bras, le servomoteur (du moins son électronique de commande) va appliquer une correction si le bras n'est pas orienté à l'angle imposé par la consigne.



Synoptique de fonctionnement de l'asservissement du servomoteur

Afin de garder la position de son bras stable, il est donc important de savoir quelle est la charge maximale applicable sur le bras du servomoteur. En somme, bien vérifier que le poids de la charge que vous comptez mettre sur votre servomoteur ne dépasse pas celui maximal qu'il peut supporter. Avant de passer à la suite, je vous propose de regarder cette superbe vidéo que j'ai trouvée par hasard sur [ce site web](#). Vous allez pouvoir comprendre au mieux le fonctionnement de la mécanique du servomoteur :



Mais au fait, comment est transmise la consigne de commande de position du bras ? On lui dit par la liaison série ?

C'est ce que nous allons voir tout de suite dans la partie suivante. En avant !

La commande d'un servo-moteur

Ce qu'il est intéressant de découvrir à présent, c'est de savoir comment piloter un moteur de ce type. Et oui car cela n'a pas beaucoup de ressemblances avec le moteur à courant continu. Il ne va pas être question de pont en H ou autres bizarreries de ce type, non, vous allez voir, ça va être très simple.

Sachez toutefois qu'il existe deux types de servomoteur : ceux qui possèdent une électronique de commande de type analogique, qui sont les plus courants et les moins chers et ceux qui sont asservis par une électronique de commande numérique, très fiables et très performants, mais bien plus onéreux que leur homologues analogiques. Vous comprendrez pourquoi notre choix s'oriente sur le premier type. 😊 De plus, leur contrôle est bien plus simple que les servomoteurs à régulation numérique qui utilisent parfois des protocoles bien particuliers.

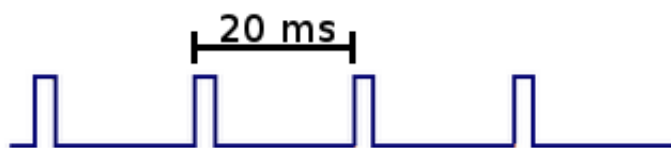
Le signal de commande

La consigne envoyée au servomoteur n'est autre qu'un signal électronique de type PWM. Il dispose cependant de deux caractéristiques indispensables pour que le servo puisse comprendre ce qu'on lui demande. À savoir : une fréquence fixe de valeur 50Hz (comme celle du réseau électrique EDF) et d'une durée d'état HAUT elle aussi fixée à certaines limites. Nous allons étudier l'affaire.

Certains sites de modélisme font état d'un nom pour ce signal : une PPM pour *Pulse Position Modulation*. J'utiliserais également ce terme de temps en temps, n'en soyez pas surpris !

La fréquence fixe

Le signal que nous allons devoir générer doit avoir une fréquence de 50 Hz. Autrement dit, le temps séparant deux fronts montants est de 20 ms. Je rappelle la formule qui donne la relation entre la fréquence (F) et le temps de la période du signal (T) : $F = \frac{1}{T}$



Malheureusement, la fonction `analogWrite()` de Arduino ne possède pas une fréquence de 50Hz, mais dix fois plus élevée, de 500Hz environ. On ne pourra donc pas utiliser cette fonction.

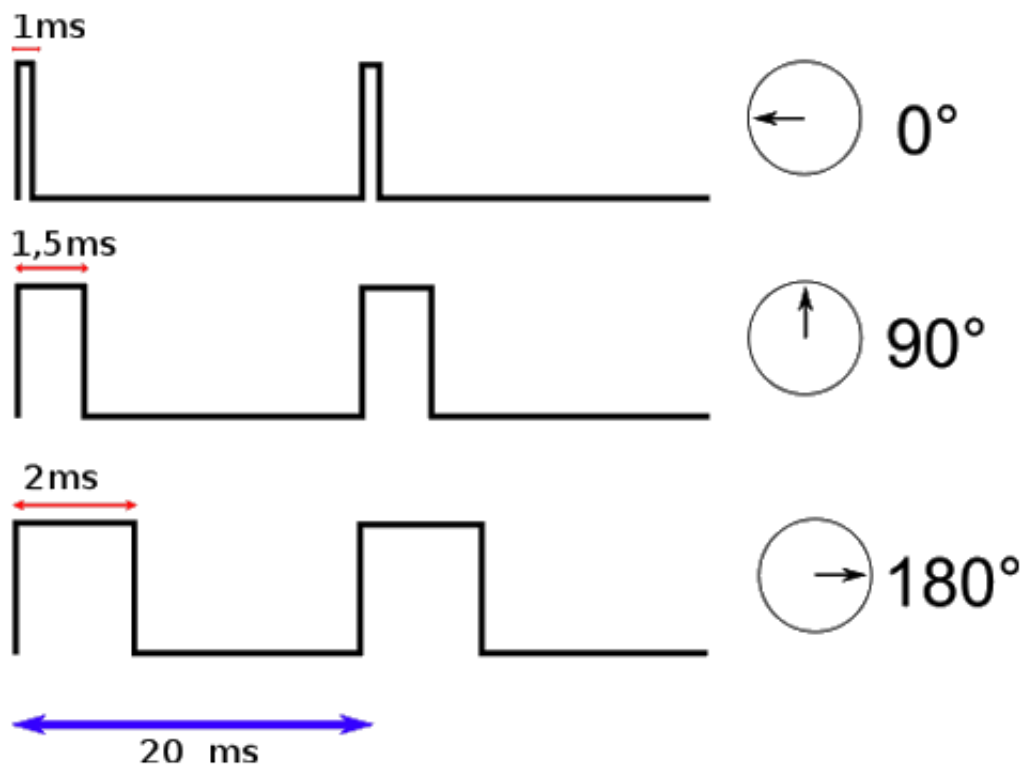
Haaaaaaa ! Mais comment on va faire !!! :(

Ola, vous affolez pas ! Il existe une alternative, ne vous pressez pas, on va voir ça dans un moment. 🌐

La durée de l'état HAUT

Pourquoi est-ce si important ? Qu'avons nous à savoir sur la durée de l'état HAUT du signal PWM ? À quoi cela sert-il, finalement ? Eh bien ces questions trouvent leurs

réponses dans ce qui va suivre, alors tendez bien l'oreille et ne perdez pas une miette de ce que je vais vous expliquer. (Eh ! Entre nous, c'est pas mal cette petite intro, non ? Elle captive votre attention tout en faisant durer le suspense. Perso j'aime bien, pas vous ? Bon, je continue. 🎧) Cette durée, chers petits zéros, est ce qui compose l'essentiel du signal. Car c'est selon elle que le servomoteur va savoir comment positionner son bras à un angle précis. Vous connaissez comment fonctionne un signal PWM, qui sert également à piloter la vitesse d'un moteur à courant continu. Eh bien, pour le servomoteur, c'est quelque peu semblable. En fait, un signal ayant une durée d'état HAUT très faible donnera un angle à 0° , le même signal avec une durée d'état HAUT plus grande donnera un angle au maximum de ce que peut admettre le servomoteur. Mais, soyons rigoureux ! Précisément, je vous parlais de valeurs limites pour cet état HAUT et ce n'est pas pour rien, car ce dernier est limité entre une valeur de 1ms au minimum et au maximum de 2ms (ce sont bien des millisecondes puisque l'on parle de durée en temps) pour les servos standards. Comme un schéma vaut mieux qu'un long discours :



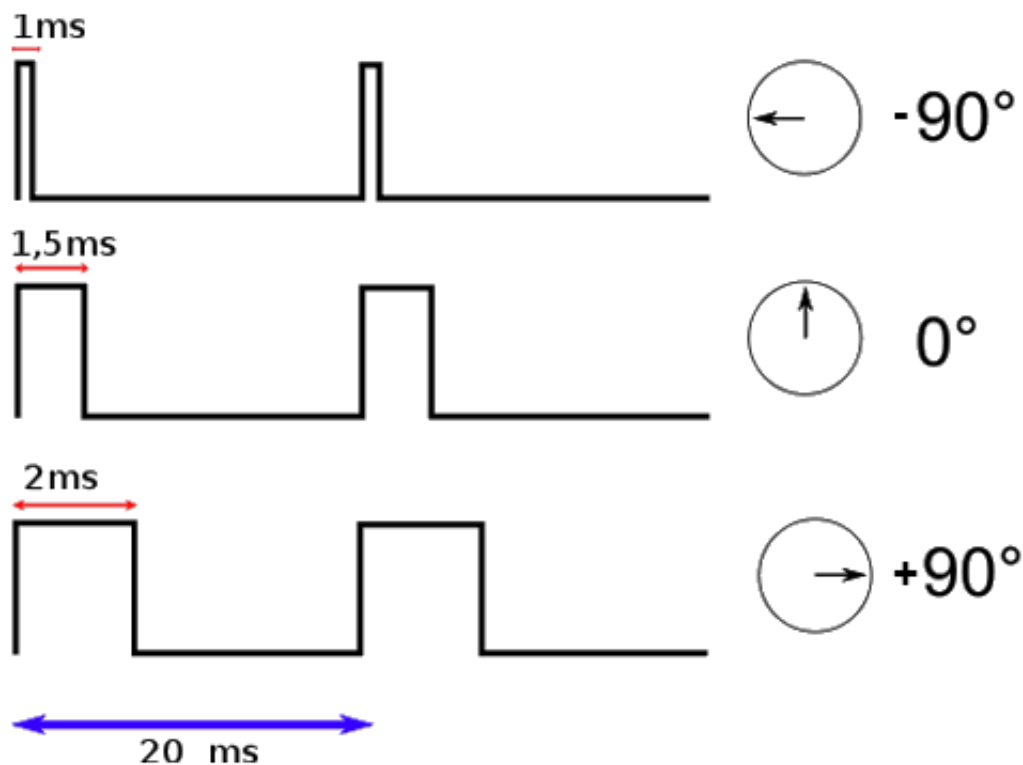
Vous aurez deviné, à travers cette illustration, que la durée de l'état HAUT fixe la position du bras du servomoteur à un angle déterminé.

Et comment je fais si je veux que mon servomoteur face un angle de 45° ? Ça marche pas ? o_O

Si, bien sûr. En fait, il va falloir faire jouer le temps de l'état HAUT. Pour un angle de 45° , il va être compris entre 1ms et $1,5\text{ms}$. À $1,25\text{ms}$ précisément. Après, c'est un rapport qui utilise une relation très simple, le calcul ne vous posera donc aucun problème. Tous les angles compris dans la limite de débattement du bras du servomoteur sont possibles et configurables grâce à ce fameux état HAUT.

Et si mon servomoteur n'a pas l'angle 0° pour origine, mais 90° , comment on fait ?

C'est pareil ! Disons que 90° est l'origine, donc on peut dire qu'il est à l'angle 0° , ce qui lui donne un débattement de -90° à $+90^\circ$:



Et dans le cas où le servo peut faire un tour complet (donc 360°), c'est aussi la même chose. En fait c'est toujours pareil, quelque soit le débattement du moteur. En revanche, c'est légèrement différent pour les servomoteurs à rotation continue. Le signal ayant un état HAUT de 1ms donnera l'ordre "vitesse maximale dans un sens", la même ayant 2ms sera l'ordre pour "vitesse maximale dans l'autre sens" et 1.5ms sera la consigne pour "moteur arrêté". Entre chaque temps (par exemple entre 1ms et 1,5ms) le moteur tournera à une vitesse proportionnelle à la durée de l'état HAUT. On peut donc commander la vitesse de rotation du servo.

Arduino et les servomoteurs

Bon, eh bien à présent, voyons un peu comment utiliser ces moteurs dont je vous vante les intérêts depuis tout à l'heure. Vous allez le voir, et ça ne vous surprendra même plus, la facilité d'utilisation est encore améliorée grâce à une bibliothèque intégrée à l'environnement Arduino. Ils nous mâchent vraiment tout le travail ces développeurs ! 🇵🇷

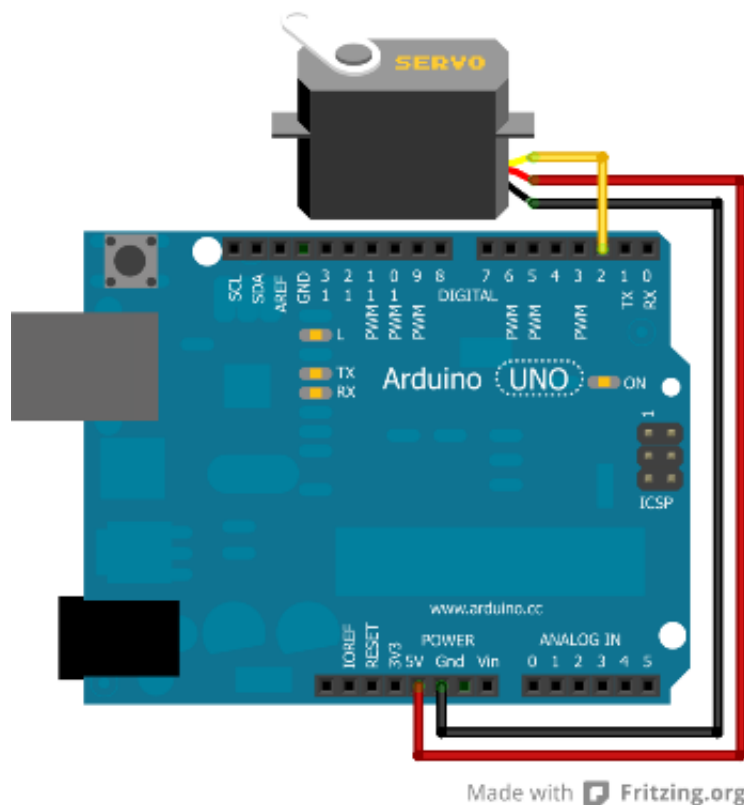
Câblage

Nous l'avons vu plus haut, la connectique d'un servomoteur se résume à trois fils : deux pour l'alimentation positive et la masse et le dernier pour le signal de commande. Rappelons qu'un servomoteur accepte généralement une plage d'alimentation comprise entre 4.5V et 6V (à 6V il aura plus de couple et sera un peu plus rapide qu'à 4.5V). Si vous n'avez besoin d'utiliser qu'un ou deux servomoteurs, vous pouvez les brancher sur la sortie 5V de la carte Arduino. Si vous voulez en utiliser plus, il serait bon d'envisager une alimentation externe car le régulateur de l'Arduino n'est pas fait pour

délivrer trop de courant, vous risqueriez de le cramer. Dans ce cas, n'oubliez pas de relier la masse de l'alimentation externe et celle de l'Arduino afin de garder un référentiel électrique commun. Le câble permettant le transit du signal de commande du servo peut-être branché sur n'importe quelle broche de l'Arduino. Sachez cependant que lorsque nous utiliserons ces derniers, les sorties 9 et 10 ne pourront plus fournir un signal PWM (elles pourront cependant être utilisées comme de simples entrées/sorties numériques). C'est une des contraintes de la bibliothèque que nous allons utiliser.

Ces dernières contraintes s'appliquent différemment sur les cartes MEGA. [Cette page](#) vous dira tout !

Voici maintenant un petit exemple de montage d'un servo sur l'Arduino :



La librairie Servo

Pour utiliser le servo avec Arduino, il va nous falloir générer le signal PPM vu précédemment. C'est à dire créer un signal d'une fréquence de 50Hz et modifier l'état haut d'une durée comprise entre 1 et 2ms. Contraignant n'est-ce pas ? Surtout si on a plusieurs servos et tout un programme à gérer derrière... C'est pourquoi l'équipe d'Arduino a été sympa en implémentant une classe très bien nommée : Servo. Tout comme l'objet Serial vous permettait de faire abstraction du protocole de la voie série, l'objet Servo va vous permettre d'utiliser les servomoteurs. Et comme elle est développée par une équipe de personnes compétentes, on peut leur faire totalement confiance pour qu'elle soit optimisée et sans bugs ! 😊 Voyons maintenant comme s'en servir !

Préparer le terrain

Tout d'abord, il nous faut inclure la librairie dans notre sketch. Pour cela vous pouvez au choix écrire vous même au début du code `#include <Servo.h>` ou alors cliquer sur

library dans la barre de menu puis sur “Servo” pour que s’écrive automatiquement et sans faute la ligne précédente. Ensuite, il vous faudra créer un objet de type Servo pour chaque servomoteur que vous allez utiliser. Nous allons ici n’en créer qu’un seul que j’appellerai “monServo” de la manière suivante : `Servo monServo;`. Nous devons lui indiquer la broche sur laquelle est connecté le fil de commande du servo en utilisant la fonction `attach()` de l’objet Servo créé. Cette fonction prend 3 arguments :

- Le numéro de la broche sur laquelle est relié le fil de signal
- La valeur basse (angle à 0°) de la durée de l’état haut du signal de PPM en microsecondes (optionnel, défaut à 544 µs)
- La valeur haute (angle à 90°, 180°, 360°, etc.) de la durée de l’état haut du signal de PPM en microsecondes (optionnel, défaut à 2400 µs)

Par exemple, si mon servo possède comme caractéristique des durées de 1ms pour 0° et 2ms pour 180° et que je l’ai branché sur la broche 2, j’obtiendrais le code suivant :

```
1 #include <Servo.h>;
2
3 Servo monServo;
4
5 void setup()
6 {
7     monServo.attach(2, 1000, 2000);
8 }
```

Utiliser le servo

Une fois ces quelques étapes terminées, notre servo est fin prêt à être mis en route. Nous allons donc lui donner une consigne d’angle à laquelle il doit s’exécuter. Pour cela, nous allons utiliser la fonction prévue à cet effet : `write()`. Tiens, c’est la même que lorsque l’on utilisait la liaison série ! Eh oui. 😊 Comme son nom l’indique, elle va écrire quelque chose au servo. Ce quelque chose est l’angle qu’il doit donner à son axe. Cette fonction prend pour argument un nombre, de type `int`, qui donne la valeur en degré de l’angle à suivre. Si par exemple je veux placer le bras du servo à mi-chemin entre 0 et 180°, j’écrirais :

```
1 monServo.write(90);
```

Pour terminer, voilà le code complet qui vous permettra de mettre l’angle du bras de votre servomoteur à 90° :

```
1 #include <Servo.h>;
2
3 Servo monServo;
4
5 void setup()
6 {
7     monServo.attach(2, 1000, 2000);
8     monServo.write(90);
9 }
10
11 void loop()
12 {
13 }
```

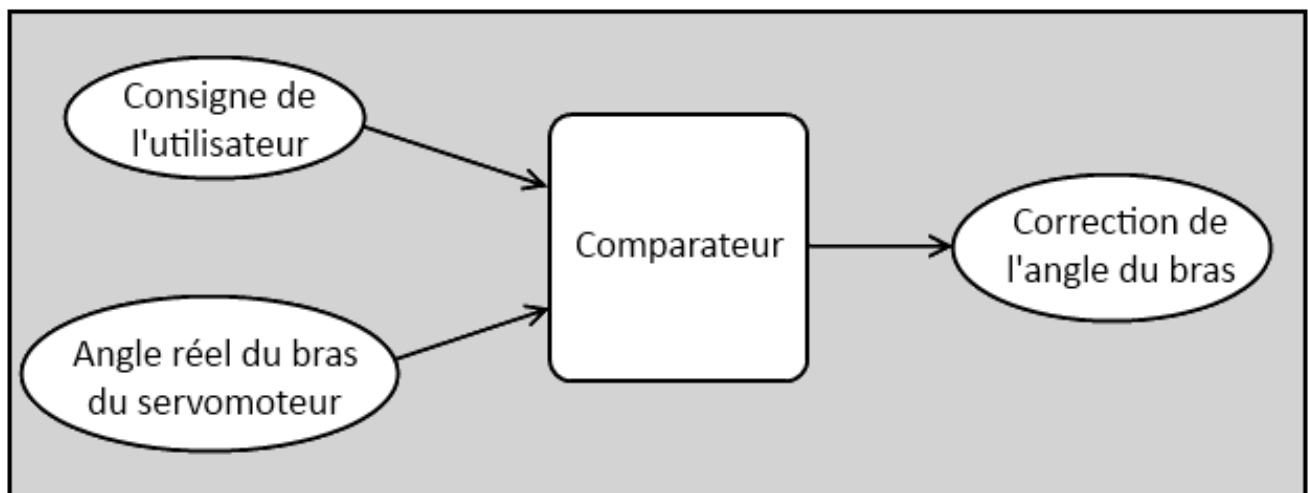
J'ai mis l'ordre de l'angle dans la fonction `setup()` mais j'aurais tout autant pu la mettre dans la `loop()`. En effet, lorsque vous utilisez `write()`, la valeur est enregistrée par Arduino et est ensuite envoyée 50 fois par seconde (rappelez vous du 50Hz du signal 😊) au servo moteur afin qu'il garde toujours la position demandée.

L'électronique d'asservissement

Je le disais donc, on va voir un peu comment se profile le fonctionnement de l'électronique interne des servomoteurs analogiques. Je précise bien *analogiques* car je rappelle qu'il y a aussi des servomoteurs numériques, beaucoup plus complexes au niveau de l'électronique.

Principe de fonctionnement

Commençons par un simple synoptique de fonctionnement. Référez-vous à la vidéo et aux explications que je vous ai données jusqu'à présent pour comprendre ce synoptique :



Principe de fonctionnement de l'électronique de commande d'un servomoteur

Rapidement : la consigne donnée par l'utilisateur (dans notre cas il va s'agir du signal envoyé par la carte Arduino), est comparée par rapport à la position réelle de l'axe du moteur. Ainsi, s'il y a une différence d'angle entre la consigne et l'angle mesuré par le capteur (le potentiomètre qui est fixé sur l'axe du servomoteur) eh bien le comparateur va commander le moteur et le faire tourner jusqu'à ce que cette différence s'annule.

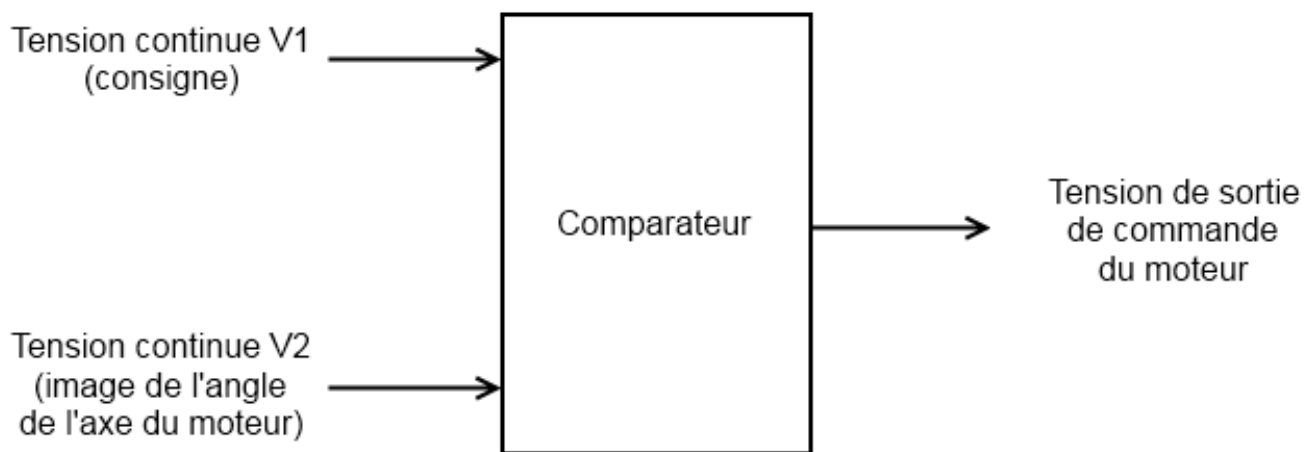
Avant d'aller plus loin, il faut savoir que les servomoteurs analogiques du commerce emploient en fait, dans leur électronique de commande, un microcontrôleur. Je ne vais donc pas vous expliquer comment ceux-là fonctionnent, mais je vais prendre le montage le plus basique qui soit. D'ailleurs, à l'issue de mes explications, vous serez capable de mettre en œuvre le montage que je vais donner et créer votre propre servomoteur avec un moteur CC anodin. 😊

Électronique à consigne manuelle

On va commencer par un montage dont la simplicité est extrême mais dont vous ne connaissez pas encore le fonctionnement d'un composant essentiel : le **comparateur**. Allez, c'est parti pour un bon petit cours d'électronique pure ! 😊 Alors, déjà, pourquoi "manuelle" ? Simplement parce que la consigne envoyée à l'électronique de commande est une tension continue et qu'elle sera réglable par un potentiomètre. En gros vous aurez simplement à faire tourner l'axe d'un potentiomètre pour régler l'angle du bras du servomoteur.

Synoptique de l'électronique interne

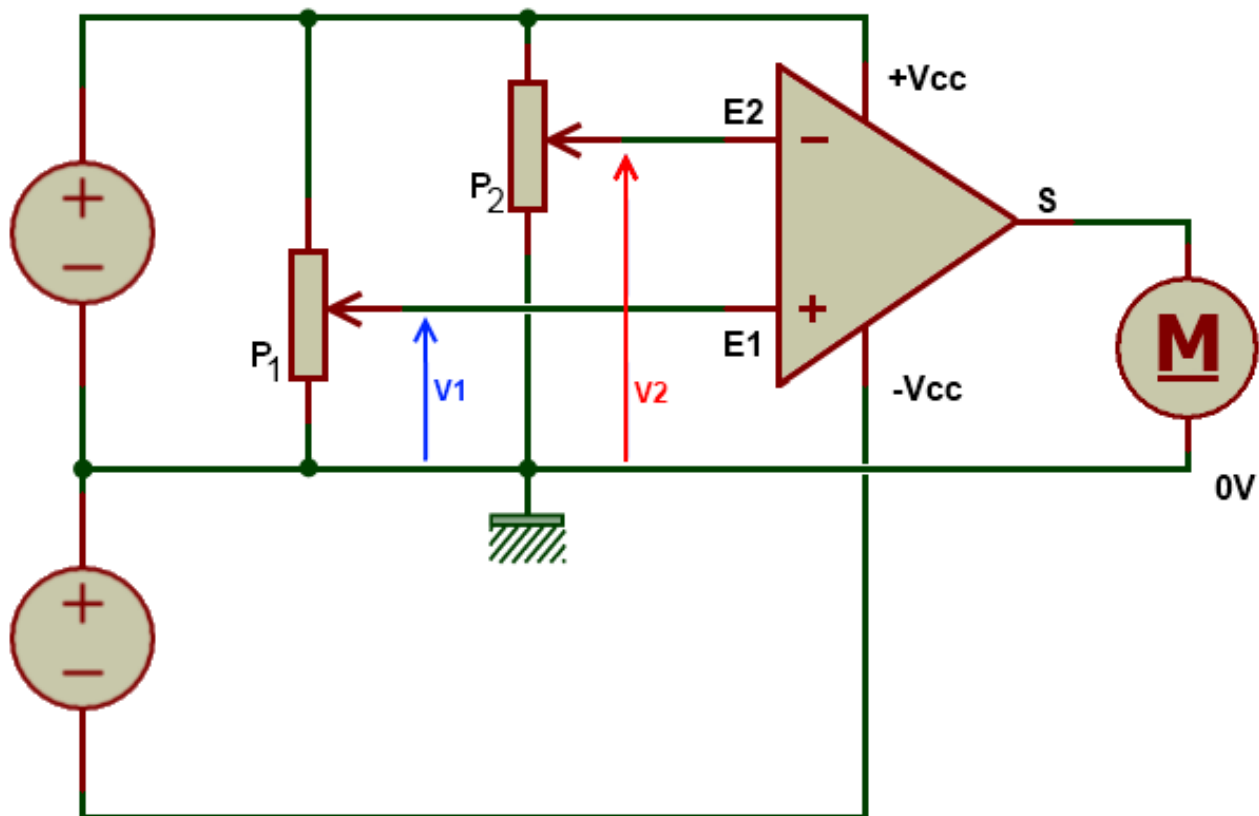
Commençons par un petit synoptique qui établit le fonctionnement de l'électronique de contrôle :



Il y a donc en entrée les deux paramètres : la consigne et l'angle réel de l'axe du moteur; Et en sortie, la tension qui va commander le moteur. On l'a vu, un moteur à courant continu doit être commandé par une tension continue, si cette tension est positive, le moteur tournera dans un sens, si elle est négative, le moteur tournera dans l'autre sens. C'est pourquoi le comparateur délivrera une tension positive ou négative selon la correction d'angle à effectuer.

Schéma de principe

À présent voici le schéma de principe qui a pour fonctionnement celui expliqué par le synoptique précédent :



De gauche à droite on a : les alimentations qui fournissent la tension positive et négative ; les potentiomètres P1 et P2 ; le comparateur (oui c'est ce gros triangle avec un plus et un moins) : enfin le moteur à courant continu.

Fonctionnement du comparateur

Un comparateur est un composant électronique de la famille des circuits intégrés car, il contient en vérité d'autres composants, essentiellement des semi-conducteurs (diodes, transistors) et des résistances. **Ce composant a toujours besoin d'une alimentation externe pour fonctionner, c'est à dire qu'on ne peut lui mettre des signaux à son entrée que s'il est alimenté.** Autrement il pourrait être endommagé (c'est pas souvent le cas, mais mieux vaut être prudent). Vous le constatez par vous-même, le comparateur est un composant qui possède deux entrées et une sortie. Et, de la manière la plus simple qui soit, en fait il n'y a rien de plus simple qui puisse exister, son fonctionnement réside sur le principe suivant :

- Si la tension (je me base par rapport au schéma) V_1 qui arrive sur l'entrée $E1$ du comparateur est supérieure à la tension V_2 qui entre sur l'entrée $E2$ du comparateur, alors la tension en sortie S du comparateur est égale à $+V_{cc}$ (l'alimentation du comparateur).
- Tandis que dans le cas opposé où la tension V_2 va être supérieure à V_1 , la sortie S du comparateur aura une tension égale à $-V_{cc}$.

En transposant mes dires sous une forme mathématique, cela donnerait ceci :

- Si $V_1 > V_2$, alors $V_s = +V_{cc}$
- Si $V_2 > V_1$, alors $V_s = -V_{cc}$

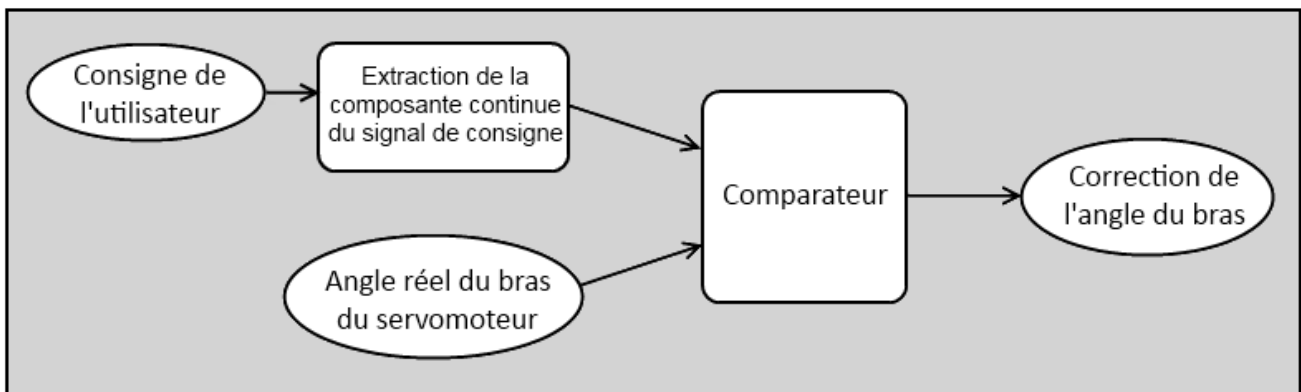
Comment s'en rappeler ? Eh bien grâce aux petits symboles "+" et "-" présents dans le triangle représentant le comparateur. La sortie du comparateur prendra $+V_{cc}$ si la

tension sur l'entrée "+" du comparateur est supérieure à celle sur l'entrée "-" et inversement. Voyez, j'avais dit que c'était ultra simple. 😊 Il y a encore quelque chose à savoir : Il est impossible que les tensions V_1 et V_2 soient égales ! Oui, car le comparateur ne peut pas fournir une tension positive ET une tension négative en sa sortie, c'est pourquoi, même si vous reliez E1 et E2 avec un fil, la tension en sortie du comparateur sera toujours OU $+V_{cc}$ OU $-V_{cc}$.

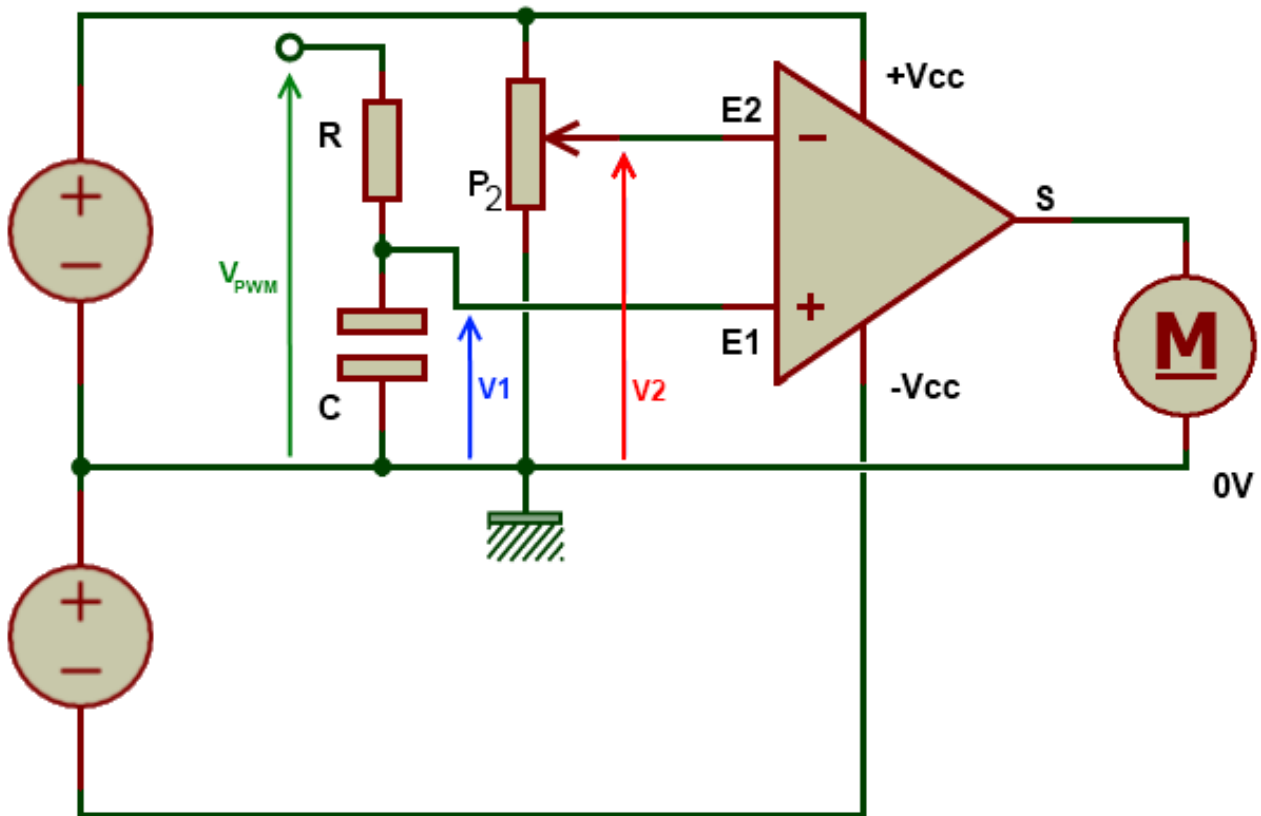
Électronique à consigne PWM

Synoptique de principe

Prenons l'exemple d'un servomoteur qui utilise une PWM, oui j'ai bien dit... euh écrit **PWM**. Je prends cet exemple fictif car comme je le disais il y a quelques instants, c'est bien souvent un microcontrôleur qui gère l'asservissement du servomoteur. Et puis, avec l'exemple que je vais vous donner, vous pourrez vous-même créer un servomoteur. 😊 En fait, on ne peut pas utiliser directement ce signal PWM avec le schéma précédent. Il va falloir que l'on fasse une extraction de la composante continue de ce signal pour obtenir une consigne dont la tension varie et non la durée de l'état HAUT du signal. Et ceci, nous l'avons déjà vu dans [un chapitre dédié à la PWM](#) justement. Le synoptique ne change guère, il y a simplement ajout de ce montage intermédiaire qui va extraire cette tension continue du signal :



Le schéma électrique ne change pas non plus de beaucoup, on retire le potentiomètre qui permettait de régler la consigne manuellement en le remplaçant par le montage qui fait l'extraction de la composante continue :



À la place du potentiomètre de commande manuelle on retrouve un couple résistance/condensateur avec R et C , qui permet d'extraire la tension continue du signal V_{PWM} qui est donc un signal de type PWM dont le rapport cyclique varie de 0 à 100%. Et là, tenez-vous bien, on en arrive au point où je voulais vous amener ! 😊 Que remarquez-vous ? Rien ? Alors je vous le dis : que se passe-t-il si on arrête d'envoyer le signal V_{PWM} ? Le moteur garde son bras au même angle ? Ou bien il reprend sa position initiale ? Réponse : il reprend sa position initiale. Eh oui, car la tension continue $V1$ n'existe plus puisqu'elle est créée à partir du signal V_{PWM} . Quand il y avait le potentiomètre, la tension $V1$ gardait la même valeur tant que vous ne tourniez pas l'axe du potentiomètre, hors là, si on enlève le signal V_{PWM} , eh bien la tension $V1$ perd sa valeur et retombe à 0V. Par conséquent, le moteur redonne à son bras sa position initiale.

Et si je veux que mon servomoteur continue de garder l'angle de la consigne qui lui a été transmise sans que je continue à lui envoyer cette consigne, est-ce possible ?

Oui, c'est tout à fait possible. En fait, cela va peut-être paraître un peu "barbare", mais c'est la seule solution envisageable avec les servomoteurs analogiques : il suffit de le positionner à l'angle voulu et de couper son alimentation. L'angle du bras du servomoteur sera alors conservé. **Mais attention, cet angle ne sera conservé que s'il n'y a pas de contrainte mécanique exercée sur le bras du servo !** C'est à dire qu'il n'y ait pas un poids accroché à l'axe du moteur, ou alors il faut qu'il soit bien inférieur à la force de maintien de la position du bras du servo lorsque celui-ci n'est plus alimenté.

Et pour l'électronique à consigne PPM alors ? o_O

Pour ce type d'électronique de commande (présent dans tous les servos du commerce),

je vous l'ai dit : il y a utilisation d'un microcontrôleur. Donc tout se fait par un programme qui scrute la position réelle du bras du moteur par rapport à la consigne PPM qu'il reçoit. Je n'ai donc rien d'intéressant à vous raconter. 😊

Un peu d'exercice !

Bon allez, il est temps de faire un peu d'entraînement ! Je vous laisse découvrir le sujet...

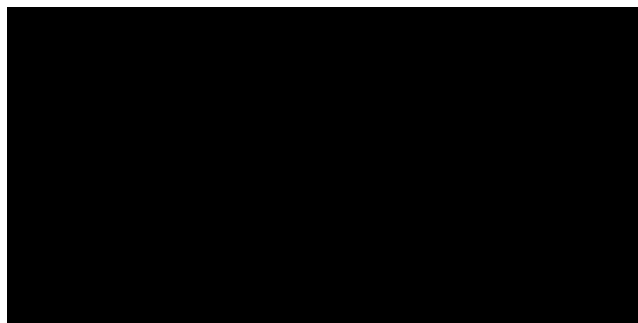
Consigne

Nous allons utiliser trois éléments dans cet exercice :

- un servomoteur (évidemment)
- un potentiomètre (valeur de votre choix)
- la liaison série

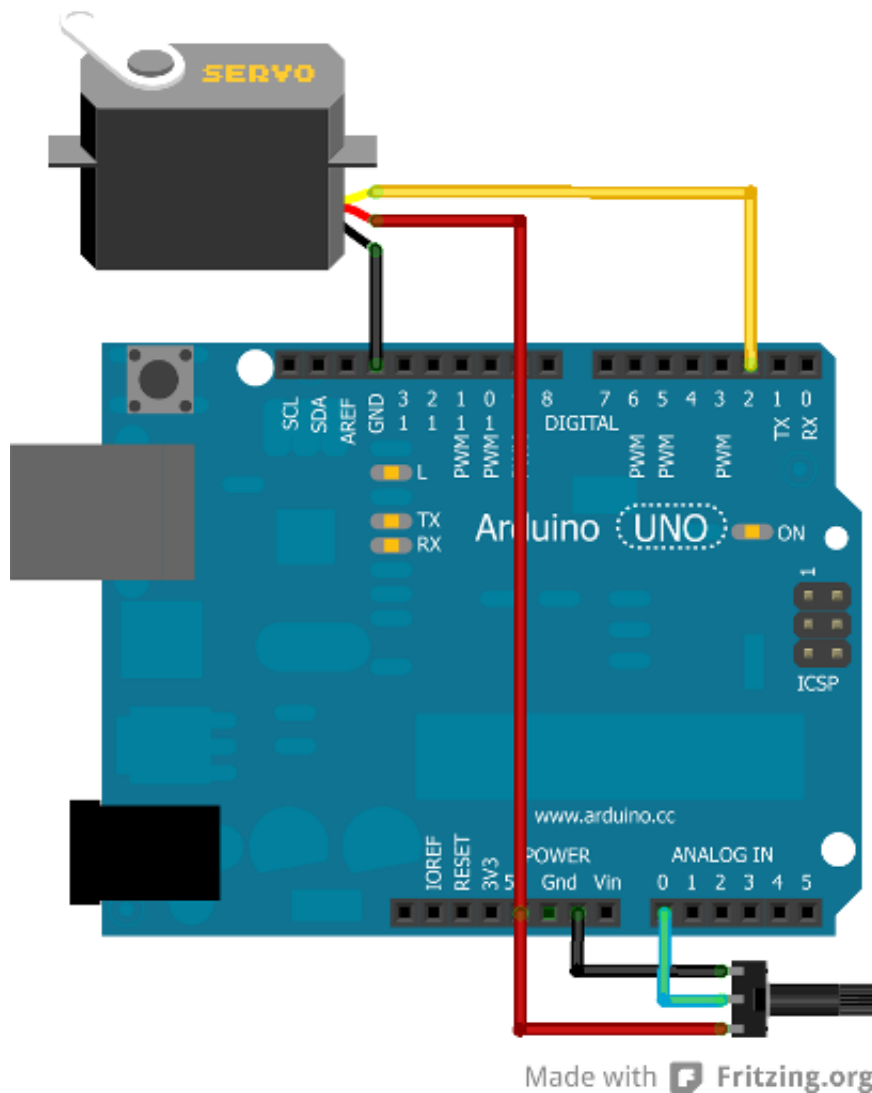
Objectif

Le servo doit "suivre" le potentiomètre. C'est à dire que lorsque vous faites tourner l'axe du potentiomètre, le bras du servomoteur doit tourner à son tour et dans le même sens. Pour ce qui est de l'utilisation de la liaison série, je veux simplement que l'on ait un retour de la valeur donnée par le potentiomètre pour faire une supervision. Je ne vous en dis pas plus, vous savez déjà tout faire. Bon courage et à plus tard ! 😊



Correction

J'espère que vous avez réussi ! Tout d'abord le schéma, même si je sais que vous avez été capable de faire les branchements par vous-même. C'est toujours bon de l'avoir sous les yeux. 😊



Pour ma part, j'ai branché le servo sur la broche numérique 2 et le potentiomètre sur la broche analogique 0. J'ai donc le code suivant pour préparer l'ensemble :

```
1  #include <Servo.h> //On oublie pas d'ajouter la bibliothèque !
2
3  //notre potentiomètre
4  const int potar = 0;
5
6  //création d'un nouveau servomoteur
7  Servo monServo;
8
9  void setup()
10 {
11     //on déclare l'entrée du servo connectée sur la broche 2
12     monServo.attach(2);
13     //on oublie pas de démarrer la liaison série ;-)
14     Serial.begin(9600);
15 }
```

Voilà qui est fait pour les préparatifs, il n'y a plus qu'à travailler un tout petit peu pour faire la logique du code. Commençons par la lecture analogique que nous allons renvoyer sur le servo ensuite. Le potentiomètre délivre une tension variable de 0 à 5V selon sa position. La carte Arduino, elle, lit une valeur comprise entre 0 et 1023. Ce nombre est stocké au format `int`. Il faut ensuite que l'on donne à la fonction qui permet d'envoyer la consigne au servo une valeur comprise entre 0 et 180°. On va donc utiliser

une fonction dédiée à cela. Cette fonction permet de faire le rapport entre deux gammes de valeurs ayant chacune des extremums différents. Il s'agit de la fonction `map()` (nous en avons parlé dans le chapitre sur les lectures analogiques) :

```
1 map(value, fromLow, fromHigh, toLow, toHigh)
```

Avec pour correspondance :

- **value** : valeur à convertir pour la changer de gamme
- **fromLow** : valeur minimale de la gamme à convertir
- **fromHigh** : valeur maximale de la gamme à convertir
- **toLow** : valeur minimale de la gamme vers laquelle est convertie la valeur initiale
- **toHigh** : valeur maximale de la gamme vers laquelle est convertie la valeur initiale

Nous utiliserons cette fonction de la manière suivante :

```
1 map(valeur_potentiometre, 0, 1023, 0, 180) ;
```

On aurait pu faire un simple produit en croix, non ?

Tout à fait. Mais les programmeurs sont de véritables fainéants et aiment utiliser des outils déjà prêt. 😊 Cela dit, ils les ont créés. Et pour créer de nouveaux outils il est plus facile de prendre des outils déjà existants. Mais si vous voulez on peut recréer la fonction `map()` par nous-mêmes :

```
1 int conversion(int mesure)
2 {
3     return mesure*180/1023;
4 }
```

Fonction `loop()`

Dans la fonction `loop()` on a donc la récupération et l'envoi de la consigne au servomoteur :

```
1 void loop()
2 {
3     //on lit la valeur du potentiomètre
4     int val = analogRead(potar);
5     //mise à l'échelle de la valeur lue vers la plage [0;180]
6     int angle = map(val, 0, 1023, 0, 180);
7     //on met à jour l'angle sur le servo
8     monServo.write(angle);
9 }
```

Avez vous remarqué que ces trois lignes de code auraient pu être réduites en une seule ? 😊 Comme ceci :

```
1 monServo.write(map(analogRead(potar), 0, 1023, 0, 180));
```

Ou bien la version utilisant le produit en croix :

```
1 void loop()
```

```

2 {
3     //on lit la valeur du potentiomètre
4     int val = analogRead(potar);
5     //on convertit la valeur lue en angle compris dans l'intervalle [0;180]
6     int angle = val / 5.7;
7     //5,7 provient de la division de 1023/180 pour la mise à l'échelle de la valeur
8
9     //on met à jour l'angle sur le servo
10    monServo.write(angle);
11 }

```

Et à nouveau une condensation de ces trois lignes en une :

```

1 monServo.write(analogRead(potar)/5.7);

```

Mais comme il nous faut renvoyer la valeur convertie vers l'ordinateur, il est mieux de stocker cette valeur dans une variable. Autrement dit, préférez garder le code à trois lignes.

Et la liaison série

Pour renvoyer la valeur, rien de bien sorcier :

```

1 Serial.println(angle);

```

Code final

Au final, on se retrouve avec un code tel que celui-ci :

```

1  #include //On oublie pas d'ajouter la bibliothèque !
2
3  const int potar = 0; //notre potentiomètre
4  Servo monServo;
5
6  void setup()
7  {
8      //on déclare le servo sur la broche 2 (éventuellement régler les bornes)
9      monServo.attach(2);
10     //on oublie pas de démarrer la voie série
11     Serial.begin(9600);
12 }
13
14 void loop()
15 {
16     //on lit la valeur du potentiomètre
17     int val = analogRead(potar);
18     //on convertit la valeur lue en angle compris dans l'intervalle [0;180]
19     int angle = val / 5.7;
20     //on met à jour l'angle sur le servo
21     monServo.write(angle);
22     //on renvoie l'angle par la voie série pour superviser
23     Serial.println(angle);
24     //un petit temps de pause
25     delay(100);
26 }

```

Je vous laisse mixer avec les différents codes que l'on vous a donné pour que vous

fassiez celui qui vous convient le mieux (avec la fonction `map()`), ou bien celui qui est tout condensé, etc.). Dorénavant, vous allez pouvoir vous amuser avec les servomoteurs ! 😊

Tester un servomoteur “non-standard”

C'est déjà la fin ? o_O

Eh oui, je n'ai plus grand chose à vous dire, car ce n'est pas très compliqué puisqu'il suffit d'utiliser un outil déjà tout prêt qui est la bibliothèque `Servo`. Je vais cependant vous montrer deux autres fonctions bien utiles.

`writeMicroSeconds()`

En premier, la fonction `writeMicroSeconds()`. Cette fonction permet de définir un temps à l'état HAUT du signal PPM autre que celui compris entre 1 et 2 ms. Elle est très pratique pour tester un servo dont vous ne connaissez pas les caractéristiques (servo 0 à 90° ou autre). De plus, il arrive que certains constructeurs ne se soucient pas trop des standards [1ms-2ms] et dépassent un peu ces valeurs. De par ce fait, si vous utilisez un servo avec les valeurs originales vous n'obtiendrez pas le comportement escompté. En utilisant cette fonction, vous pourrez ainsi tester le servo petit à petit en envoyant différentes valeurs une à une (par la voie série par exemple).

Une valeur incorrecte se repère assez facilement. Si vous voyez votre servo “trembler” aux alentours des 0° ou 180° ou bien encore s'il fait des allers-retours étranges sans que vous n'ayez changé la consigne alors c'est que la valeur utilisée est probablement fausse.

`read()`

Une deuxième fonction pouvant être utile est la fonction `read()`. Tout l'intérêt de cette fonction est perdu si elle est utilisée pour le code que l'on a vu dans l'exercice précédent. En revanche, elle a très bien sa place dans un système où le servomoteur est géré automatiquement par le programme de la carte Arduino et où l'utilisateur ne peut y accéder.

Programme de test

En préparant ce chapitre, j'ai pu commencer à jouer avec un servomoteur issu de mes fonds de tiroirs. N'ayant bien entendu aucune documentation sur place ou sur internet, j'ai commencé à jouer avec en assumant qu'il utiliserait des valeurs “standards”, donc entre 1000 et 2000µs pour l'état haut de la PPM. J'ai ainsi pu constater que mon servo fonctionnait, mais on était loin de parcourir les 180° attendu. J'ai donc fait un petit code utilisant une des fonctions précédentes pour tester le moteur en mode “pas à pas” et ainsi trouver les vrais timings de ces bornes. Pour cela, j'ai utilisé la liaison série. Elle m'a servi pour envoyer une commande simple ('a' pour augmenter la consigne, 'd' pour la diminuer). Ainsi, en recherchant à tâtons et en observant le comportement du moteur, j'ai pu déterminer qu'il était borné entre 560 et 2130 µs. Pas super proche des 1 et 2ms

attendues ! 😊 Comme je suis sympa (😁), je vous donne le code que j'ai réalisé pour le tester. Les symptômes à observer sont : aucune réaction du servo (pour ma part en dessous de 560 il ne se passe plus rien) ou au contraire, du mouvement sans changement de la consigne (de mon côté, si on augmente au dessus de 2130 le servo va continuer à tourner sans s'arrêter).

```
1 #include
2
3 int temps = 1500; //censé être à mi-chemin entre 1000 et 2000, un bon point de départ
4
5 Servo monServo;
6
7 void setup()
8 {
9     Serial.begin(115200);
10    Serial.println("&quot;Hello World&quot;");
11
12    monServo.attach(2);
13    //on démarre à une valeur censé être la moitié de
14    //l'excursion totale de l'angle réalisé par le servomoteur
15    monServo.writeMicroseconds(temps);
16 }
17
18 void loop()
19 {
20    //des données sur la liaison série ? (lorsque l'on appuie sur 'a' ou 'd')
21    if(Serial.available())
22    {
23        char commande = Serial.read(); //on lit
24
25        //on modifie la consigne si c'est un caractère qui nous intéresse
26        if(commande == 'a')
27            temps += 10; //ajout de 10µs au temps HAUT
28        else if(commande == 'd')
29            temps -= 10; //retrait de 10µs au temps HAUT
30
31        //on modifie la consigne du servo
32        monServo.writeMicroseconds(temps);
33
34        //et on fait un retour sur la console pour savoir où on est rendu
35        Serial.println(temps);
36    }
37 }
```

Ce programme est très simple d'utilisation et vous pouvez d'ailleurs le modifier comme bon vous semble pour qu'il corresponde à ce que vous voulez faire avec. Il suffit en fait de brancher la carte Arduino à un ordinateur et ouvrir un terminal série (par exemple le moniteur intégré dans l'environnement Arduino). Ensuite, appuyez sur 'a' ou 'd' pour faire augmenter ou diminuer le temps de l'état HAUT du signal PPM. Vous pourrez ainsi avoir un retour des temps extrêmes qu'utilise votre servomoteur.

On en termine avec les servomoteurs. Vous avez sans doute plein de nouvelles idées avec lesquelles vous emploieriez les servomoteurs qui vous permettront de faire beaucoup de choses très utiles, voir inutiles mais indispensables. 😊

[Arduino 603] A petits pas, le moteur pas-à-pas

Pour en terminer avec les différents types de moteurs qui existent, nous allons parler d'un moteur un peu particulier (encore plus que le servomoteur !) et qui est cependant très utilisé dans le domaine de la robotique et tout ce qui touche à la précision d'un mouvement. Comme à l'habitude, nous allons d'abord voir le fonctionnement de ces moteurs, pour ensuite apprendre à les utiliser.

Ce moteur utilise des éléments que nous avons vus dans des chapitres précédents (sur les moteurs à courant continu). Si vous ne vous rappelez pas du L298 je vous conseille de retourner prendre quelques informations à ce sujet 😊

Les différents moteurs pas-à-pas et leur fonctionnement

Les moteurs pas-à-pas... encore un nouveau type de moteur. Une question vous taraude sûrement l'esprit :

Pourquoi il existe tant de moteur différent !?

Et bien je vous répondrais par une autre question : Pourquoi il existe autant de langages de programmation différent !? La réponse est pourtant simple : Car ils ont tous leurs avantages et leurs inconvénients. Par exemple, un servomoteur pourra facilement maintenir la position de son axe, tandis que le moteur à courant continu sera plus facile à faire tourner à différentes vitesses. Eh bien le but du moteur pas-à-pas (que j'abrégerais moteur pàp) est un peu une réunion de ces deux avantages. Vous pourrez le faire tourner à des vitesses variables et la position parcourue sera aussi facile à déterminer. En contrepartie ces moteurs ne peuvent pas tourner à des vitesses hallucinantes et sont plus délicat à mettre en œuvre que les moteurs CC par exemple (mais rien d'insurmontable je vous rassure). En parlant de précision, savez-vous dans quel objet du quotidien on retrouve beaucoup de moteur pàp ? Dans l'imprimante (éventuellement scanner aussi) qui traîne sur votre bureau ! En effet, l'aspect "précision" du moteur est utilisé dans cette situation sans avoir besoin d'aller vraiment vite. Vous pourrez donc en trouver un pour faire avancer les feuilles et un autre pour déplacer le chariot avec les cartouches d'encre (et encore un autre pour déplacer le capteur du scanner). Donc si vous avez une vieille imprimante destinée à la poubelle, vous savez ce qu'il vous reste à faire 😊 ! Les moteurs que vous pourrez trouver posséderont 4, 5 voire 6 fils. Le premier (4 fils) est appelé **moteur bipolaire**, les deux autres sont des moteurs **unipolaires** ou à **reluctance variable**. Tout cela doit-être encore un peu confus. Voyons donc plus clairement comment cela marche !

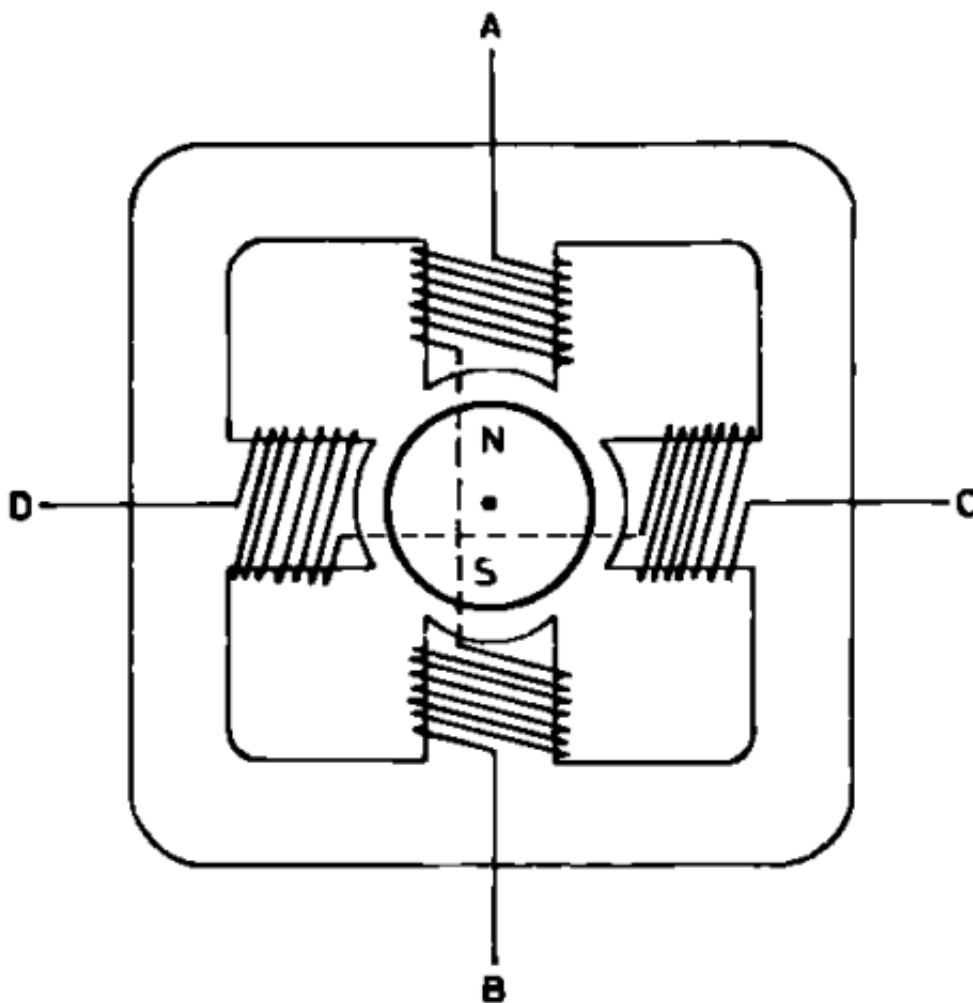
Fonctionnement des moteurs

Comme précédemment avancé, ce moteur possède une certaine complexité pour être mis en œuvre. Et ce, plus que les précédents. Vous souvenez-vous du moteur CC (j'espère bien ! 😊). Il était composé d'un ensemble d'aimants sur l'extérieur (le stator) et d'une partie bobinée où le champ magnétique était créée dynamiquement avec un ensemble collecteur/balais qui transmettait l'électricité aux bobines au centre (rotor).

Dans le cas du moteur pàp, c'est sur le rotor (au centre) que l'on retrouve l'aimant permanent, et les bobines sont sur le stator (autour du rotor). Comme pour les moteurs à courant continu, le but du jeu, en quelque sorte, est de "faire tourner un champ magnétique" (à prendre avec des pincettes) pour faire tourner l'aimant fixé au rotor. Il existe cependant différents types de moteurs pàp dont le placement des bobinages diffère les uns des autres et la façon de les alimenter n'est pas non plus identique (d'où une complexité supplémentaire lorsque l'on veut changer le type de moteur pàp à utiliser...). Nous allons maintenant les étudier l'un après l'autre en commençant par celui qui semble avoir le fonctionnement le plus simple à assimiler.

Moteur pàp bipolaire à aimants permanents

Ce moteur possède quatre fils d'alimentation pour piloter des bobines par paire. Comme un schéma vaut mieux qu'un long discours, voici comment il est constitué :



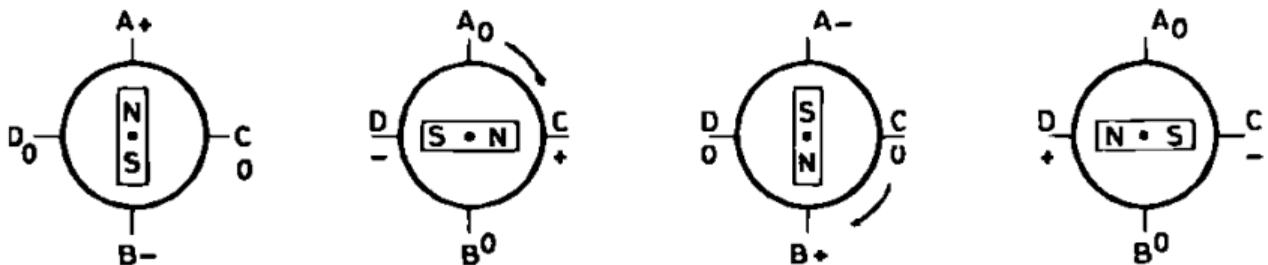
Vous l'aurez compris, les bobines sont reliées deux à deux en série et sont donc pilotées ensemble. Il n'y a donc finalement que deux enroulements à commander puisque deux bobines montées en série n'en font plus qu'une. Leur placement de part et d'autre de l'aimant permanent du rotor permet de piloter ce dernier. Voyons comment.

- Lorsqu'il n'y a aucun courant traversant les bobines, le rotor (où l'axe de sortie est lié) est libre de tourner, rien ne cherche à le retenir dans sa course
- Maintenant, si nous décidons de faire passer du courant entre les points C et D

pour alimenter la bobine de gauche et celle de droite. Un courant va s'établir et deux champs électromagnétiques vont apparaître de part et d'autre du rotor. Que va-t-il alors se passer ? L'aimant du rotor va tourner sur lui même pour se placer de façon à ce que son pôle Nord soit en face du pôle Sud du champ magnétique créé dans la première bobine et que son pôle Sud soit en face du pôle Nord créé dans la deuxième bobine

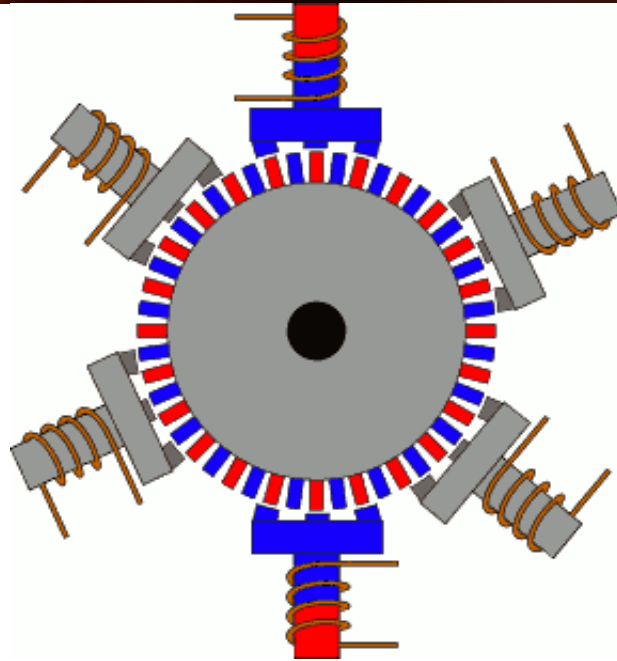
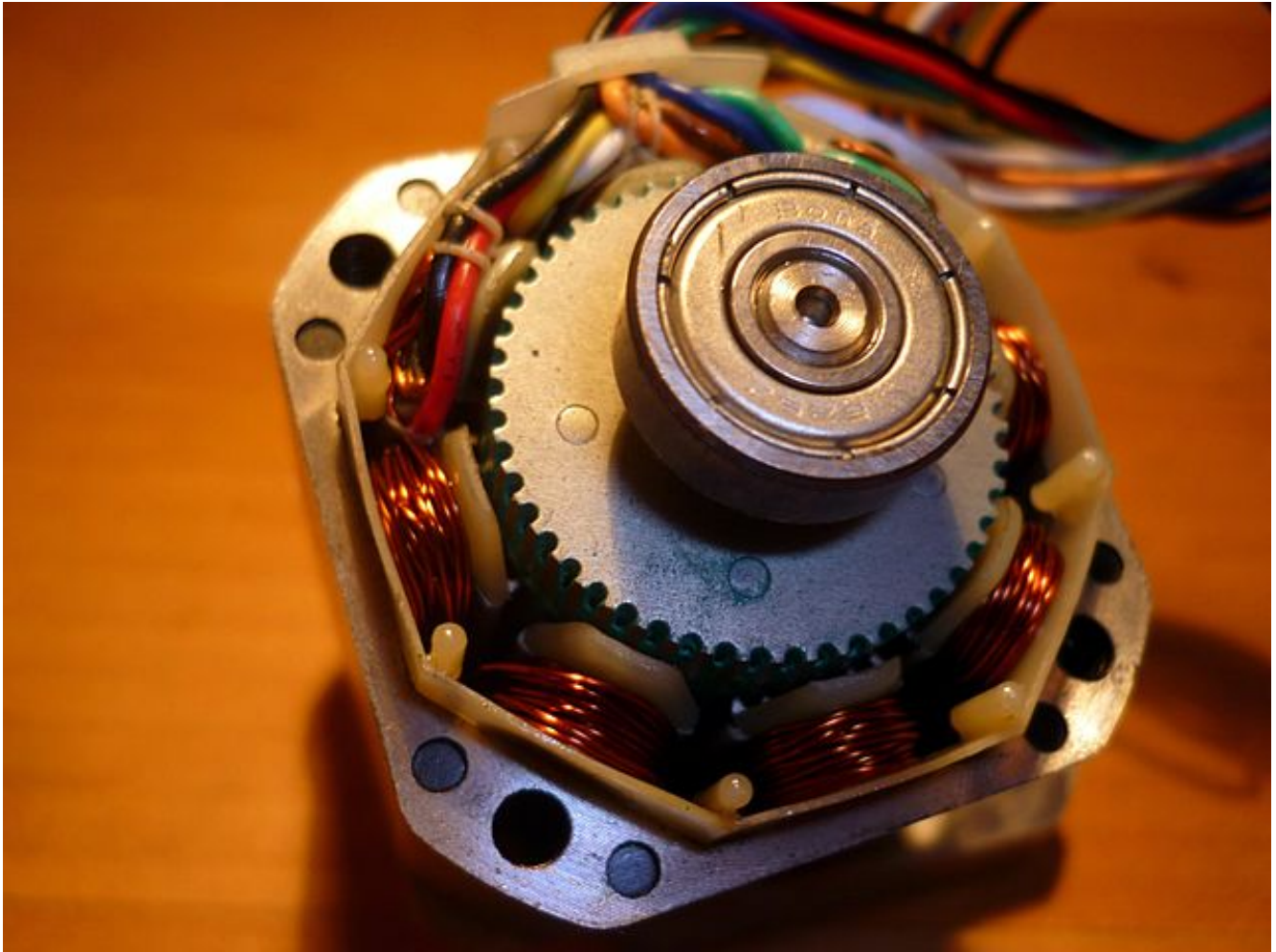
- Si ensuite on alimente non plus les bobines entre C et D mais plutôt celles entre A et B, le rotor va alors tourner pour s'aligner à nouveau vers les pôles qui l'intéressent (Nord/Sud, Sud/Nord)
- Et c'est reparti, on va alors alimenter de nouveau les bobines entre D et C, donc avec un courant de signe opposé à la fois où l'on les a alimentés entre C et D (par exemple C était relié au "+" de l'alimentation tout à l'heure et là on le fait passer au "-", idem pour D que l'on fait passer du "-" au "+") et le moteur va encore faire un quart de tour
- On peut continuer ainsi de suite pour faire tourner le moteur en faisant attention de ne pas se tromper dans les phases d'alimentation

A chaque phase on va donc faire tourner le moteur d'un quart de tour :

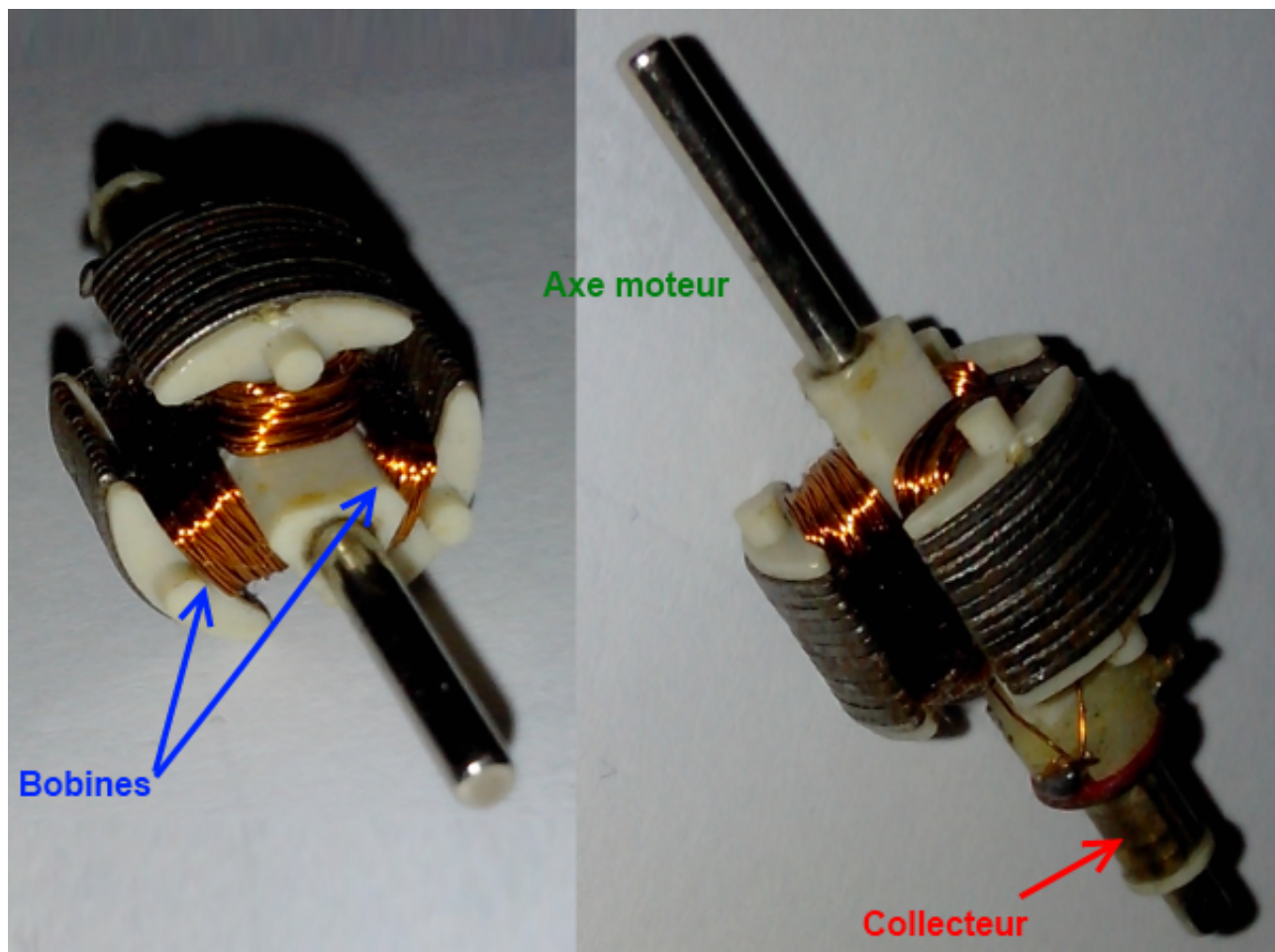


Ce quart de rotation s'appelle un **pas**. Et comme il faut plusieurs pas pour faire tourner le moteur sur 360°, on l'a donc appelé ainsi, le moteur *pas-à-pas*.

Dans le cas illustré ci-dessus, on dit que le moteur fait 4 pas par tour. Il existe bien des moteurs qui font ce nombre de pas, mais il en existe qui ont un nombre de pas plus conséquent (24, 48, etc.). Leur constitution mécanique est différente, ce qui leur confère ce pouvoir, bien que le fonctionnement reste identique, puisque l'on cherche toujours à attirer un aimant grâce à des champs magnétiques créés par des bobines parcourues par un courant. Pour avoir plus de pas, on multiplie les aimants au centre. Sur l'image ci-dessous, on peut bien voir les bobines (en cuivre à l'extérieur) et tout les aimants au centre (les petites dents). Il existe aussi deux autres modes de fonctionnement que nous verrons dans la partie suivante : **le pilotage avec couple maximal** et **le pilotage par demi-pas**.

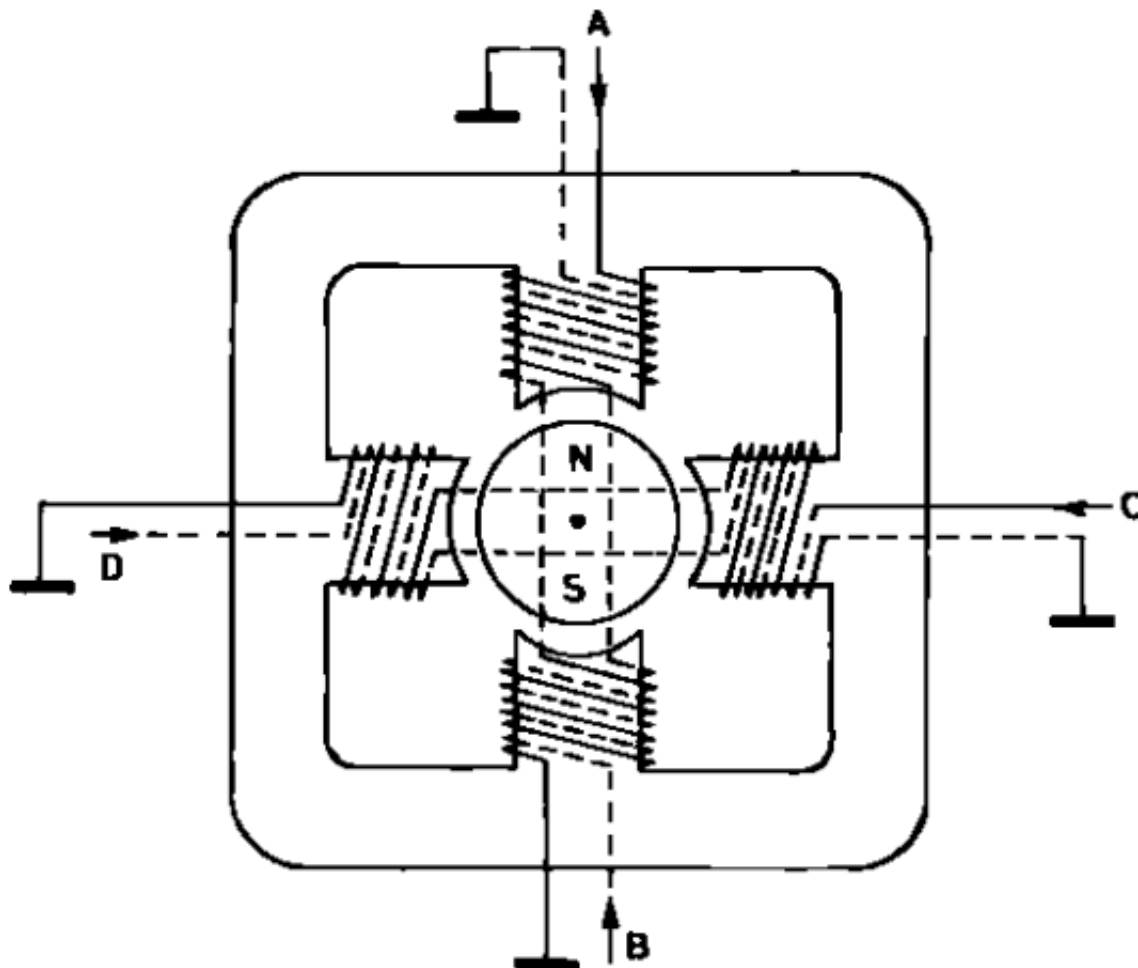


(Pour rappel, voici la vue d'un moteur à courant continu :



Le moteur unipolaire

Le moment est enfin venu de vous révéler la véritable signification des noms du moteur vu précédemment et de celui-ci même... non il ne faut pas lire ça sur un ton tragique. 🤖 Le moteur bipolaire est nommé ainsi car il présente la faculté d'être commandé en inversant simplement la polarité de ces bobinages. Quant au moteur unipolaire, pas besoin de faire cette inversion, chaque bobinage est commandé séparément et ne requiert qu'une alimentation présente ou absente selon que l'on veuille ou non créer un champ magnétique en son sein. La commande sera donc plus simple à mettre en place qu'avec le moteur bipolaire. Cependant, le nombre de bobine étant plus important, la quantité de cuivre également et le prix s'en ressent ! En effet, il s'agit bien de 4 bobine bien distinctes, alors que le moteur bipolaire à aimant permanent en possède finalement quatre moitiés de bobines, donc deux bobine complètes.

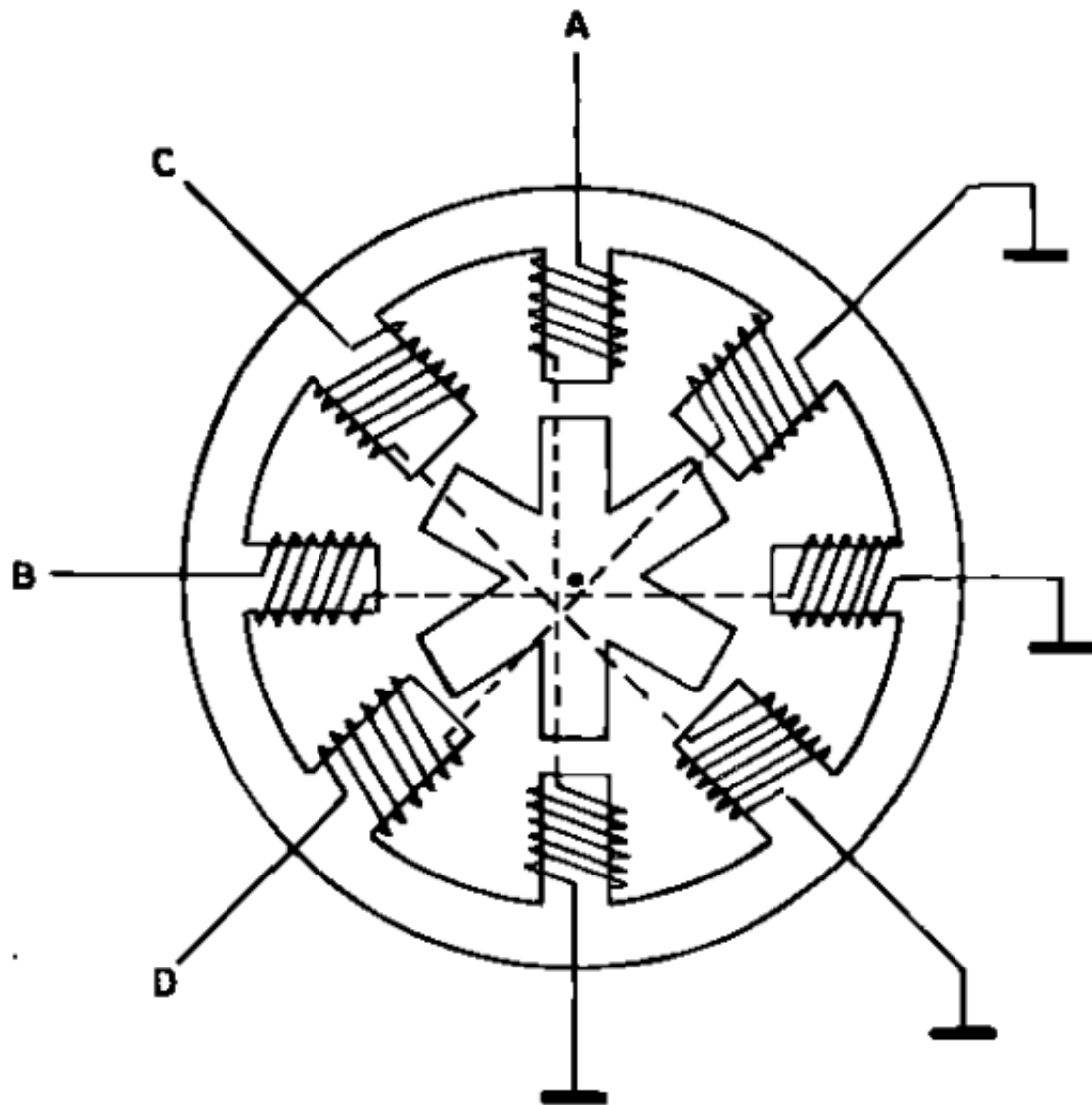


On retrouve nos quatre fils A, B, C et D ainsi qu'un fil de masse commun (bon ben imaginez-le puisqu'il n'est pas dessiné comme tel sur le schéma). Soit 5 fils (v'là pas qu'on sait compter maintenant ! 🤖). Le fonctionnement est rigoureusement identique que le précédente moteur. On cherche à créer un champ magnétique "rotatif" pour faire passer l'aimant alternativement devant chacune des bobines. On va donc alimenter la bobine A, puis la C, puis la B, puis la D selon le schéma ci-dessus. Et voila, le moteur aura fait tout un tour assez simplement. Il suffit d'avoir quatre transistors (un par enroulement) sans avoir besoin de les disposer en H et de les piloter deux à deux. Ici il suffit de les alimenter un par un, chacun leur tour. Facile, non ? 😊

Le moteur à réluctance variable

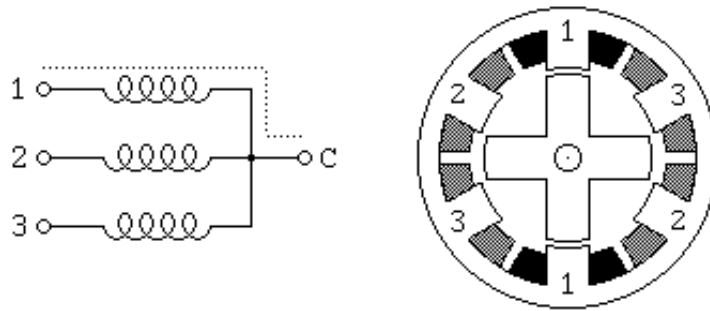
gné 🤖 ? c'est quoi ce nom barbare ?

Ce moteur est un peu plus compliqué, mais c'est aussi le dernier que nous verrons et le plus fascinant ! Contrairement aux deux précédent, ce moteur ne possède pas d'aimants permanent, ni même d'aimant tous court ! Non, en son centre on trouve un simple morceau de **fer doux**. Ce dernier a la particularité de très bien conduire les champs magnétiques. Du coup, si un champ magnétique le traverse, il voudra absolument s'aligner dans son sens. C'est cette propriété qui est exploitée. Commençons par voir tout de suite le schéma de ce moteur :



Comme vous pouvez le voir, il possède 4 enroulements (formant 4 paires) et le morceau de fer doux au milieu à une forme d'étoile à 6 branches. Et ce n'est pas un hasard ! Le ratio de 6 pour 8 a une raison très précise car cela introduit un léger décalage (15°) entre une branche et une bobine. En effet, si on a 8 bobines (4 paires) on a un décalage entre chaque bobine de : $\frac{360^\circ}{8} = 45^\circ$. Donc tous les 45° le long du cylindre qu'est le moteur, on trouve un bobinage. En revanche il n'y a que 60° entre chaque extrémité de l'étoile du rotor : $\frac{360^\circ}{6} = 60^\circ$. Mais pourquoi exactement ? Eh bien c'est simple avec un peu d'imagination (et quelques dessins). Si l'on commence par alimenter le premier enroulement, le A, le rotor va s'aligner avec. Maintenant que se passera-t-il si l'on alimente B ? Le rotor, qui était alors positionné avec une de ses branches bien en face de A, va bouger pour s'aligner correctement vers B. Ensuite si l'on alimente C il va se passer de même, le rotor va encore tourner de 15° pour s'aligner. etc etc... Si l'on effectue cette opération 24 fois, on fera un tour complet car : $24 \times 15^\circ = 360^\circ$.

Vous remarquerez que dans cet exemple le rotor tourne dans le sens horaire alors que l'alimentation des bobines se fera dans le sens anti-horaire.



Ces moteurs ont certains avantages. Parmi ces derniers, il n'y a pas besoin de polariser les bobines (peu importe le sens du champ magnétique, l'entrefer n'étant pas polarisé essaiera de s'aligner sans chipoter). Le fait que l'on utilise un simple entrefer en fer doux le rend aussi moins cher qu'un modèle avec des aimants permanents. Vous savez maintenant tout sur les trois types de moteurs pas à pas que l'on peut trouver, place maintenant à la pratique !

De manière générale, n'essayez pas d'ouvrir vos moteur pas-à-pas pour regarder comment c'est fait et espérer les remonter après. Le simple démontage à tendance à faire diminuer la qualité des aimants permanents à l'intérieur et donc votre moteur ne sera plus aussi bon après remontage.

Se servir du moteur

Continuons à parler de notre super moteur. Si vous avez suivi ce que j'ai dit plus tôt, j'ai expliqué qu'il y avait des bobines qui génèrent un champ magnétique. Lorsqu'elles sont alimentées, ces bobines ont besoin de courant pour pouvoir générer un champ magnétique suffisant. Vous ne serez donc pas surpris si je vous dis qu'il faudra utiliser un composant pour faire passer la puissance dans ces dernières. Et là, comme les choses sont bien faites, nous allons retrouver le pont en H et le L298. 😊

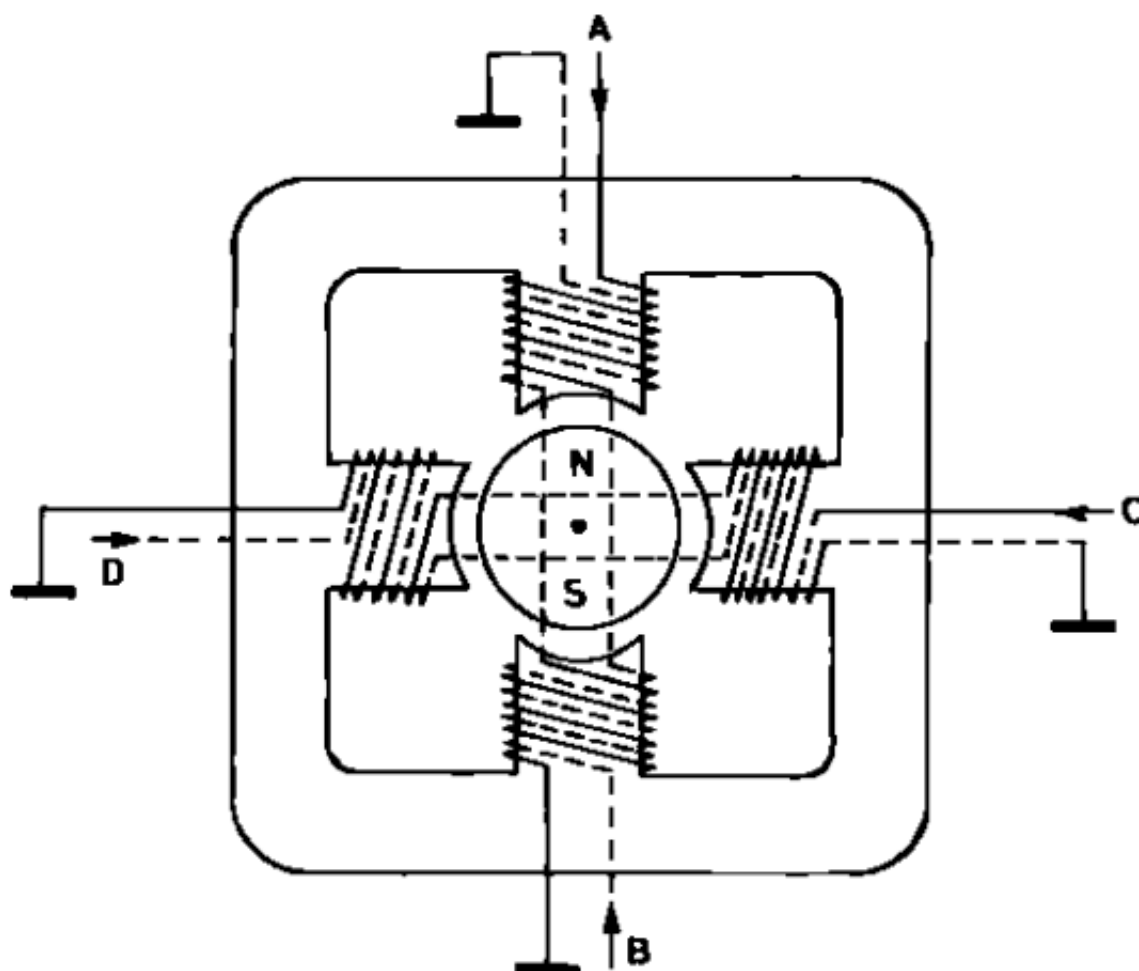
Afin de limiter la redondance d'informations, je vais me contenter de vous expliquer le pilotage du moteur unipolaire et bipolaire. Si vous comprenez correctement ces derniers vous n'aurez aucun problème avec le moteur restant 😊

Les schémas qui vont suivre ont pour source d'énergie une pile +9V. Cependant il est déconseillé de les faire avec car la consommation des moteurs est assez importante et la pile risque de se fatiguer très vite. Utilisez plutôt une source d'alimentation prévue à cet effet (une alimentation de laboratoire).

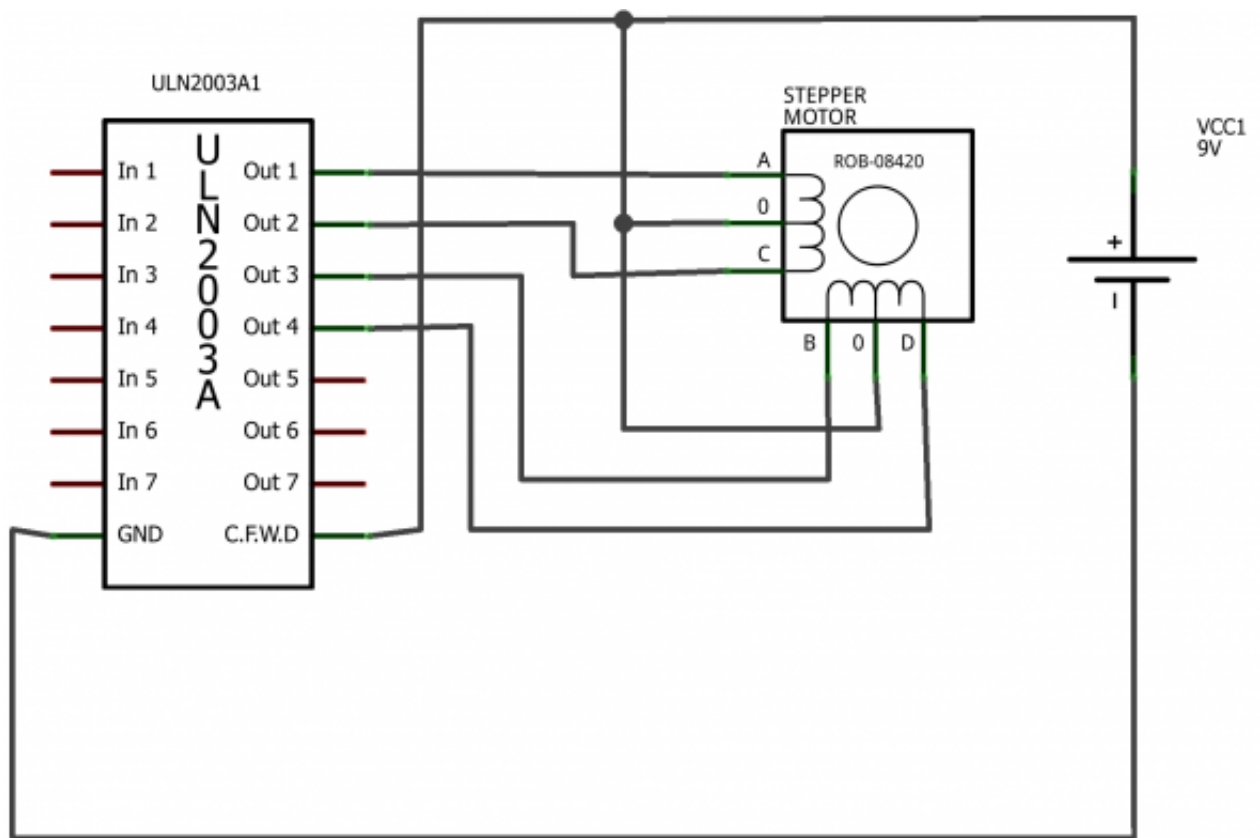
Le moteur unipolaire

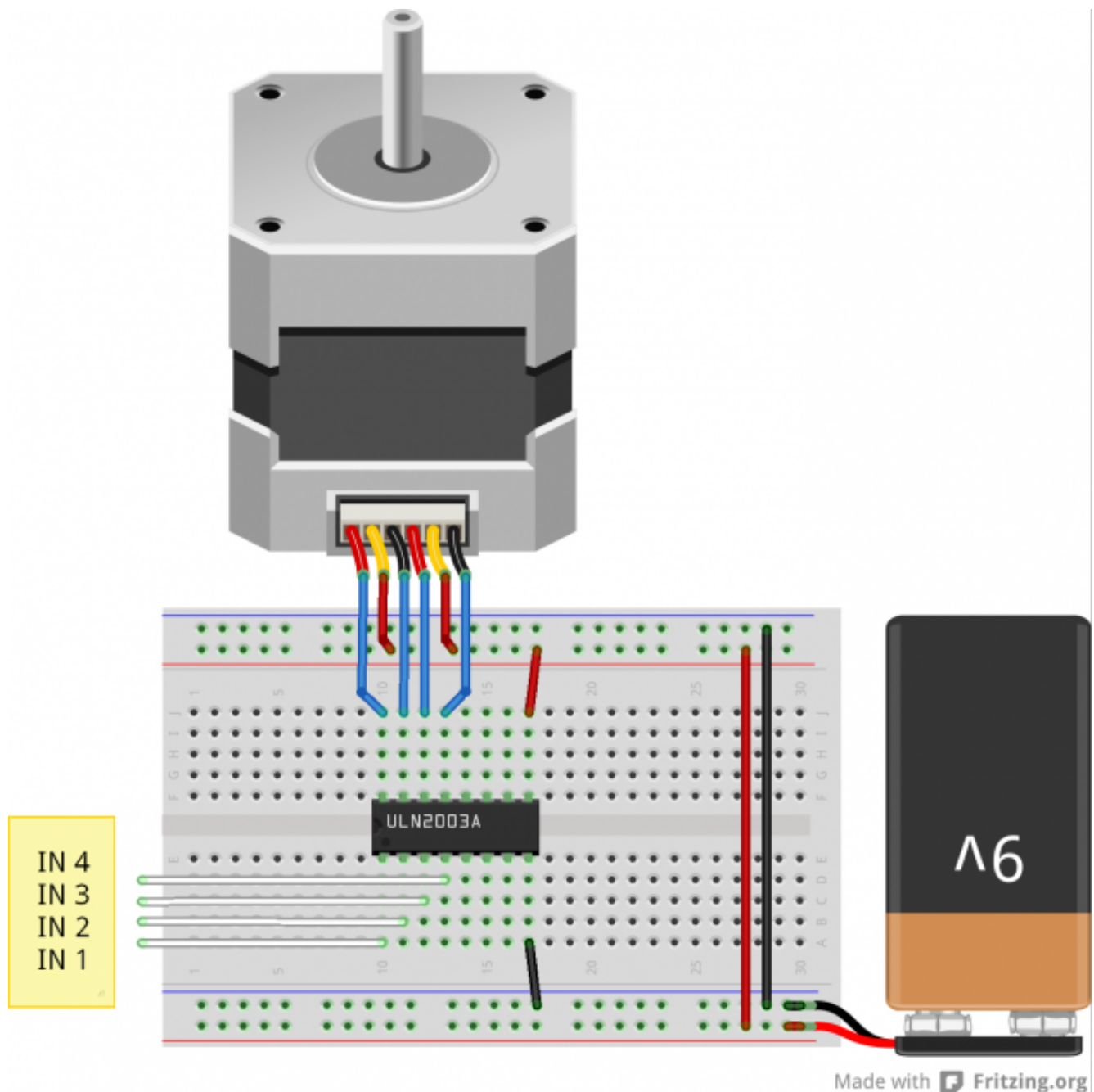
Connecter un moteur unipolaire

Pour rappel, voici la structure du moteur unipolaire :



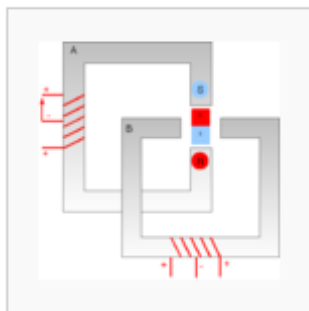
Si vous avez bien lu la partie précédente, vous devez avoir retenu que ce moteur est assez simple à piloter. En effet, il suffit d'alimenter une à une les bobines pour que le moteur tourne. Et c'est tout ! Il nous faut juste utiliser le bon composant pour alimenter les bobines et en avant ! A priori, le bon composant serait un bon transistor qui pourrait supporter 50V et 500mA pour débiter. A cela il faudrait ensuite rajouter une diode de roue libre pour ne pas l'abîmer lors des phases de roue libre (tout cela multiplié par 4 puisqu'on a 4 bobines). Plutôt que de s'embêter à câbler tout ça, je vous propose l'intervention d'un nouveau composant : le ULN2003A. Ce dernier regroupe les transistors pour faire passer la puissance et aussi la diode. Il est très simple à câbler puisqu'il faut simplement amener l'alimentation et les broches de commandes. Chacune de ces dernières possède respectivement une sortie où la tension sera celle de l'alimentation. Voici une illustration de ce câblage :



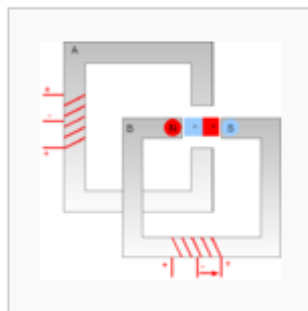


Utiliser un moteur unipolaire

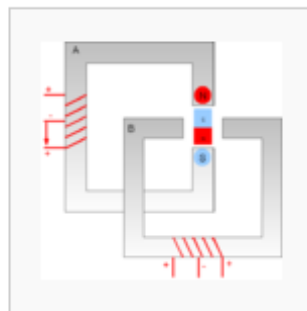
Je l'ai maintenant dit et répété plusieurs fois, pour ce moteur il suffit de piloter chaque bobine une à une, chacune leur tour. Je vais donc vous résumer tout cela de manière plus schématique et on sera bon pour ce moteur. 😊 A la fin, si vous avez bien compris vous devriez être capable de le faire tourner tout doucement en plaçant alternativement les fils In1 à 4 (abrégé In1..4) au 5V ou à la masse.



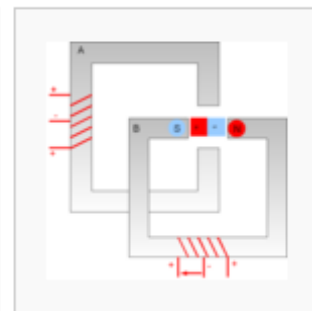
Pas N°1



Pas N°2



Pas N°3



Pas N°4

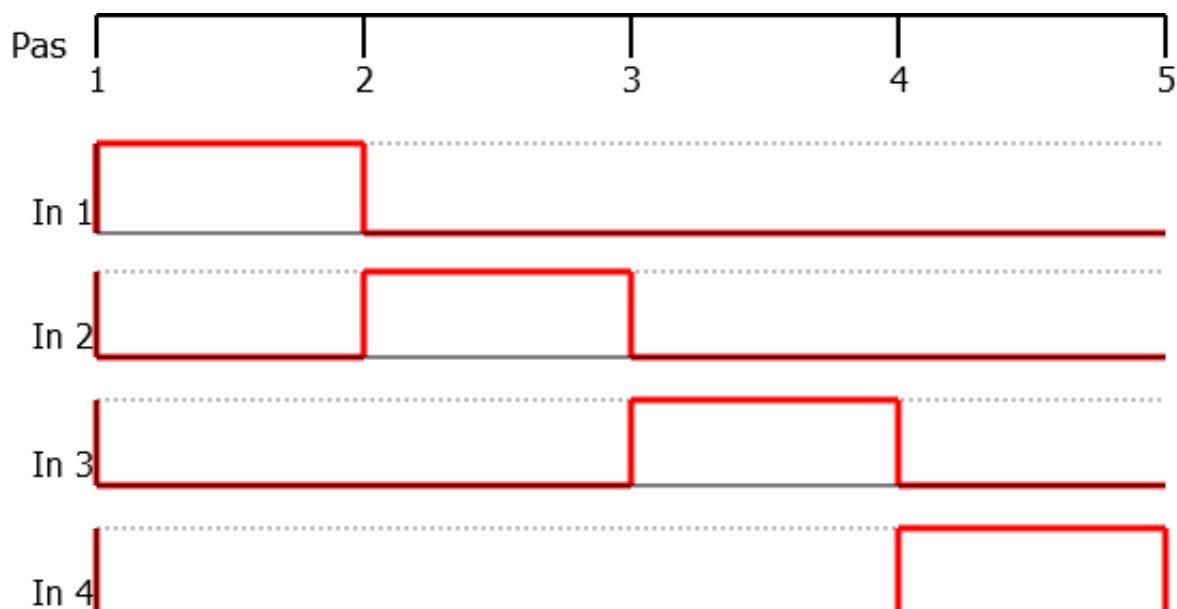
Etape In 1 In 2 In 3 In 4

Pas n°1 HIGH LOW LOW LOW

Pas n°2 LOW HIGH LOW LOW

Pas n°3 LOW LOW HIGH LOW

Pas n°4 LOW LOW LOW HIGH

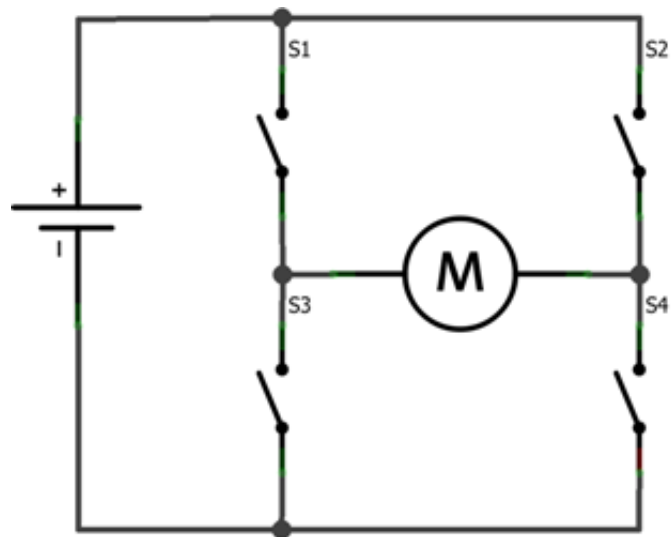


Si vous placez les fils de commande à la masse ou au 5V convenablement, votre moteur devrait tourner 😊 Vous n'avez plus qu'à créer le programme qui va bien pour piloter tout ce bazar... ben vous vous attendiez à quoi ? Une solution peut-être ? Non. 😞 Bon bon, d'accord... vous vous souvenez de vos premiers pas avec le chenillard ? Tout est dit. 😊 (et si vous ne voulez pas vous fatiguer, attendez la suite du tuto 😊)

Le moteur bipolaire

Le câbler, la théorie

Nous avons vu dans le chapitre précédent que le moteur bipolaire a besoin, à un moment, d'une inversion de courant si l'on veut pouvoir utiliser les bobines à bon escient. Vous vous souvenez probablement que nous avons vu précédemment un composant capable de faire cela : le pont en H. L'idée va donc être de câbler correctement les bobines pour pouvoir faire passer le courant dans un sens, ou dans l'autre. Je vous rappelle la structure du pont en H :

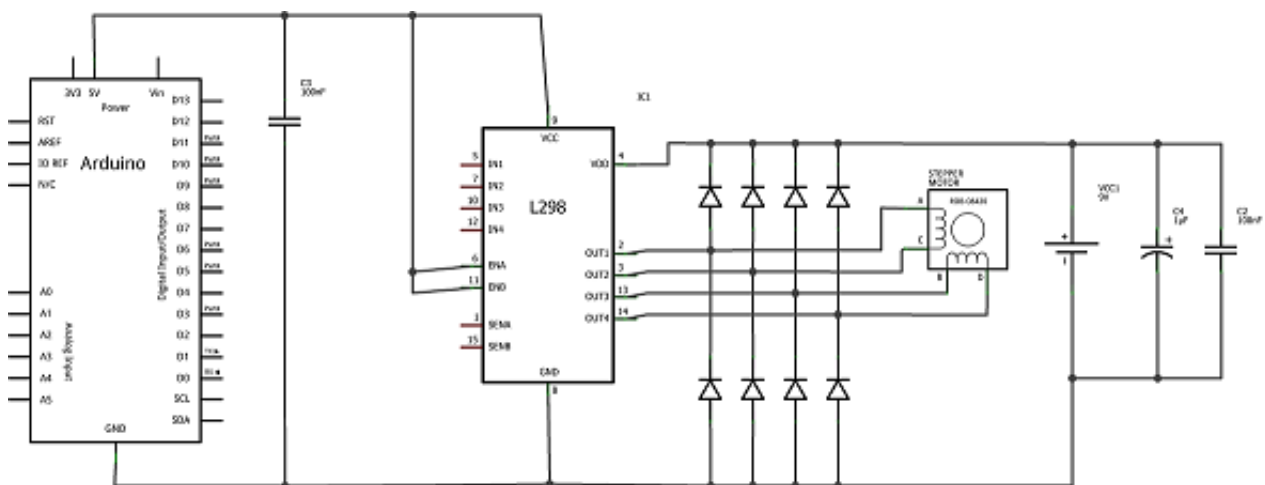


Made with  Fritzing.org

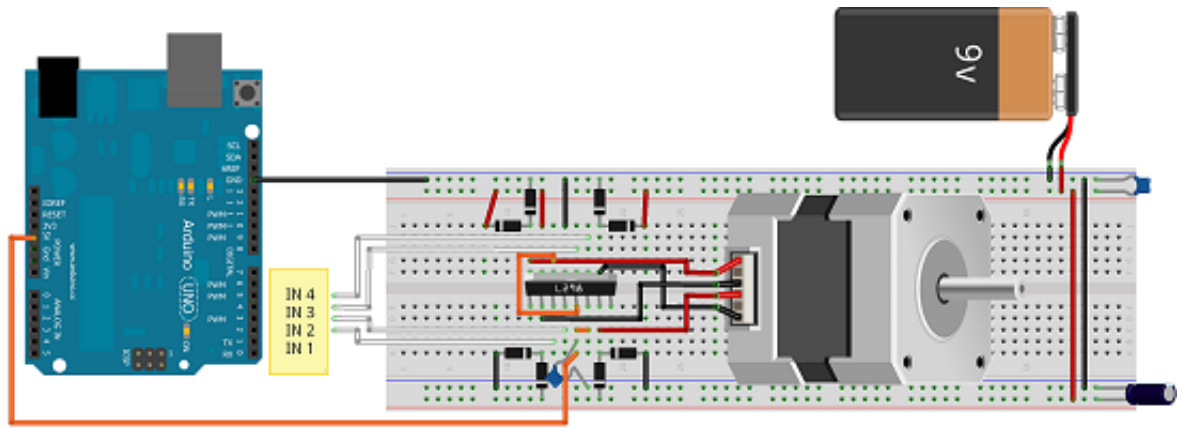
Problème, un pont en H ne peut piloter qu'une "paire de bobine" (celle qui sont dans le même axe et relié entre elles), hors nous en avons deux à piloter. Heureusement le L298 est un composant très bien pensé et ne possède non pas un mais bien deux ponts en H ! Du coup, nous pourrons en utiliser un par couple de bobine 😊. Plutôt sympa non ?

La pratique avec le L298

Dans le chapitre précédent, nous avons justement vu un composant qui possède deux ponts en H : le L298. Le reste du travail semble presque trop simple ! Pour la partie "sortie/puissance", vous devrez relier les sorties Out1 (broche 2) et Out2 (broche 3) à la première bobine et ensuite Out3 et Out4 (13 et 14) à la seconde. N'oubliez pas les diodes de roue libre (8 au total et qui supporte la puissance) ! Enfin, il ne reste qu'à connecter les entrées In1..4 à 4 entrée/sortie numérique (pas besoin de PWM). Pour l'instant, pas besoin de les relier à l'Arduino, contentez-vous de les mettre à la masse, nous allons faire un test ensemble. Comme nous voulons activer les deux ponts, mettez les deux entrées "enable" au +5V. Nous verrons dans la prochaine partie comment l'utiliser avec Arduino 😊. Voici un petit schéma récapitulatif :

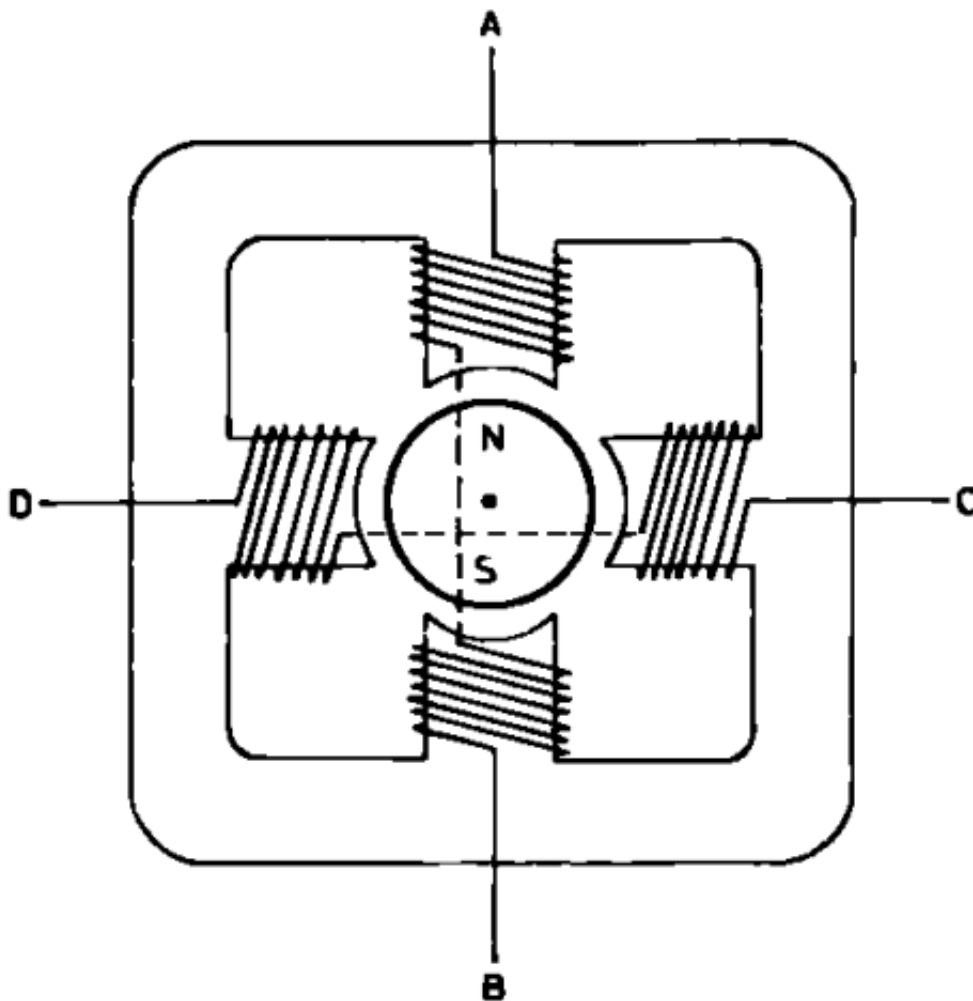


Le schéma de montage avec quelques condensateurs de filtrage qui viennent aider l'alimentation en cas de forte demande de courant et des condensateurs qui filtrent les parasites engendrés par les bobines du moteurs.



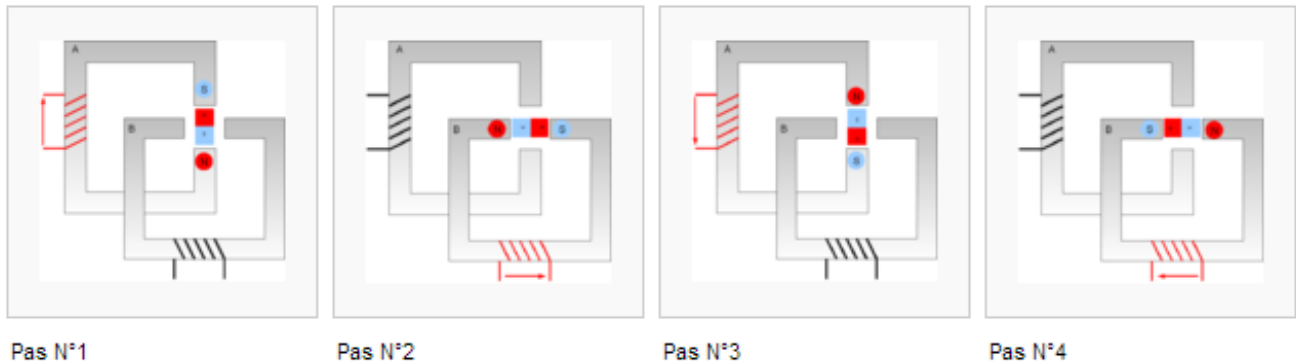
Piloter le moteur bipolaire

Une fois que le moteur est branché, nous allons pouvoir le faire tourner. Comme son nom l'indique, il s'appelle pas à pas et donc nous allons le faire pivoter étape par étape et non de manière continue comme le moteur à courant continu. Cependant, en répétant les étapes de rotation rapidement et successivement, le moteur donnera l'impression de tourner sans s'arrêter entre chaque étape. Il existe trois méthodes distinctes pour piloter les moteurs bipolaires. Nous allons maintenant les voir une par une. Dans les cas qui vont suivre, on va considérer un moteur de 4 pas par tour (ce qui est ridiculement faible). Voici ce à quoi il va ressembler (comment sont placés ses bobines) :



Rotation par pas complet

Ce mode de fonctionnement est le plus simple, c'est pourquoi nous allons commencer par lui. Grâce à ce dernier, vous allez pouvoir faire des rotations "pas par pas". Pour cela, nous allons alternativement alimenter les bobines de droites et de gauche et inverser le sens du courant pour pouvoir faire une rotation complète du champ magnétique. Voici l'illustration Wikipédia très bien faite à ce sujet :



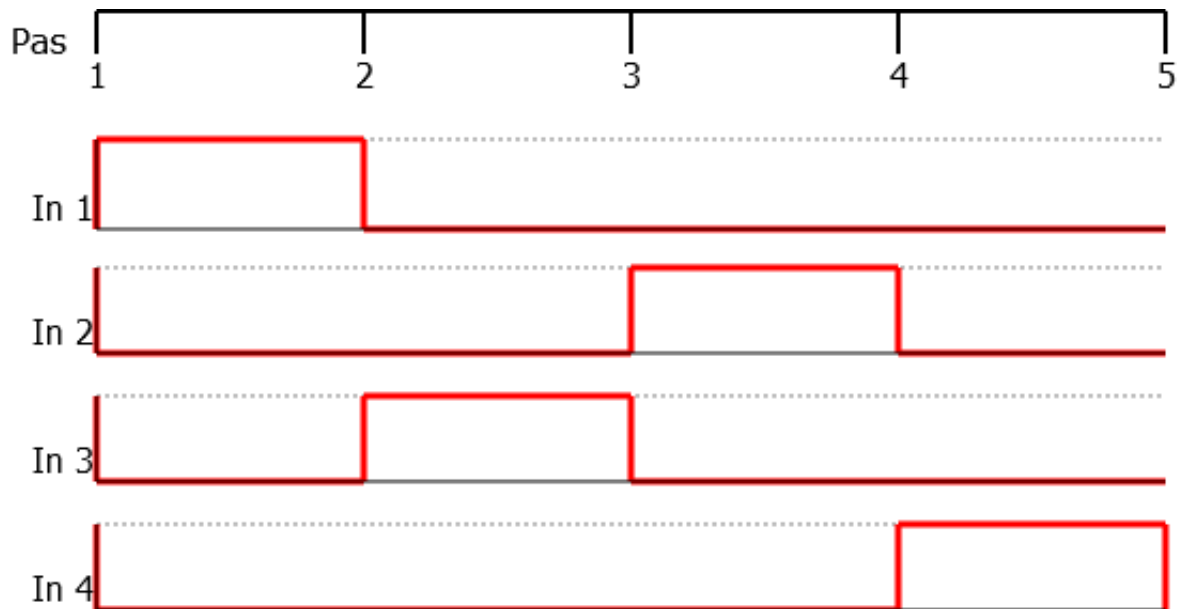
En rouge la bobine alimentée ainsi que le sens du courant symbolisé par une flèche et en noir la bobine qui n'est pas alimentée.

En considérant que la bobine A est connectée aux entrées IN1 et IN2 et que la bobine B est connectée aux commandes IN3 et IN4, on peut donc écrire la séquence de commande suivante :

Etape	In 1	In 2	In 3	In 4
Pas n°1	HIGH	LOW	-	-
Pas n°2	-	-	HIGH	LOW
Pas n°3	LOW	HIGH	-	-
Pas n°4	-	-	LOW	HIGH

(un état '-' signifie "non nécessaire", placez-le à 0V pour que la bobine soit bien inactive).

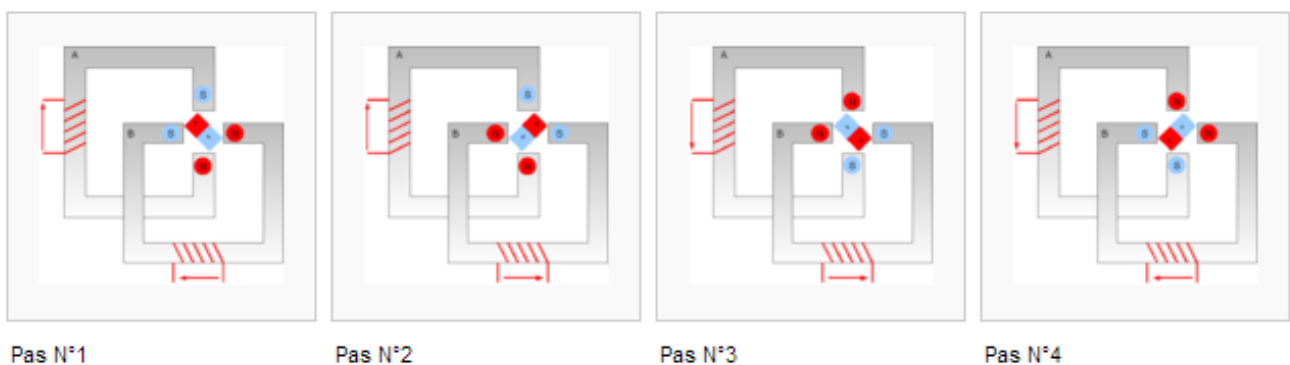
On peut traduire cette activité par le chronogramme suivant :



Comme vous pouvez le voir à travers ces différents moyen d'explication, c'est somme toute assez simple. On va chercher à déplacer l'aimant central en le faisant tourner petit à petit. Pour cela on cherchera à l'attirer dans différentes positions.

Rotation à couple maximal

Un autre mode de fonctionnement est celui dit à **couple maximal**. Cette méthode de pilotage utilise toutes les bobines à la fois pour pouvoir immobiliser au maximum l'aimant central. En effet, en utilisant plus de champs magnétique on obtient une force supplémentaire. Par contre on consomme évidemment d'avantage de courant. Pour comprendre ce fonctionnement, voyons les différentes étapes par un dessin puis par un chronogramme. Vous verrez, ce n'est pas très compliqué, le fonctionnement est très similaire, seule les activations de bobines changent un peu :



Avez-vous remarqué quelque chose de particulier ? Dans cette utilisation, l'aimant ne fait plus face aux bobines mais se place *entre* les deux. Par contre, il effectue toujours des pas entiers, ces derniers ont juste un décalage constant par rapport à avant.

Voici le tableau correspondant au pilotage des bobines :

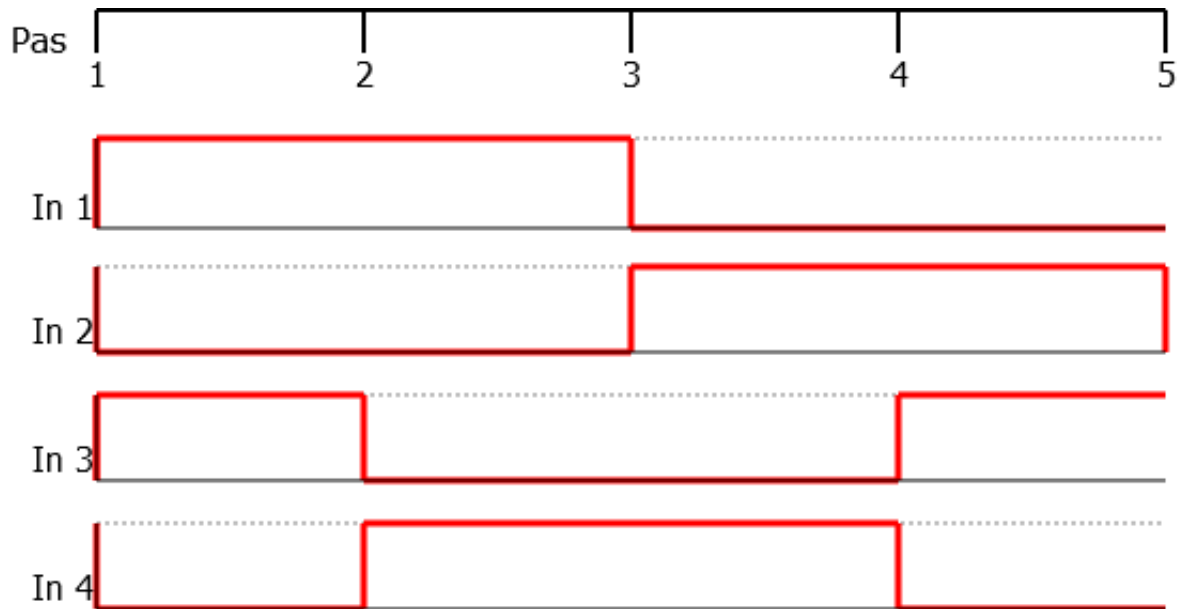
Etape	In 1	In 2	In 3	In 4
Pas n°1	HIGH	LOW	HIGH	LOW

Pas n°2 HIGH LOW LOW HIGH

Pas n°3 LOW HIGH LOW HIGH

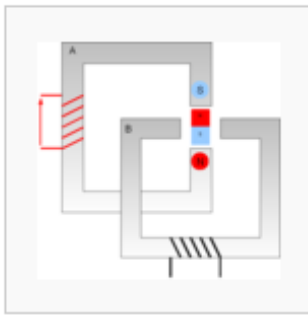
Pas n°4 LOW HIGH HIGH LOW

(un état '-' signifie "non nécessaire", placez le à 0V pour que la bobine soit bien inactive).

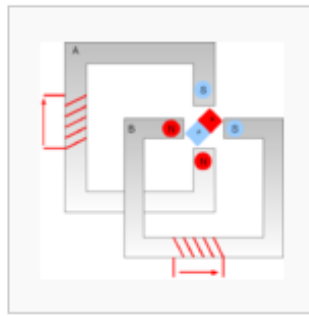


Rotation par demi-pas

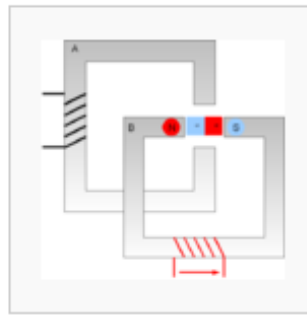
Enfin, le dernier mode de fonctionnement est celui dit à **demi-pas**. Ce mode mélange les deux précédents puisqu'on va alterner les étapes du mode à pas complet et les étapes du mode à couple maximal. En effet, comme nous avons pu le voir dans les explications précédentes, les deux modes placent l'aimant central de manière différente. L'un est "en face des bobines" alors qu'avec l'autre est plutôt "entre les bobines". Ainsi, en se mettant alternativement "en face" puis "entre" les bobines on va effectuer deux fois plus de pas que précédemment puisqu'on intercalera des étapes supplémentaires. Attention, lorsque je dit "deux fois plus de pas" je veux surtout dire que l'on aura des étapes intermédiaires qui augmentent la précision du déplacement. Ce mode de pilotage est un peu plus compliqué que les précédents puisqu'il est "plus long" (8 étapes au lieu de 4) mais rien d'insurmontable vous allez voir !



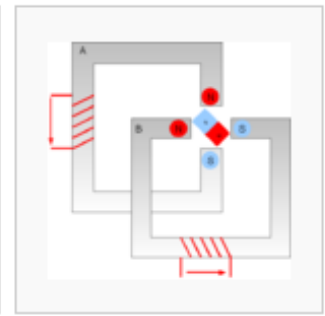
Pas N°1



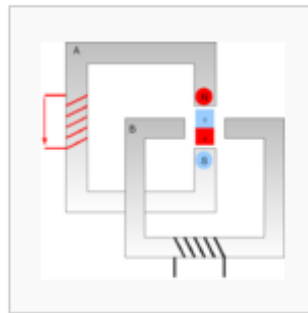
Pas N°2



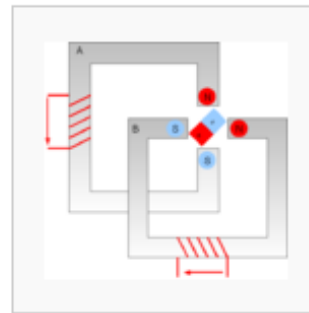
Pas N°3



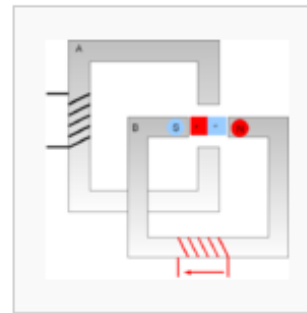
Pas N°4



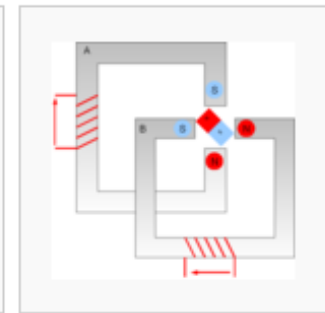
Pas N°5



Pas N°6



Pas N°7

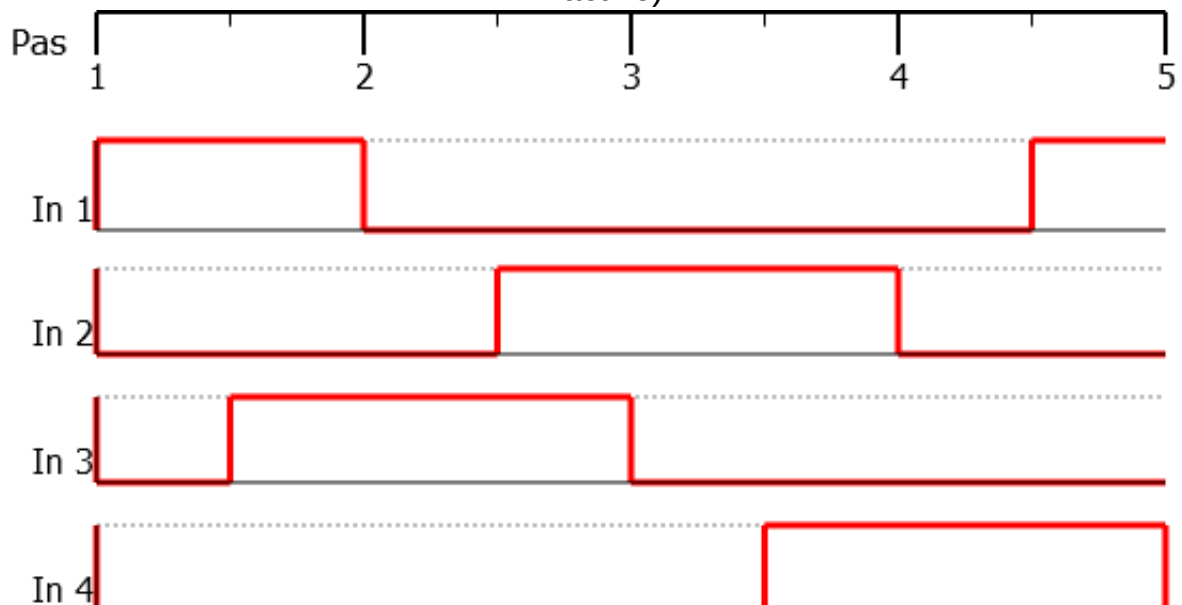


Pas N°8

Etape In 1 In 2 In 3 In 4

Pas n°1	HIGH	LOW	-	-
Pas n°1 ½	HIGH	LOW	HIGH	LOW
Pas n°2	-	-	HIGH	LOW
Pas n°2 ½	LOW	HIGH	HIGH	LOW
Pas n°3	LOW	HIGH	-	-
Pas n°3 ½	LOW	HIGH	LOW	HIGH
Pas n°4	-	-	LOW	HIGH
Pas n°4 ½	HIGH	LOW	LOW	HIGH

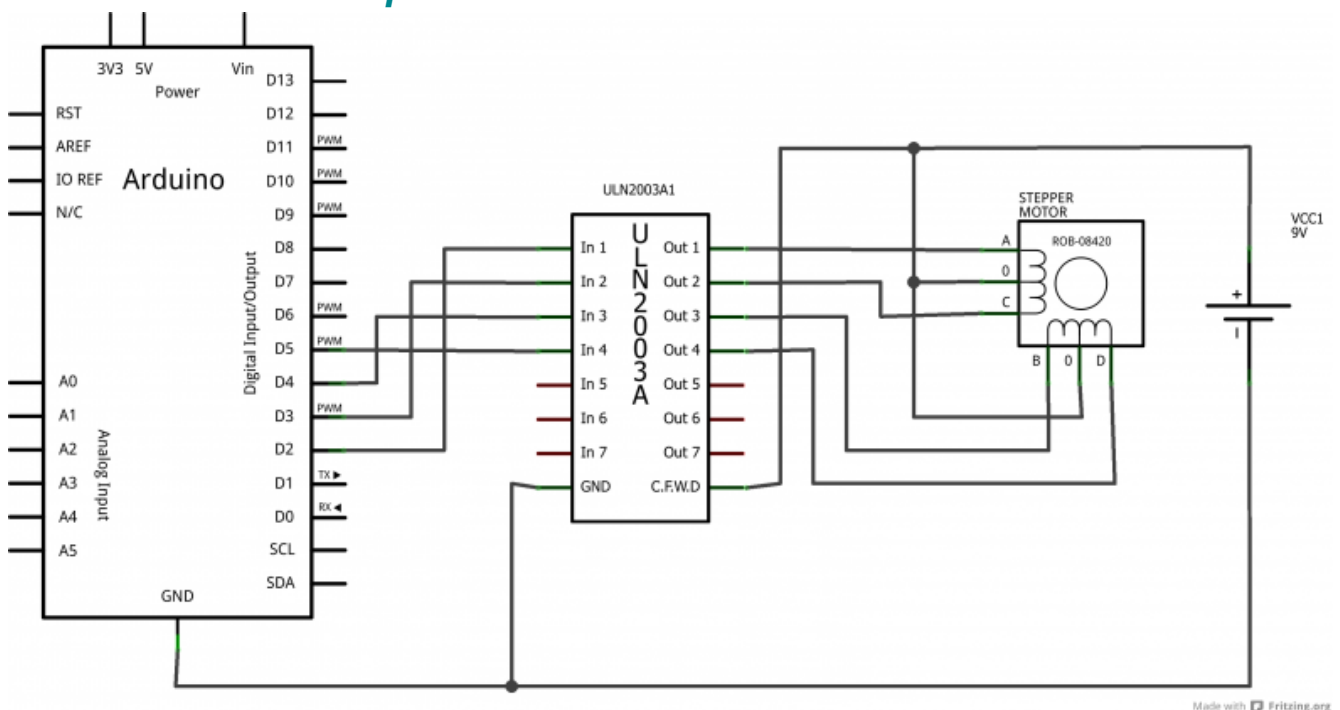
(un état '-' signifie "non nécessaire", placez le à 0V pour que la bobine soit bien inactive).

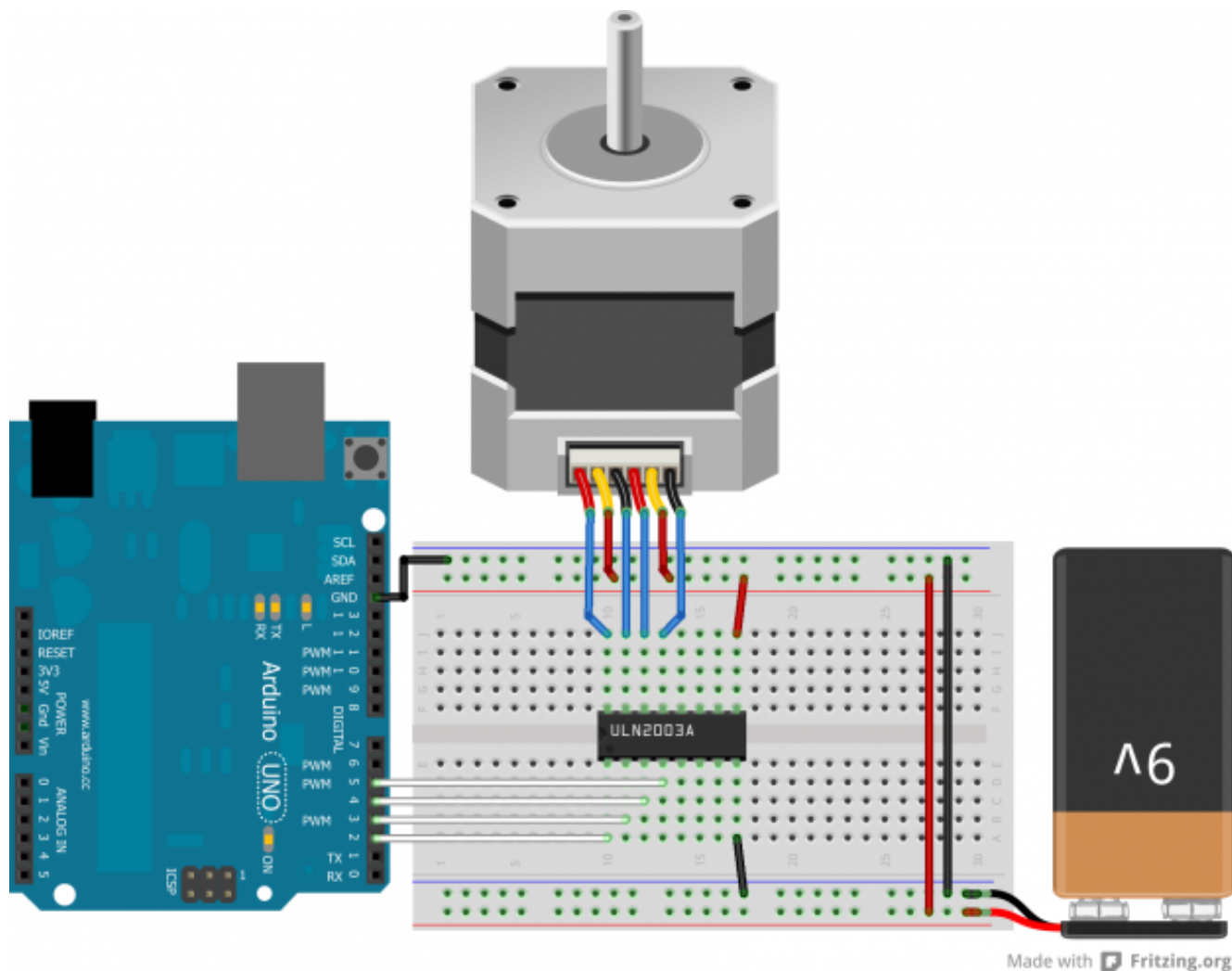


Si vous avez une charge qui demande trop de couple (par exemple un poids à faire monter), il peut arriver que le moteur “saute” un/des pas. Cette donnée est à prendre en compte si vous vous servez du nombre de pas effectué logicielllement comme moyen de calcul de distance.

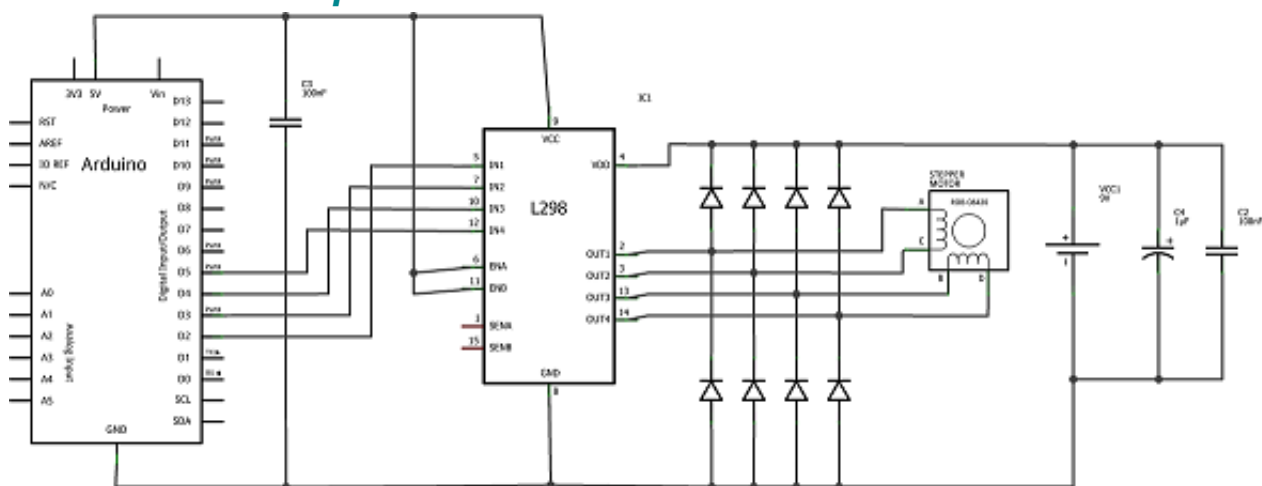
Câbler les moteurs

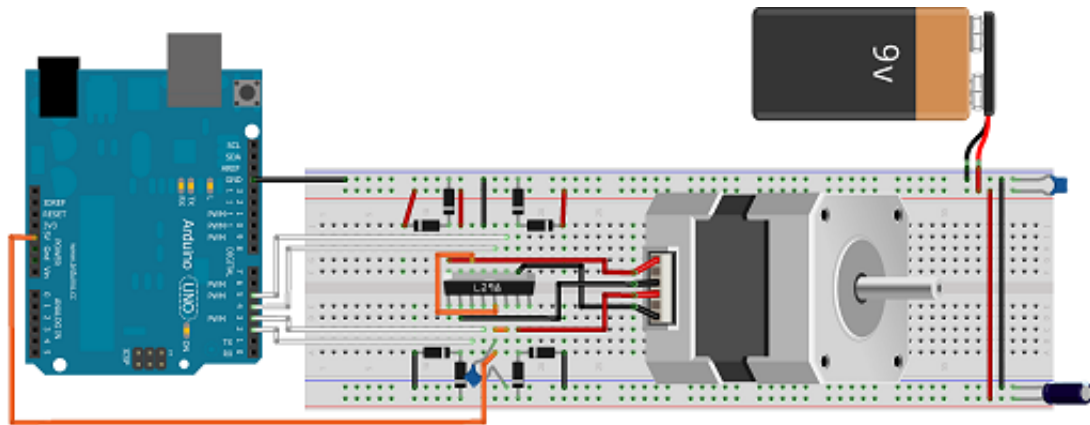
Le moteur unipolaire





Le moteur bipolaire





Jusque là rien de vraiment compliqué, on passe à la suite !

Piloter les moteurs avec Arduino

Le principe

L'idée est toute simple, il suffit de générer la bonne séquence pour piloter les moteurs à la bonne vitesse, vous vous en doutez surement. La principale difficulté réside dans la génération des signaux dans le bon ordre afin que le moteur se déplace correctement. Bien entendu, c'est plus facile à dire qu'à faire. En effet, pour que le mouvement soit fluide, il faut que les changements dans la séquence soient fait de manière régulière et pour cela il faut une gestion du temps correcte. Ça peut sembler simple au premier abord, mais quand il s'agira de mixer le comportement du moteur avec celui du programme principal (qui devra peut-être faire des traitements assez lourd) cela deviendra probablement beaucoup moins trivial. Une bonne méthode consisterait à utiliser un **timer** sur lequel on réglerai la période à avoir qui refléterais ainsi la vitesse à obtenir. Mais avec Arduino vous allez voir que tout devient plus simple...

L'objet Stepper

Sur Arduino les choses sont bien faite pour rester simples et accessibles, un objet a déjà été créé pour vous aider à piloter un moteur pas à pas. Attention cependant, il ne fonctionne que pour les moteurs unipolaire et bipolaire. Il tire partie du fait que ces deux types de moteur peuvent fonctionner avec une séquence commune. Ainsi, tout est généralisé et utilisable le plus simplement possible ! Ce nouveau composant s'appelle "Stepper". A sa création il prend en argument le nombre de pas total que fait le moteur pour faire un tour (information trouvable dans la documentation constructeur ou empiriquement). Cette information sert à déterminer la vitesse de rotation par minute que vous pourrez ensuite régler à loisir pour faire des déplacements lents ou rapides. Il prend aussi en arguments les quatre broches servant à contrôler l'engin. Son constructeur est donc : Stepper(steps, pin1, pin2, pin3, pin4). Pour initialiser le moteur, nous pouvons donc écrire la ligne suivante :

```
1 //pour un moteur de 200 pas par tour et brancher sur les broches 2, 3, 4, 5
2 Stepper moteur(200, 2, 3, 4, 5);
```

Pour l'utiliser, deux fonctions sont utilisables. La première sert à définir la vitesse de rotation, exprimée en tours par minute (**trs/min**). Pour cela, on utilise la fonction step(steps) qui prend en paramètre le nombre de pas à effectuer. Si ce nombre est négatif, le moteur tournera en sens inverse du nombre de pas spécifié. Voici un petit

exemple qui va faire faire un aller-retour de 200 pas toute les 2 secondes à votre moteur :

```
1 #include <Stepper.h>
2
3 //pour un moteur de 200 pas par tour et brancher sur les broches 2, 3, 4, 5
4 Stepper moteur(200, 2, 3, 4, 5);
5
6 void setup()
7 {
8   moteur.setSpeed(30); //30 tours par minute
9   //(rappel : ici le moteur fait 200 pas par tour, on fera donc 6000 pas par minute)
10 }
11
12 void loop()
13 {
14   moteur.step(1000);
15   delay(100);
16   moteur.step(-1000);
17   delay(2000);
18 }
```

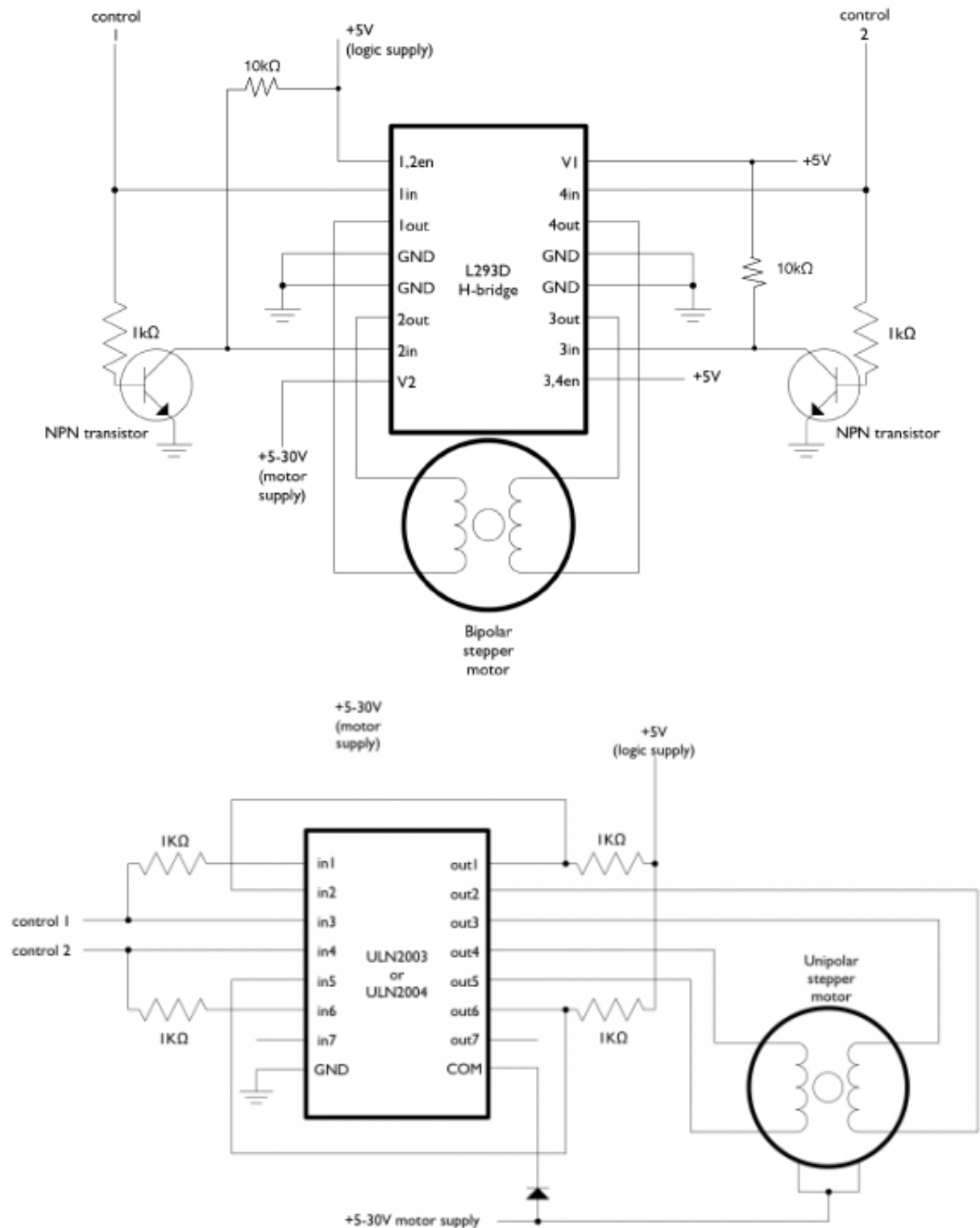
La fonction step(x) est bloquante. Cela signifie qu'elle agit comme un délai. Tant que le moteur n'a pas fait les x pas demandés, le reste du programme est en attente.

Aller plus loin

Vous êtes devenus incollables sur les moteurs pas à pas ? Vous en voulez encore plus ? Suffit de demander, voilà du bonus d'informations rien que pour toi chère lecteur !

2 fils au lieu de 4 !

On a toujours utilisé 4 fils pour commander les moteurs. C'est bien, mais que diriez-vous de sauver deux broches et de passer à seulement 2 fils au lieu de 4 ? Pas mal comme amélioration non ? Petite anecdote : un jour, un utilisateur des moteurs pas à pas s'est rendu compte d'un truc, dans une paire de fils pour piloter un moteur (dans la séquence utilisée par Arduino), l'information est toujours antagoniste. Si un fil est à HIGH, sa paire sera à LOW et vice versa. Du coup il suffit d'un peu d'électronique pour pouvoir inverser un des deux signaux et se retrouver ainsi avec seulement deux fils au lieu de 4 sortant d'Arduino. 😊



(source des images : tigue.net) Cette solution est plutôt intéressante du fait que les entrées/sorties sont parfois une denrée rare sur Arduino ! 🇵🇷

Le L297

Lorsque vous utilisez votre Arduino, vous ne pouvez utiliser qu'une seule séquence. Par exemple pour un moteur bipolaire vous n'avez pas le choix entre le mode pas entier, demi-pas ou couple max. Une des solutions serait de générer vous-même les séquences. Mais c'est assez fastidieux. Une autre solution est électronique et compensera le développement informatique à faire. Un composant, nommé L297 (de la

famille du L298 vous vous en doutez) est justement fait pour générer les séquences de moteur pas à pas. Il possède 4 broches de sorties pour générer la séquence et plusieurs en entrée pour “paramétrer” le fonctionnement voulu. Parmi elles on en retrouve trois principales :

- CW/CCW : (ClockWise ou Counter ClockWise) qui décidera du sens de rotation du moteur (horaire ou anti-horaire).
- Half/Full : Qui décide si on est en mode pas entier ou demi-pas.
- Clk : (Clock) qui est l’horloge pour la vitesse. A chaque front descendant le moteur fera un pas.

Je vous laisse un peu chercher sur le net, vous trouverez de plus amples informations à ce sujet. Avant même de regarder sur le net, en fait, regardez plutôt sa datasheet !! 😊
Un des avantages de délester le travail des séquences au L297 est que vous n’aurez plus besoin de l’objet Stepper et de sa fonction step() bloquante. Il faudra cependant toujours utilisé un composant de puissance pour laisser passer les forts courants nécessaires au moteur (comme le L298 par exemple).
